

# Introduction

## Contexte et enjeux

Le nombre grandissant des plateformes telles que Smartphone, tablette, table interactive, etc. a vulgarisé l'usage des applications ayant des modalités d'interactions nouvelles par rapport aux claviers et à la souris. Les tables interactives par exemple offrent la possibilité d'écrire des interfaces utilisateurs tangibles et ils permettent aussi l'utilisation par plusieurs personnes d'une interface utilisateur (UI). La conception des UI des applications pour ces plateformes implique de prendre en compte les règles ergonomiques liées à ces plateformes. En ce qui concerne les applications conçues pour d'autres plateformes, leur réutilisation contraint les concepteurs à les adapter aux règles ergonomiques de la plateforme d'arrivée. Cette réutilisation peut se faire par la migration des applications.

La migration des applications est un problème qui est traité de diverse manière, la portabilité des logiciels telle que définie par Tanenbaum et al. [TKB78] permettait déjà de migrer facilement les applications, l'utilisation des patrons de conception et des architectures permettant une séparation entre l'interface utilisateur (UI) et le noyau fonctionnel(NF) permet la conception des applications facilement réutilisable. L'ingénierie dirigée par les modèles préconise des modèles et des transformations [FBBN07] qui permettent de migrer des applications. L'ensemble de ces approches de migrations offrent aux concepteurs des outils, des modèles et mécanismes de migrations du NF et de l'UI qui requiert un savoir et une expérience.

L'objectif de cette thèse est de faciliter le travail du concepteur d'UI en lui permettant de migrer une application vers une table interactive en se concentrant que sur l'adaptation de UI. Comme la conception assistée d'UI [Van97], la migration assistée d'UI permet aux concepteurs la mise en ?uvre d'UI en respectant les critères ergonomiques. En effet les étapes du processus qui font intervenir le concepteur doivent proposer une assistance soit en recommandant des choix conformes aux principes de conception dans le cas où le concepteur a plusieurs options, soit en indiquant les critères ergonomiques et les principes de conceptions

correspondants à une étape du processus de migration, soit en vérifiant la conformité d'une étape aux critères ergonomiques de la plateforme d'arrivée.

## Contribution de la thèse

Nos travaux de recherche se situent dans plusieurs domaines de recherches que sont : le domaine des IHM, le domaine de l'utilisabilité des interfaces utilisateurs et le domaine de l'Ingénierie de modèles. Dans le domaine des IHM, on s'intéresse à la migration des interfaces utilisateurs tout en prenant compte les critères ergonomiques de la plateforme d'arrivée. Dans le domaine de l'utilisabilité, on s'intéresse aux travaux qui modélisent les critères ergonomiques en règles opérationnelles et utilisable pendant la conception. Dans le domaine l'Ingénierie des modèles, nous travaillons à un niveau abstrait dans le but de faciliter la réutilisation des nos travaux. [Todo]

## Plan du manuscrit

Le chapitre 1 fait une étude des tables interactives en illustrant les problématiques liées à la migration des applications par un exemple d'application. L'objectif est de migrer une application desktop utilisée avec un clavier et une souris par une personne vers la table interactive en prenant compte l'utilisation des objets tangibles comme moyens d'interactions et l'utilisation par plusieurs personne de l'UI à générer.

Le chapitre 2 est un état de l'art des approches et des modèles permettant la migration d'UI indépendamment des plateformes de départ et d'arrivée. Ces approches comportent plusieurs phases et nous étudions les phases d'adaptation assistée ou non. Par ailleurs nous présenterons aussi les modèles utilisés par chaque approche de migration et discuterons de la possibilité d'en tenir compte dans la solution à proposer. Ce chapitre se termine par une synthèse des différentes approches présentées.

Le chapitre 3 fait partie de la contribution de cette thèse car il propose une modélisation des interactions et de la structure de l'UI pour le processus de migration assistée vers une table interactive. Le chapitre 6 décrit le moteur de suggestion et d'adaptation utilisé par le processus de migration pour assister le concepteur. Il présente aussi de manière détaillée les différentes phases du processus de migration et la prise en compte des guidelines à chaque étape du processus.

Le chapitre 4 présente les transformations qui permet la de prendre en compte les guidelines. **[Todo]**

Le chapitre 5 est une preuve de concept et évaluation **[Todo]**

Le chapitre 6 est une conclusion générale qui présente les contributions et les perspectives de cette thèse. **[Todo]**

## **Première partie**

### **Domaine d'étude et état de l'art**

# Chapitre 1

## Tables interactives et Migration d'UI

### Sommaire

---

<b>1.1</b>	<b>Motivations . . . . .</b>	<b>6</b>
1.1.1	Cas de l'application CBA . . . . .	7
1.1.2	Problématiques liées à la migration des UI . . . . .	7
1.1.3	Problématiques liées à la migration d'UI vers une table interactive . . . . .	9
<b>1.2</b>	<b>Spécificités de la migration d'UI vers les tables interactives . . . . .</b>	<b>10</b>
1.2.1	Modèle d'interactions d'une table interactive . . . . .	11
1.2.2	Propriétés caractéristiques du modèle d'interactions des tables interactives . . . . .	19
1.2.3	Principes de conception d'UI pour les tables interactives . . . . .	20
<b>1.3</b>	<b>Concepts pour la migration d'UI . . . . .</b>	<b>27</b>
1.3.1	Architecture des applications à migrer . . . . .	27
1.3.2	Modèle d'interactions abstraites . . . . .	31
<b>1.4</b>	<b>Conclusion . . . . .</b>	<b>33</b>

---

Dans le but de motiver la migration des applications vers des tables interactives, ce chapitre s'appuie sur un exemple d'UI d'application conçue pour un desktop (décrit à la section 1.1) pour identifier les problématiques liées à la migration d'UI. Ensuite la section 1.2 étudie les éléments caractéristiques des tables interactives dans l'objectif d'identifier des principes qui vont guider la migration d'une UI vers ces tables interactives. La section 1.3 quant à elle aborde des concepts utiles pour la migration des UI. Enfin la section 1.4 présente une synthèse des problématiques principales étudiées dans ce manuscrit.

## 1.1 Motivations

Les tables interactives comme les desktops et les smartphones sont utilisées dans plusieurs domaines d'activités. La figure 1.1 présente une répartition des applications conçues pour les tables interactives en 2011 [Kub11]; la musique, la photo et les jeux constituent environ 55% des domaines d'utilisation des tables interactives. La migration des applications existantes vers les tables interactives présente un intérêt car il existe une multitude d'applications de musique, photo et jeux pour desktop. Par ailleurs, il existe un faible pourcentage d'applications sur les tables interactives pour des domaines tels que la cartographie(6%) ou le brainstorming(2%) qui peut être développé par la migration de ces applications vers une table interactive. De manière globale, les applications sont migrées d'un desktop ou d'un smartphone vers une table interactive dans le but d'avoir un contexte collaboratif, de bénéficier des moyens d'interactions tangibles ou d'une surface d'affichage plus grande.

Dans cette section nous présentons (au paragraphe 1.1.1) un cas d'application desktop à migrer vers une table interactive, ensuite les problématiques générales liées à la migration d'UI des applications existantes (au paragraphe 1.1.2 ) et enfin les questions soulevées par la migration d'une UI vers les tables interactives (au paragraphe 1.1.3 ).

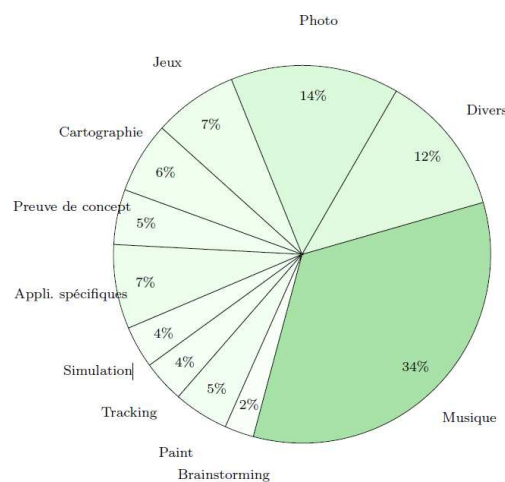


FIGURE 1.1 – Répartition des domaines d'utilisation des tables interactives en 2011

### 1.1.1 Cas de l'application CBA

Considérons le cas d'étude d'une application conçue pour un desktop qui permet l'élaboration des bandes dessinées (BD) telle que *Comics Book Application* (CBA). Cette application est conçue pour être utilisée par un dessinateur de BD qui est assis devant son écran en utilisant sa souris et son clavier. La fenêtre principale de l'application CBA décrite par la figure 1.2 nous permet d'identifier trois zones majeures que sont la zone des menus correspondant aux composants graphiques situés en haut de la fenêtre principale, l'espace de travail qui contient les cadres d'une page de bande dessinée et la zone de légende correspondant aux groupes, aux formulaires et listes à gauche de la fenêtre.

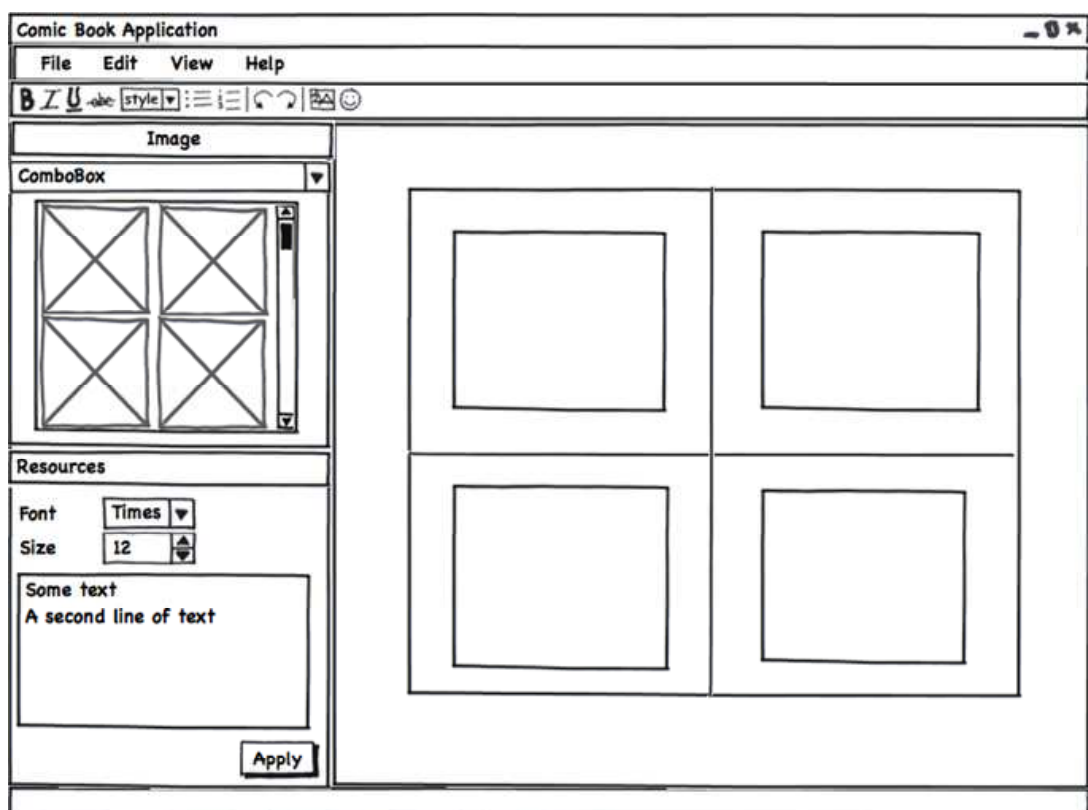


FIGURE 1.2 – Fenêtre principale de l'application CBA

### 1.1.2 Problématiques liées à la migration des UI

La migration des UI des applications existantes vers une nouvelle plateforme tout en conservant le NF peut se faire en concevant une nouvelle UI pour la plateforme en se basant sur le NF et sur les spécificités de la plateforme d'arrivée. Il est aussi possible de migrer une

UI en l'adaptant à sa cible en tenant compte ou non des spécificités de cette cible. Ces deux types de migration d'UI (adaptation ou nouvelle conception) soulèvent plusieurs problèmes.

### **Problématiques liées à une reconception de l'UI**

La reconception de l'UI suppose de s'appuyer sur le NF de l'application de départ, particulièrement sur les liens entre ce NF et l'UI à migrer et sur les spécificités de la plateforme d'arrivée (les dispositifs d'interactions, les bibliothèques graphiques, les critères ergonomiques, etc.). En effet, il est possible de déduire une UI en se basant sur les liens entre le NF et l'UI, il existe des travaux qui préconisent la génération des UI à partir des descriptions des web services par exemple[KKM03]. La conception de l'UI en se basant sur le NF suppose de savoir comment déduire une UI à partir d'une description abstraite du NF :

- les composants graphiques qui composent l'UI,
- la structure et le layout de ces composants graphiques,
- le comportement de l'UI (c'est-à-dire l'enchaînement des fenêtres et des activités de l'UI),
- et évaluer le respect des critères ergonomiques de l'UI générée.

### **Problématiques liées à une adaptation de l'UI**

L'adaptation de l'UI pendant la migration suppose de s'appuyer sur les spécifications de l'UI à migrer décrites par des modèles abstraits et d'adapter ces modèles à la plateforme d'arrivée. Les modèles servent à décrire les différents aspects de l'UI tels que les tâches ou les activités décrites par l'UI, le placement ou le layout des éléments de l'UI, les interactions entre l'utilisateur et l'UI, les styles de présentations des aspects visuels (tailles et couleurs des textes, etc.) Le problème lié à la migration sont est : comment transformer les différents aspects d'une UI pour qu'ils soient conforme aux spécificités de la plateforme cible ? En effet la migration d'une UI desktop vers un smartphone peut entrainer le changement de layout et/ou du placement des éléments de l'UI pour adapter l'UI de départ à la taille du smartphone, où la différence des moyens d'interactions entre un desktop et une table interactive implique une adaptation du modèle d'interaction de l'UI de départ par exemple.

De manière générale, les migrations d'UI basées sur l'adaptation de l'existant évoquent des problèmes liées aux transformations des différents aspects de l'UI pour la plateforme d'arrivée.



### 1.1.3 Problématiques liées à la migration d'UI vers une table interactive

Dans le cas spécifique d'une migration d'UI vers une table interactive, les problématiques évoquées à la section 1.1.2 peuvent être précisées d'avantages. En considérant l'application CBA (cf section 1.1.1), la migration de son UI sur une table interactive se fait en soulevant les questions suivantes :

- comment placer et organiser les éléments de l'UI de départ pour une table interactive ?

En effet, la fenêtre principale de l'application CBA est conçue suivant la métaphore de bureau qui permet de décrire les applications pour les ordinateurs personnels [App95]. Les menus sont placés en haut et à des positions fixées, les outils ou les légendes sont toujours placés à gauche ou à droite et la zone de travail au centre. Cette structuration des zones permet aux utilisateurs des desktops de se retrouver facilement quelque soit l'application.

Sur une table interactive, cette structuration n'est pas recommandée car elle ne permet pas une utilisation par plusieurs utilisateurs et elle fixe l'orientation des composants graphiques de chaque zone. Pour l'application CBA, les différentes zones de la structure de départ peuvent être conservées mais chaque zone n'est plus associée à un espace géographique spécifique de l'écran. Sur la droite de la figure 1.3, les différentes zones de l'UI CBA de départ n'ont plus de position fixe sur l'UI CBA de la table interactive à droite de la même figure.

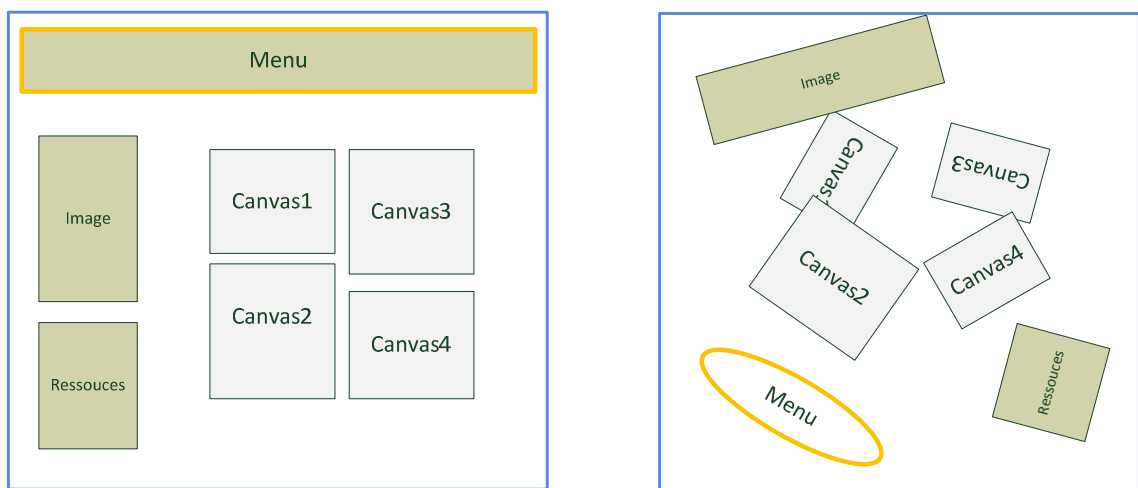


FIGURE 1.3 – Migration de la structure d'une UI Desktop sur une table interactive

- comment utiliser les interactions (tactiles, tangibles) d'une table interactive avec l'UI de départ ?
- comment prendre en compte la taille de la surface d'affichage d'une table interactive ?
- comment prendre en compte le nombre d'utilisateurs pendant la migration de l'UI de CBA ? Cette problématique consiste aussi à transformer une UI mono utilisateur en UI

collaborative dans le cas où le nombre d'utilisateur est supérieur à un. Le projet Co-Word [XSS<sup>+</sup>04] permet à plusieurs utilisateurs d'utiliser le logiciel Microsoft Word en même temps mais pas sur un même support, en effet les utilisateurs sont dispersés géographiquement et ils ont chacun un ordinateur de bureau. Dans notre cas les utilisateurs devront partager une même surface d'affichage qui est l'écran de la table interactive ciblée.

Dans la section suivante, nous détaillerons ces problématiques en présentant les spécificités des tables interactives à travers leur modèle d'interaction et leurs principes de conception des UI.

## 1.2 Spécificités de la migration d'UI vers les tables interactives

Les tables interactives sont des surfaces qui offrent la possibilité d'interagir avec des systèmes interactifs à travers des UI. Les spécificités d'interactions entre les utilisateurs et ces tables interactives peuvent être décrites avec des interactions instrumentales [BL00]. Une interaction instrumentale est une action d'un utilisateur à l'aide de dispositifs physiques (écran tactile par exemple) et de composants graphiques (boutons, scrolls par exemple) pour modifier ou accéder à des objets d'un domaine (images, données numériques, etc.). Ensuite une interaction en sortie est une réponse traduite en une réaction ou un feedback par les instruments d'interaction en sortie. La figure 1.4 montre les différents éléments de ce modèle d'interaction, les instruments servant de moyen d'interactions pour l'utilisateur sont constitués de l'écran tactile et des composants graphiques. La migration d'une UI décrite suivant un autre modèle d'interactions vers ce modèle implique l'adaptation des objets du domaine et des interactions de l'UI de départ aux nouveaux instruments qui constituent la plateforme d'arrivée. L'adaptation des différents aspects de l'UI de départ à la table interactive, se fait suivant un ensemble de principes et des recommandations liés aux tables interactives, dans l'objectif d'avoir une UI respectant ses critères ergonomiques.

Dans l'objectif d'identifier et de caractériser les principes de conception des UI pour les tables interactives, cette section étudie les différents éléments qui composent le modèle d'interaction d'une table interactive (instruments matériels, instruments logiciels, représentation des interactions en entrée et en sortie entre les utilisateurs et les objets du domaine). Elle présente l'ensemble des principes de conception des UI liées aux tables interactives nécessaires pour la migration.

### 1.2.1 Modèle d'interactions d'une table interactive

Les instruments d'interactions constituent l'ensemble des dispositifs matériels et logiciels d'une table interactive qui permettent d'interagir avec un SI. Les dispositifs matériels d'interactions sont des moyens d'interactions en entrée (actions) ou en sortie (réactions et feedback) cf. figure 1.4. Et les bibliothèques graphiques sont des instruments logiciels pour décrire des interfaces utilisateurs graphiques dans le cadre des tables interactives. Dans ce paragraphe nous nous appuierons sur des tables interactives (Microsoft Surface, TangiSense, DiamondTouch, etc.) pour caractériser les dispositifs matériels et d'interaction, les bibliothèques graphiques et les modalités d'interactions des tables interactives.



FIGURE 1.4 – Modèle d'interaction instrumentale d'une table interactive

#### Les dispositifs matériels d'interaction d'une table interactive

Dans ce paragraphe nous étudions les instruments d'interactions de trois tables interactives. Nous étudions d'abord DiamondTouch [DL01] qui est l'une des première table interactive utilisée dans un cadre non expérimental, elle fut industrialisée par MERL [Mit], nous l'étudions car elle comporte une bibliothèque graphique DiamondSpin et permet de décrire des interactions multi utilisateurs, collaboratives et tactiles. Ensuite, l'on s'intéresse à meta-Desk [UI97] qui supporte que des objets tangibles comme moyen d'interaction. Enfin nous nous intéresserons aux tables Microsoft Surface 1.0 et 2.0 [Mic11] pour leurs bibliothèques graphiques et leurs moyens d'interactions, contrairement aux deux autres tables précédentes, les tables Surface supportent à la fois les interactions collaboratives, tangibles et tactiles.

**DiamondTouch** - Les moyens d'interaction de la table DiamondTouch sont une surface tactile pour les interactions en entrée et un vidéo projecteur pour les interactions en sortie. Cette table supporte des interactions multi utilisateurs et elle est capable d'identifier les contacts de chaque utilisateur autour de la table. Elle ne supporte pas les interactions tangibles car elle ne peut pas détecter des objets ou des tags. Cette table permet de décrire des UI collaboratives. Les UI collaboratives pour les tables interactives ont pour objectif de permettre

à plusieurs utilisateurs de partager en même temps une UI. Dans le cadre de la table interactive DiamondTouch les utilisateurs d'une UI collaborative partagent un même espace (la table interactive) et il est possible de créer à chaque utilisateur son espace de travail. Cette solution implique une répartition géographique fixe autour de la table. En effet pour migrer l'UI de l'application CBA par exemple sur une table DiamondTouch, il est indispensable de connaître le nombre d'utilisateurs de l'UI migrée car chaque utilisateur doit être assis sur une chaise pour utiliser la table(cf. figure 1.5). DiamondTouch ne permet aux utilisateurs d'être debout pendant l'utilisation de la surface d'affichage.

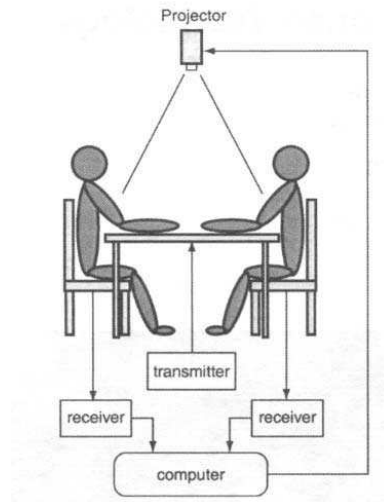


FIGURE 1.5 – Table interactive DiamondTouch

**metaDESK** - Les moyens d'interaction de la table metaDESK sont constitués des objets tangibles et de caméra infrarouge. Ces objets permettent à chaque utilisateur de manipuler des objets virtuels associés aux objets physiques. Elle ne limite pas le nombre d'utilisateurs comme la table DiamondTouch et permet à chaque utilisateur d'être mobile autour de la table. Cette Table permet de décrire des UI tangibles (TUI).

Ulmer et Ishii [IU97] définissent un TUI comme des systèmes interactifs qui utilisent des objets physiques pour représenter et utiliser des informations digitales. Le mapping entre le monde physique et digital peut se faire en représentant les différents composants graphiques d'une UI graphique à l'aide des objets concrets. La conception de TUI pour la table metaDESK [IU97], les concepteurs associent une lentille à une fenêtre, un plateau à un menu, etc. comme l'indique la figure 1.6 d'instanciation physique des éléments GUI de metaDESK de la figure 1.6. Ce type d'association permet de concrétiser des composants graphiques virtuels à travers des objets physiques.

Dans le cadre de la migration d'une UI desktop vers la table metaDESK, toutes les inter-

actions de l'UI de départ doivent être adaptées ou émulées avec des objets tangibles. En considérant l'UI de l'application CBA par exemple, le formulaire *Ressources* doit être affiché et rempli avec des objets tangibles, la sélection de la taille ou de la police peuvent être émuler avec des objets circulaires en les tournant par exemple. Par ailleurs l'émulation d'un clavier à l'aide d'un objet tangible pour la saisie des textes est possible mais n'est pas facilement utilisable [?]. Les tables interactives uniquement tangibles comme metaDESK ou TangiSense [KLL<sup>+</sup>09] sont en générale conçues pour faire de la réalité augmentée pour des applications de cartographie par exemple.

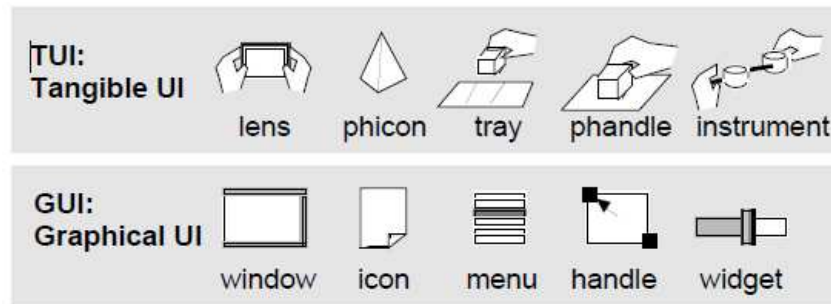


FIGURE 1.6 – Instanciation physique des éléments GUI dans TUI

**Microsoft Surface** Les moyens d'interaction de la table interactive Microsoft Surface sont un écran tactile permettant la reconnaissance des objets physiques, un clavier virtuel et des dispositifs sonores. Elles permettent de décrire des UI tactiles et tangibles par l'utilisation des tags. Elles supportent 50 contacts simultanés et permet donc de décrire des UI multi utilisateurs dans la limite des contacts supportés. En effet une UI ne peut pas supporter plus de 50 points de contacts simultanés, cette contrainte et la taille de l'écran limite le nombre d'utilisateurs utilisant une application sur cette table au même moment.

Cette plateforme permet la description de TUI par la reconnaissance des Tags et la forme des objets. L'utilisation de tags permet de ne pas limiter les objets à utiliser dans la description des UI. Les tags sont des codes barres à deux dimensions qui sont facilement identifiables par la table surface. La Surface offre deux types tags : Identity tags et Byte tags<sup>1</sup>. Ils peuvent être utilisés pour reconnaître des objets physiques ou les distinguer parmi plusieurs, pour déclencher une commande ou une action (afficher un menu ou une application par exemple), pour pointer et orienter une application par un objet tagué peut être utilisé pour le contrôle de volume car il peut détecter le changement d'angle d'un objet.

1. Les Identity tags sont des tags sur avec une plage de valeurs sur 128 bits, les Byte tags on une plage de valeurs sur 8 bits

**Résumé** Les tables interactives de manière générale ont deux types de dispositifs physiques d'interactions : les dispositifs d'interaction en entrée qui peuvent être tactiles et/ou tangibles et les dispositifs d'interaction en sortie qui sont des surfaces d'affichage. Ces surfaces sont de tailles variables et de différentes formes (rectangulaire, circulaire, etc.). Elles peuvent être disposées de manière horizontale (pour DiamondTouch, TangiSense, Microsoft Surface 1.0 et 2.0) ou de manière verticale (pour Microsoft Surface 2.0, etc. ).

Le nombre d'utilisateurs et leurs dispositions autour de la surface d'affichage sont des éléments qui permettent de caractériser les interactions des tables interactives. En effet ces deux caractéristiques impactent la conception des UI car une UI destinée à plusieurs personnes doit permettre l'accessibilité des différentes fonctionnalités à tous les utilisateurs et elle doit aussi prendre en compte le partage des éléments de l'UI (menus, visualisation des contenus, etc.)

Les dispositifs matériels des tables interactives permettent de décrire des UI multi-utilisateurs, co-localisées, tactiles et tangibles. La migration des UI des applications qui ne prennent pas en compte ces caractéristiques soulève des problématiques liées à la prise en compte des différents moyens d'interactions en entrée et en sortie des tables interactives, du nombre d'utilisateurs et de la disposition des utilisateurs par rapport à la surface d'affichage.
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Bibliothèques graphiques des tables interactives

Les bibliothèques graphiques sont des boîtes à outils logiciels qui contiennent des éléments pour décrire des UI graphiques ; dans le modèle d'interactions des tables interactives, les composants graphiques sont des instruments logiciels qui permettent de faire le lien entre les dispositifs matériels d'interactions et les objets d'un domaine. Elles offrent des composants graphiques adaptés aux moyens d'interactions d'une plateforme spécifique. Dans cette section nous présentons quelques bibliothèques graphiques spécifiques aux tables interactives.

**DiamondSpin** [SVFR04] est une bibliothèque graphique pour table interactive qui offre des composants graphiques adaptés à plusieurs utilisateurs. En effet, elle offre des composants graphiques qui permettent : une manipulation des documents visuels, la manipulation directe des éléments d'UI, la possibilité d'utiliser ses doigts, un stylet ou un clavier comme moyen d'interaction, la rotation des composants graphiques, et la possibilité de créer et de gérer des espaces privés pour chaque utilisateur. Les composants graphiques *DSContainer*, *DSPanel*, *DSWindow*, *DSFrame* (cf. figure 1.7) par exemple permettent d'avoir un container qui regroupe un ensemble de composants graphiques qu'un utilisateur peut déplacer en

fonction de sa position.

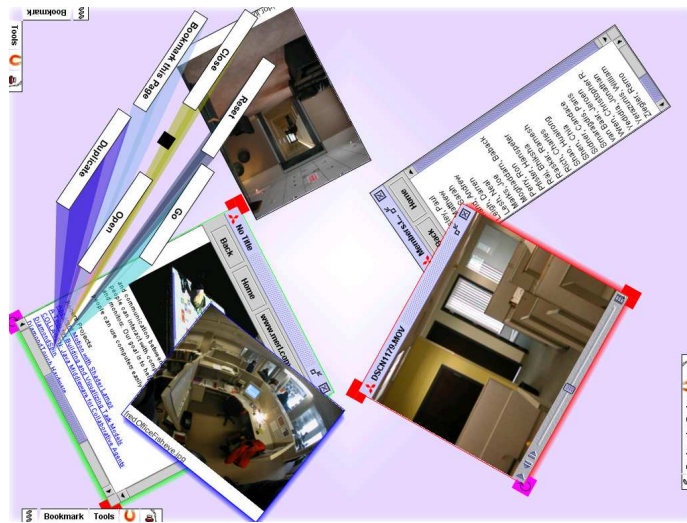


FIGURE 1.7 – Instances des container DiamondSpin

**Surface SDK 1.0 et 2.0** [Mic12a] sont des API et des boîte à outils pour développer des applications pour une Table interactive Microsoft (1.0 et 2.0). Ils font partie du Framework .Net et offrent des bibliothèques graphiques pour concevoir des UI WPF [Mic12b] et XNA [Mic12c]. Ces bibliothèques graphiques offrent des composants graphiques permettant la reconnaissance des formes d'objets, l'utilisation des tags, 50 points de contacts simultanés, la détection de l'orientation des touches, etc. Ces facilités par rapport à la bibliothèque graphique DiamondSpin évitent au programmeur la gestion du déplacement ou la rotation des composants graphiques par exemple. En effet un *ScatterView* (Figure 1.8) définit le déplacement ou la rotation de manière intrinsèque.

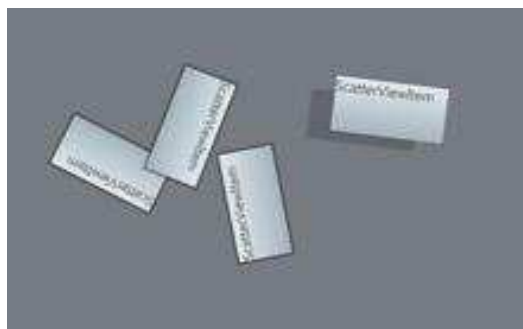


FIGURE 1.8 – Exemple de ScatterView



**Résumé** La bibliothèque DiamondSpin permet d'utiliser la table DiamondTouch à l'instar d'un bureau virtuel par plusieurs personnes assises autour de la table. Elle offre aussi des composants graphiques basés sur la métaphore du papier [BRNB07]. La métaphore du papier permet l'utilisation des éléments graphiques d'un container comme une feuille de papier en ayant la possibilité de plier, retourner, dupliquer ou déchirer le container, sur une table interactive cette métaphore permet de conserver les actions qu'on réalise sur une table de travail concrète.

La bibliothèque Surface SDK permet de décrire des TUI à l'aide des tags. De manière générale, les bibliothèques graphiques des tables interactives offrent des composants graphiques qui ont des interactions (rotation, redimensionnement, déplacement, tags, etc.) adaptées pour la description des UI pour les tables interactives. Elles permettent d'affiner les caractéristiques des tables interactives en précisant si une table interactive supporte ou non des interactions tangibles, si elle a des composants graphiques accessibles par tous les utilisateurs par exemple.

Dans le cadre de la migration, le changement de bibliothèque graphique implique une nouvelle conception de l'UI de départ pour rendre accessible ses fonctionnalités sur la plateforme d'arrivée en utilisant les composants graphiques adaptés aux dispositifs physiques de la table interactive.

## Modalités d'interactions

Nigay [NC94] propose une définition de la modalité d'interaction comme un couple  $\langle d, l \rangle$  constitué d'un dispositif d'interaction et d'un langage d'interaction [NC94] : -  $d$  désigne un dispositif physique (par exemple, une souris, une caméra, un écran, un haut-parleur), -  $l$  dénote un système représentationnel, c'est-à-dire un système conventionnel structuré de signes assurant une fonction de communication (par exemple, un langage pseudo naturel, un graphe, une table).

La migration des UI vers les tables interactives implique un changement des dispositifs d'interactions et la possibilité de décrire des équivalences entre les modalités d'interactions des plateformes de départ et celles des tables interactives.

Dans cette section nous abordons la problématique liée aux changements des modalités d'interactions des UI à migrer. En effet comment décrire les interactions de l'UI de départ à l'aide des modalités d'interactions de la plateforme d'arrivée ? Pour répondre à cette question nous étudions au paragraphe 1.2.1 les problèmes liés aux changements de modalités d'interactions pendant la migration. Et au paragraphe 1.2.1 nous présentons quelques modèles



d'interactions abstraite qui permet de décrire les interactions indépendamment des modalités d'interactions.

**Changement de modalité d'interaction** - Considérons comme plateforme de départ un desktop composé d'un écran comme moyen d'interaction en sortie et d'un clavier et d'une souris comme moyen d'interaction en entrée.

- La modalité d'interaction en sortie M1 du desktop est décrite par l'écran et par le langage de description des UI 2D (L1), M1=<Ecran, L1>. Le langage L1 correspond aux composants graphiques tels que labels, champ de texte, boîte de dialogue, etc. d'une bibliothèque graphique.
- La modalité d'interaction en entrée M2 du desktop est décrite par le clavier et par le langage des commandes (L2), M2=<Clavier, L2>. Le langage L2 décrit pour chaque actions les tâches réalisables à l'aide d'un clavier sur une UI graphique, ces actions sont par exemple : copier avec Ctrl+C, couper avec Ctrl+X, coller avec Ctrl+V, saisir un texte avec les touches alphanumériques, valider avec la touche entrée, etc.
- La modalité d'interaction en entrée M3 du desktop est décrite par la souris et par le langage de manipulation directe d'une UI 2D (L3), M3=<Souris, L3>. Le langage L3 décrit les actions réalisables à l'aide d'une souris sur une UI graphique, ces actions sont par exemple : cliquer et valider pour sélectionner un composant graphique, cliquer et déplacer pour sélectionner un texte, déplacer un élément, redimensionner, etc.

Et considérant maintenant que la table interactive cible est composée d'un écran tactile, d'un clavier virtuel et de la reconnaissance des objets physiques comme moyens d'interactions.

- La modalité d'interaction en sortie M'1 de la table interactive est décrite par l'écran tactile et par le langage de description des UI 2D (L'1), M'1=<Ecran Tactile, L'1>. Le langage L'1 correspond à la bibliothèque graphique de la table interactive qui contient les composants graphiques tels que labels, champ de texte, images, fenêtre, etc.
- La modalité d'interaction en entrée M'2 de la table interactive est décrite par l'écran tactile et par le langage de manipulation tactile d'une UI 2D (L'2), M'2=< Ecran Tactile, L'2>. Le langage L'2 décrit les actions utilisateurs sur un écran tactile. Ces actions sont par exemple : toucher avec un doigt pour activer ou sélectionner un élément, toucher avec deux doigts et déplacer pour agrandir, réduire, tourner des composants graphiques, etc.
- La modalité d'interaction en entrée M'3 de la table interactive est décrite par le clavier virtuel et par le langage de commande (L'3), M'3=< Ecran Tactile, L'3>. Le langage L'3 décrit les actions utilisateurs réalisables avec un clavier virtuel tel que saisir un texte.
- La modalité d'interaction en entrée M'4 de la table interactive est décrite par l'écran tactile et par le langage de manipulation des objets tangibles (L'4), M'4=< Objets Tan-

gibles, L'4>. Le langage L'4 décrit les actions utilisateurs sur un écran tactile à l'aide des objets tangibles. Ces actions sont par exemple : poser un objet pour afficher un menu ou un formulaire, déplacer un objet physique pour déplacer l'objet virtuel associé, tourner un objet physique pour sélectionner une fonctionnalité, etc.

Les langages d'interaction L1 et L'1 permettent de décrire les interactions en sortie des UI des applications source et cible. Ces langages peuvent être modélisés en se basant sur des boîtes à outils indépendantes des dispositifs d'interactions en sortie. Crease dans [Cre01] propose une boîte à outils décrivant des widgets multimodales et indépendantes des dispositifs d'interactions en sortie. Les widgets de Crease [Cre01] restent cependant liées aux dispositifs d'interactions en entrée tels que le clavier et la souris.

Les langages d'interaction L2, L3, L'2, L'3 et L'4 quant à eux permettent de décrire et d'interpréter les interactions en entrée des utilisateurs des plateformes de départ et d'arrivées. Ces langages d'interaction permettent d'associer à chaque action de l'utilisateur un comportement ayant un sens dans l'UI de l'application. Les actions utilisateurs telles que cliquer, sélectionner, Ctrl+C, poser un objet, etc. dépendent des dispositifs d'interactions tandis que les comportements de l'UI dépendent du type d'UI et de l'interprétation souhaitée par le concepteur.

**Équivalences des modalités d'interactions** La migration de la plateforme desktop vers une table interactive peut être considérée comme un processus de changement de modalités d'interactions. En effet dans le but de réutiliser l'UI d'une application de départ avec les dispositifs d'interactions de la plateforme d'arrivée, il est possible d'établir des équivalences entre les dispositifs d'interaction et les langages d'interactions des plateformes de départ et d'arrivée.

Pour décrire les interactions d'une UI de départ à l'aide des dispositifs d'interaction d'une table interactive, l'une des approches à envisager peut être la mise en correspondance des dispositifs d'interactions en établissant des équivalences entre les différentes modalités d'interactions des plateformes source et cible. Ce qui consiste par exemple à décrire des équivalences d'abord entre les modalités d'interactions en sortie M1 et M'1 et ensuite entre les modalités d'interactions en entrée M2, M3 et M'2, M'3 M'4.

L'équivalence entre M1 et M'1 consiste à comparer les deux dispositifs qui sont des écrans qui peuvent afficher des UI graphiques et les langages L1 et L'1 qui ont des composants graphiques appartenant à des bibliothèques graphiques différentes. L'équivalence entre les modalités d'interactions en entrée n'est possible que si l'on peut comparer les langages L2, L3, L'2, L'3 et L'4. Ces langages sont décrits en fonction des dispositifs d'interaction et aussi en fonction des applications, en effet chaque concepteur peut décrire un langage propre à son application, l'objectif de ces langages étant de faciliter l'interaction entre un dispositif

physique (clavier, souris, écran tactile, etc.) et l'UI de l'application.

### 1.2.2 Propriétés caractéristiques du modèle d'interactions des tables interactives

Les éléments du modèle d'interactions instrumentales des tables interactives tels que les dispositifs matériels d'interactions en entrée et en sortie, les bibliothèques graphiques ou les modalités d'interactions nous permettent d'identifier des propriétés caractéristiques des tables interactives qui impactent la conception des UI pour des tables interactives.

#### Dispositifs d'interactions en entrée

Ces dispositifs influencent la conception des interactions. Dans le cadre des tables interactives nous identifions deux propriétés qui correspondent aux moyens d'interactions tangibles et tactiles.

**Propriété 1** *Tangibilité des interactions*

Les dispositifs d'interactions en entrée permettent d'associer des objets physiques aux fonctionnalités<sup>2</sup> ou aux composants graphiques<sup>3</sup> d'une UI.

**Propriété 2** *Tactibilité des interactions*

Les dispositifs d'interactions en entrée des tables interactives permettent de décrire des manipulations directes des UI telles que la sélection, l'édition, le redimensionnement, le déplacement, etc.

#### Dispositifs d'interactions en sortie

Ce sont les surfaces d'affichage des tables interactives, les propriétés caractéristiques liées à la taille et à la disposition des tables interactives qui impactent la conception des UI.

**Propriété 3** *Taille de la surface d'affichage*

Cette propriété permet d'adapter la taille des composants graphiques par rapport à la taille de l'écran pour faciliter l'utilisation de l'UI.

**Propriété 4** *Disposition de la surface d'affichage*

La surface d'affichage peut être disposée de manière horizontale comme une table de travail ou de manière verticale comme un tableau collaboratif. Ces dispositions influencent l'orientation et l'utilisation des composants graphiques d'une UI.

**Utilisateurs des tables interactives**

Les utilisateurs influencent la conception de l'UI par leur nombre et leur répartition autour de la surface d'affichage.

**Propriété 5** *Nombre d'utilisateurs*

Cette propriété permet de savoir comment disposer les éléments de l'UI pour faciliter l'accessibilité en fonction du nombre d'utilisateurs.

**Propriété 6** *Répartition des utilisateurs*

Cette propriété permet de savoir comment décrire la collaboration entre les utilisateurs autour de la table. La répartition de l'espace de travail de chaque utilisateur peut se faire par une division géographique de la surface ou permettre aux utilisateurs d'accéder à toute la surface.

**1.2.3 Principes de conception d'UI pour les tables interactives**

Le paragraphe 1.2.1 nous montre les différences de modalités d'interactions entre un desktop et une table interactive. Ces différences impactent aussi la conception d'UI pour ces deux plateformes. Besacier et *al.* montrent que la réutilisation des applications desktop sur les tables interactives en adaptant les éléments de l'UI aux métaphores du papier par

exemple facilite l'utilisation des UI [BRNB07]. Par ailleurs, une réutilisation d'une application desktop sur des tables interactives sans prise en compte de ses spécificités pose deux problématiques majeures : la transformation de l'UI de départ en UI collaborative d'une part et la transformation d'une GUI en TUI d'autre part. Ces deux caractéristiques font parties de l'ensemble des principes qui guident la conception des UI pour les tables interactives.

Les principes de conception d'UI pour une plateforme constituent un ensemble de recommandations pour les concepteurs d'UI qui indiquent comment décrire les aspects tels que les interactions instrumentales, le layout (ou le placement des éléments graphiques), les activités d'une UI et aussi les styles de présentations (couleurs, polices, tailles, etc).

Les principes de conceptions sont des recommandations de haut niveau qui doivent être traduites en règles formelles utilisables pendant la migration [Van97].

Dans cette section nous caractérisons les principes de conception pour la migration des UI vers les tables interactives en trois catégories : les principes de conception pour les UI tangibles, les principes de conception pour les UI collaboratives et colocalisées et enfin les autres principes de conception d'UI sur une grande surface d'affichage telles que la taille et la position de la surface d'affichage, l'utilisation en 360 degré de l'UI et le style de l'UI.

### **Corpus des principes de conception d'UI pour Microsoft Surface**

Les principes de conception d'UI pour une Microsoft Surface sont décrits sous forme de guidelines (ou recommandations) dans le document *User Experience Design Guideline* [Mic11]. Ces guidelines sont le fruit des expériences des utilisateurs ayant développé des UI pour cette plateforme. L'objectif de ces guidelines est de faciliter la conception des interfaces utilisateurs naturelles [Ste12] et intuitives. Ces guidelines couvrent plusieurs aspects du processus de conception des UI tels que la conception des interactions entre les UI et les utilisateurs finaux, les guides de styles pour une cohérence visuelle, les guides d'utilisation des textes, etc.

### **Guidelines pour TUI**

Cette catégorie regroupe les recommandations qui permettent de décrire le comportement des éléments de l'UI qui sont des objets virtuels d'une part et l'association entre ces objets

virtuels et les objets tangibles d'autre part. L'association peut se faire par des tags qui permettent de marquer les objets physiques ou en se basant sur la forme des objets physiques. Les guidelines de cette catégorie sont inspirées par la propriété 1 de tangibilité des interactions.

**Guideline 1** *Comportement des objets virtuels*

Les comportements des éléments d'une TUI pendant leurs utilisations doivent correspondre aux objets physiques. Les éléments (ou objets virtuels) de l'UI à migrer doivent être associés à des objets physiques dans le but d'afficher des menus, des formulaires ou des fenêtres et aussi dans le but d'activer ou d'utiliser des fonctionnalités.

**Guideline 2** *Objets physiques tagués*

Un tag est associé à un objet virtuel d'une TUI (les objets virtuels sont identifiés grâce à la guideline 1). Le déplacement, l'orientation et la position du tag sur l'écran peuvent être associés à des interactions en entrée ou à des comportements de l'objet virtuel associé.

**Guideline 3** *Forme des objets physiques*

La forme d'un objet physique peut être associé à un objet virtuel ou à une fonctionnalité. Le déplacement, l'orientation et la position d'un objet physique sur l'écran peuvent être associés à des interactions en entrée ou à des comportements de l'objet virtuel associé. L'utilisation de la forme des objets physiques comme moyen d'interaction en entrée nécessite un module de reconnaissance de la forme d'un objet tangible.

En considérant l'UI de l'application CBA à la figure 1.2, le menu principal et le formulaire *Ressources* par exemple peuvent être associés à un objet physique pour les afficher facilement sur l'écran. Les règles formelles issues de la guideline 1 permettront d'identifier et de transformer les éléments concrets de l'UI

Les tags permettent par exemple d'utiliser deux objets de la forme avec des couleurs différentes et marqués des deux différents tags pour afficher un menu ou un formulaire.

## Guidelines pour UI collaborative

Cette catégorie regroupe les recommandations pour la conception d'une UI collaborative et colocalisée pour une table interactive. Les guidelines sont liées à la propriété 5 caractérisant le nombre d'utilisateurs et à la propriété 6 qui caractérise leur répartition autour de la surface d'affichage. Elles sont aussi liées à la propriété 3 et à la propriété 4 qui caractérise la taille et la disposition (horizontale ou verticale) de la surface d'affichage des tables interactives.

### **Guideline 4** *Nombre d'utilisateurs de l'UI migrée*

Cette guideline préconise de prendre en compte le nombre d'utilisateurs de l'UI après la migration. Les UI d'une table interactive peuvent être collaboratives (plusieurs utilisateurs) ou non (dans le cas d'un seul utilisateur). Elle impacte le couplage entre les tâches et les utilisateurs (qui est spécifié par la guideline 5), l'organisation de la surface de travail (qui est spécifiée par la guideline 6) et l'accessibilité des éléments d'une UI collaborative (qui est spécifiée par la guideline 7).

### **Guideline 5** *Couplage tâches et utilisateurs d'une UI*

Elle est une spécification de la guideline 4, car elle préconise dans le cas d'une UI multi utilisateurs :

- d'éliminer toutes les boîtes de dialogues bloquantes pour les autres utilisateurs de l'UI.
- de dupliquer certains éléments de l'UI pour permettre à tous les utilisateurs d'accéder aux éléments de l'UI.

### **Guideline 6** *Partage de l'espace de travail*

Cette guideline spécifie la guideline 4 en préconisant des composants graphiques qui facilitent le partage de l'écran dans le cas d'une UI multi utilisateurs. L'espace de travail doit :

- permettre à une personne d'utiliser une UI sans avoir l'aide d'autres utilisateurs,
- permettre à un utilisateur d'utiliser l'UI sans interrompre les utilisateurs présents.

Le partage de l'espace de travail est implémenté de diverses manières en fonction des tables interactives. La table DiamondTouch [DL01] par exemple permet aux concepteurs d'associer un utilisateur à une zone de l'écran. Cependant la table Microsoft Surface quant à elle ne permet pas une division de l'écran en zones associées aux utilisateurs ou à des fonctionnalités. La figure 1.9 illustre une division de l'écran en quatre zones, ce type de partage d'écran n'est pas autorisé par les guidelines de la table Surface. Dans le cadre de la migration d'UI vers les tables interactives, nous pensons que la division de l'écran en plusieurs zones ne permet pas de décrire des UI collaboratives si elle ne permet pas d'échanges entre les utilisateurs et leur mobilité autour de l'écran.

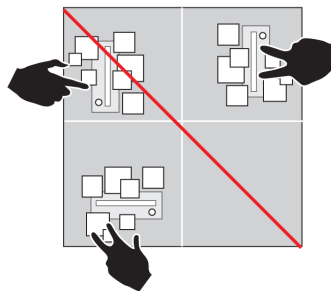


FIGURE 1.9 – Illustration de partage d'espace entre plusieurs utilisateurs

#### **Guideline 7** *Utilisation 360 degré de l'UI*

Cette guideline permet de spécifier la guideline 4 car elle préconise la sélection des composants graphiques pouvant être utilisés partout autour de la table. Elle rend accessible les éléments d'une UI migrée et renforce la collaboration entre les utilisateurs.

La figure 1.10 illustre à un cas d'utilisation des composants graphiques utilisables à 360°.

En considérant que l'UI de l'application CBA (cf. figure 1.2) est migrée vers une table Surface pour quatre dessinateurs de BD par exemple. Les ressources (images, ballons, etc.) utilisés pour la conception des BD doivent être accessibles par les quatre utilisateurs. La guideline 6 préconisant le partage de l'espace de travail et la guideline 6 préconisant l'utilisation des objets 360 degré permettent de décrire une UI sans layout avec des groupes d'éléments utilisables à 360°. La guideline 5 par exemple permettra de supprimer les composants graphiques bloquants tels que les boîtes de dialogues de l'UI de l'application CBA.



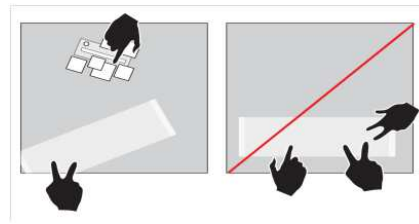


FIGURE 1.10 – Illustration de la propriété 360°

### Autres guidelines pour les UI sur surface

Les deux catégories de guidelines présentées ci-dessus qui permettent de décrire des UI tangibles et des UI collaboratives (cf. section 1.2.3) ne constituent pas le corpus des guidelines applicables pendant la migration d'une UI vers une table interactive. En effet les aspects de l'UI liés au style des composants graphiques, des textes de l'UI et les aspects liés aux interactions tactiles ne sont pas pris en compte par ces deux catégories de guidelines. Cette troisième catégorie regroupe les guidelines de mise en œuvre des interactions tactiles et des styles de l'UI.

**Guidelines pour les interactions tactiles** - Les tables interactives permettent en général de décrire des interactions tactiles. L'ensemble des actions tactiles de l'utilisateur sont interprétées par le dispositif d'interaction en entrée. Dans le cadre des UI tactiles, il existe des actions tactiles standards pour des interactions de redimensionnement, de déplacement ou de rotation. Par exemple les guidelines des interactions tactiles pour une table Surface [Mic11] identifient l'ensemble des gestes recommandés aux concepteurs pour la sélection, l'activation, le déplacement, la rotation, le zoom, etc. Cette catégorie de guidelines est liée à la propriété 2 de tactilité des interactions d'une table interactive.

Dans le cadre de la migration de l'UI de l'application CBA vers les tables interactives Surface, il est indispensable de pouvoir décrire des correspondances entre les interactions tactiles et les interactions du clavier et de la souris de l'UI départ.

**Guidelines pour le style** - Cette sous catégorie regroupe l'ensemble des recommandations pour la personnalisation des aspects visuels d'une UI pour une table interactive. Ces guidelines peuvent être utilisées dans le cadre d'une migration faisant intervenir les concepteurs comme des exemples pour les inspirer du choix de la forme, des couleurs, des icons et aussi de la disposition des textes d'une UI. Dans le cas du scénario de migration, le formulaire *Ressources* par exemple peut être migré suivant comme l'indique le tableau 1.1.


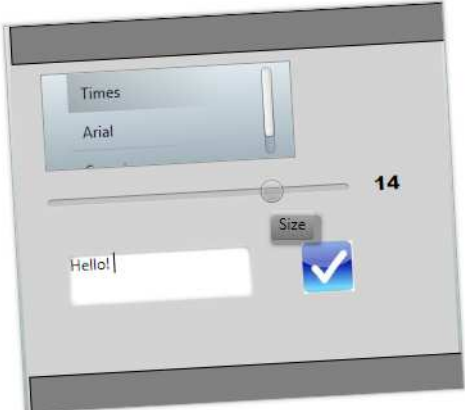
Formulaire de l'application de départ	Formulaire migré en prenant en compte les guidelines de styles
 <p>The original form is titled 'Resources'. It contains a 'Font' section with a dropdown menu set to 'Times' and a 'Size' section with a numeric input set to '12'. Below these is a text area containing 'Some text' and 'A second line of text'. An 'Apply' button is at the bottom right.</p>	 <p>The migrated form has a modern, clean design. It features a 'Times' font selection area and a 'Size' slider set to '14'. Below the slider is a 'Hello!' label and a text input field. A blue checkmark icon is visible next to the 'Size' slider.</p>

TABLE 1.1 – Migration de l'aspect visuel d'un formulaire

### Synthèse des guidelines pour la migration d'UI vers les tables interactives

Le corpus de guidelines décrit dans la section 1.2.3 sont des recommandations de haut niveau utiles et nécessaires pour la migration des UI vers une table interactive. Cette caractérisation de ce corpus a pour but de rendre une UI de départ tangible et collaborative pendant la migration tout en considérant les autres modes d'interactions des tables interactives et les aspects visuels de l'UI.

Les liens entre ces guidelines peuvent être synthétisés à l'aide du diagramme de la figure 1.11 qui présente les trois catégories de guidelines, les 7 guidelines pour les UI tangibles et collaboratives et les liens entre ces guidelines. Dans la catégorie des guidelines pour TUI, les guidelines G2 (Objets physiques tagués) et G3 (Forme des objets Physiques) spécifient les moyens d'interactions tangibles à choisir et à associer aux objets virtuels identifiés par la guideline G1.

Dans la catégorie des guidelines pour UI collaborative, la guideline nombre d'utilisateurs (G4) est spécifiée pour les aspects liés aux tâches par la guidelines de couplage des tâches et utilisateurs (G5), la guideline spécifiant le partage de l'espace de travail (G6) entre les utilisateurs et la guideline préconisant l'utilisation 360° de l'UI (G7) permet l'accessibilité de l'UI pour tous les utilisateurs autour de la table.

Dans la catégorie comportant les autres guidelines pour les UI sur surface nous regroupons deux sous catégories ; la première concerne les interactions tactiles et la seconde le style de l'UI migrée.

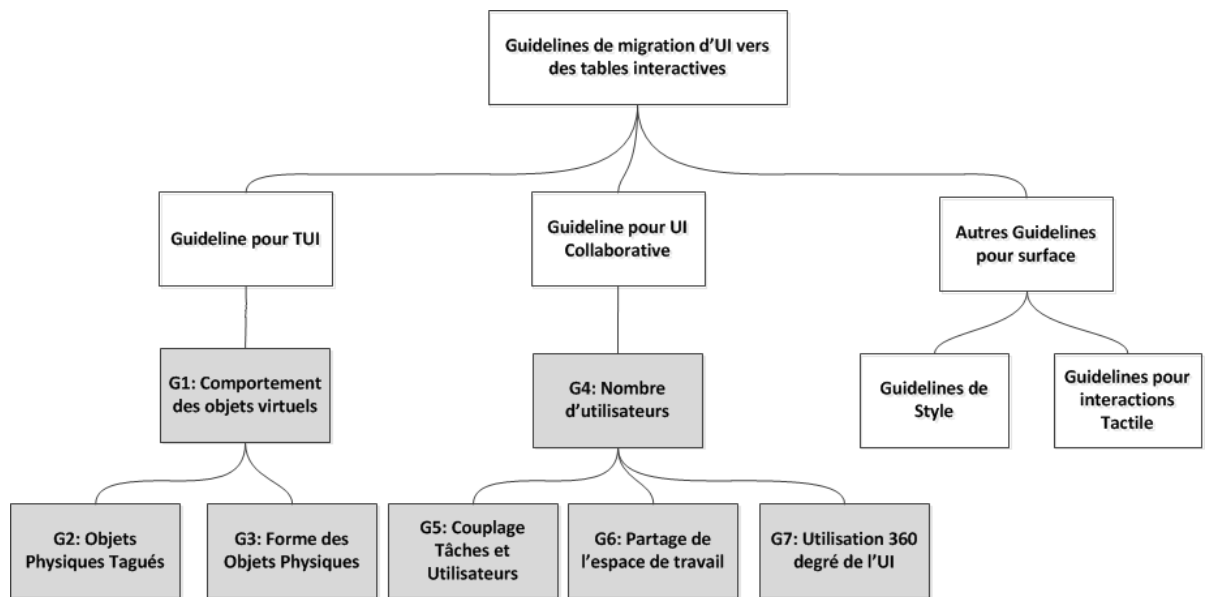


FIGURE 1.11 – Liens entre les guidelines de migration d'une UI vers les tables interactives

### 1.3 Concepts pour la migration d'UI

Dans les sections précédentes, nous avons identifié quels sont les problèmes liés à la migration d'UI vers une table interactives en partant d'un exemple d'application (à la section 1.1). Ensuite nous avons aussi identifié les spécificités des tables interactives qui sont importantes pour la migration des UI à travers ses propriétés caractéristiques et les guidelines. Dans cette section nous abordons les problématiques liées aux architectures des applications à migrer et les modèles d'interactions abstraites pour décrire des équivalences entre les modalités d'interactions.

#### 1.3.1 Architecture des applications à migrer

Les applications à migrer sont des systèmes interactifs (SI) conçus en respectant un modèle d'architecture. Les modèles d'architectures sont des patrons de conception logiciel, ils préconisent des stratégies de répartition des services qui se traduisent par un ensemble de constituants logiciels. En général, la décomposition minimale des SI préconise une séparation entre le Noyau Fonctionnel (NF) et ceux de l'UI. Dans cette section nous présentons deux modèles d'architectures, d'abord le modèle ARCH [UIM92] qui se base sur des composants logiciels spécifiques et indépendants des plateformes, ensuite le modèle MVC [KP88] .

## ARCH

ARCH [UIM92] est un modèle d'architecture qui se base sur des composants conceptuels du modèle de Seeheim [Pfa85]. Comme l'indique la figure 1.12, ce modèle permet une séparation entre le NF, le Contrôleur de Dialogue (CD) et la Présentation. Les deux pieds de l'arche sont des composants spécifiques à une plateforme ; le composant NF décrit un domaine précis et les composants d'Interaction sont liés à des dispositifs du monde réel. Le CD gère l'enchaînement des tâches ainsi que les liens avec les objets des deux composants voisins. L'Adaptateur de domaine joue un rôle d'interface avec le composant NF pour corriger les différences de conceptions. Le composant Présentation est une boîte à outils virtuelle, telle que XVT [Val89] qui implémente les objets de présentations concrétisés finalement par les objets d'interaction de la bibliothèques graphique.

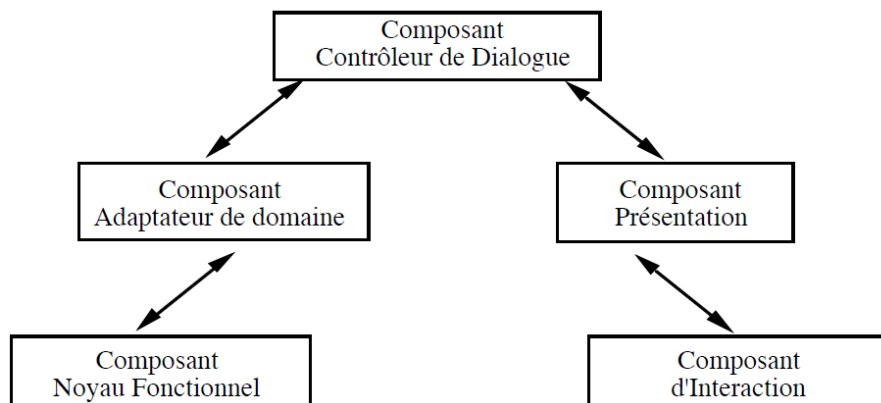


FIGURE 1.12 – Composants du modèle ARCH

**Migration d'une application respectant une architecture ARCH** - Thevenin et al. [TC02] se basent sur l'architecture ARCH pour concevoir des applications avec des UI multi cibles et adaptables. Dans le cadre de la migration, cette approche consiste à remplacer les composants des interacteurs logiques et physiques de l'UI à migrer pour qu'ils soient conformes aux principes de conception de la table interactive ciblée.

La migration des interacteurs physiques concerne la migration du composant d'interaction du modèle ARCH [TC02]. Elle consiste en un changement de bibliothèque graphique de l'application de départ et à utiliser celle de la plateforme d'arrivée. Ce type de migration permet de conserver la nature des composants graphiques mais leurs rendus sont distincts en fonction des plateformes et des bibliothèques graphiques. En considérant l'application CBA par exemple, la nature des éléments logiques de l'UI décrits dans le composant Présentation

tels que Bouton, Liste d'images, etc. sont interprétés en fonction des bibliothèques graphiques (DiamondSpin, Surface SDK, etc.).

La migration des interacteurs physiques permet certes d'utiliser la bibliothèque graphique de la plateforme d'arrivée mais elle ne prend pas en compte les guidelines de la plateforme ciblée.

En effet en migrant les interacteurs physiques de l'UI CBA desktop par les éléments de la boîte à outils d'une table interactive sans prendre en compte les guidelines pour les TUI ou les UI collaboratives, le layout et les interactions de UI résultantes par exemple ne seront pas conformes aux spécificités de la table interactive ciblée.

**Prise en compte des principes de conceptions** - Dans le cadre de la migration d'une application respectant le modèle d'architecture ARCH [Pfa85], les guidelines identifiées à la section 1.2.3 sont utiles pour transformer le Composant de présentation et pour la génération du composant d'interaction en permettant par exemple la sélection des composants graphiques. En considérant l'UI CBA et les guidelines pour UI collaboratives, la guideline pour le partage d'espace de travail (G6) permet par exemple de réorganiser le layout de l'UI CBA en transformant le composant de présentation et la guideline d'utilisation 360 degré permet de choisir des panels avec des mouvements de rotation, de déplacement pendant la génération du composant d'interaction.

## MVC

MVC [KP88] est un modèle d'architecture pour la conception des systèmes interactifs réutilisables. Il divise les applications en trois types de composants : le modèle(M), la vue(V) et le contrôleur(C). Le modèle est une représentation du domaine d'une application, il peut contenir des données, des services, etc. et il fait partie du NF d'une application. La vue est la structure de l'UI d'une application, elle est constituée des éléments d'une bibliothèque graphique. Le contrôleur est une interface entre le modèle, la vue et les dispositifs d'interactions en entrée.

**Migration d'une application respectant une architecture MVC** - En considérant une partie de l'UI de l'application CBA, la figure 1.13 représente des instances des différents éléments du modèle MVC. La vue est composée de *JPanel*, *JLabel*, *JComboBox* et d'une *JList* contenant des images. La sélection d'une catégorie du sélecteur *JComboBox* avec un clavier ou une souris modifie la liste d'images en faisant d'abord appel au contrôleur de *JComboBox* ensuite au modèle de *JList* pour mettre à jour la vue. La migration de cet artéfact d'UI vers une

table interactive qui supporte la bibliothèque graphique DiamondSpin par exemple implique la modification de la vue en remplaçant les composants graphiques par leurs équivalents et en adaptant le code du contrôleur et du modèle pour remplacer les variables correspondant aux composants graphiques de la vue. Par exemple dans le cas où *DSComboBox* correspond à *JComboBox* dans DiamondSpin, le type de la variable *cb* du contrôleur doit être remplacé dans le contrôleur et le modèle.

Par conséquent, la migration de l'UI vers la table interactive avec cette hiérarchie implique une modification à la fois du contrôleur, de la vue et du modèle. Pour éviter ces modifications, il faut que le code des applications à migrer respecte des heuristiques qui favorisent la réutilisation des différents composants de l'application. Parmi ces heuristiques on peut citer : la séparation entre les éléments spécifiques à la plateforme (PSM) et les éléments indépendants de la plateforme (PIM) au niveau du contrôleur et de la vue d'une part, et la non utilisation des éléments PSM du contrôleur de la vue au niveau du modèle d'autre part. Cette séparation permettra de changer les contrôleurs de dialogue spécifiques aux dispositifs d'interaction et les vues spécifiques aux bibliothèques graphiques.

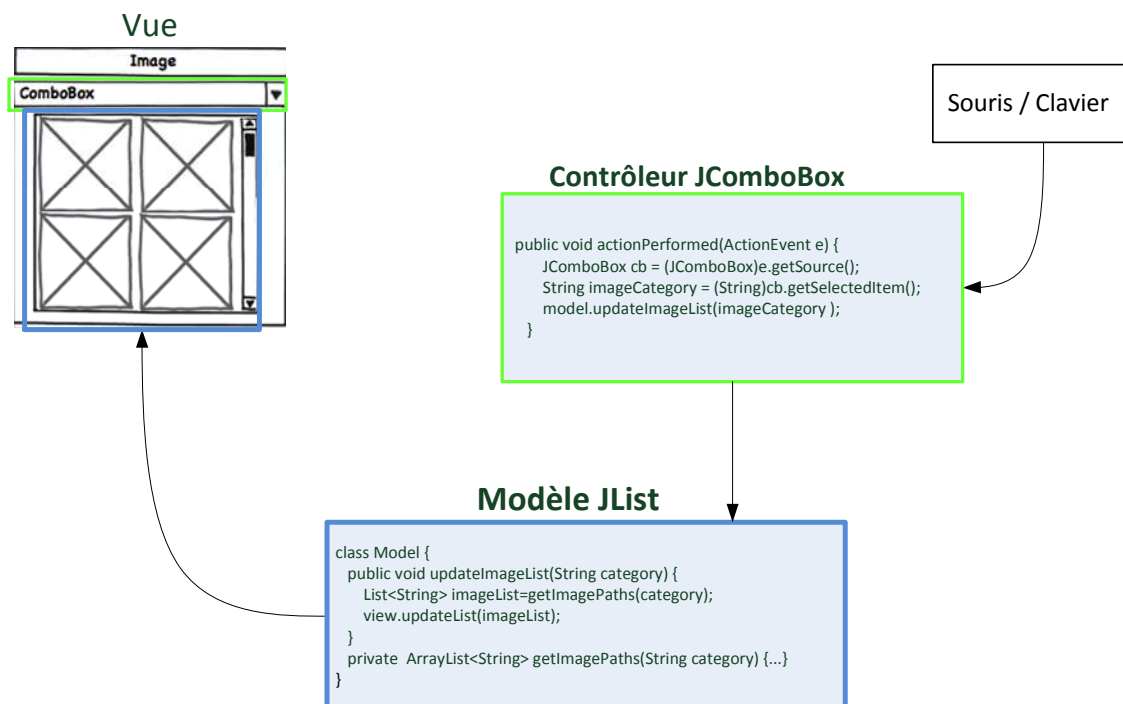


FIGURE 1.13 – Modèle Vue Contrôleur

La prise en compte guidelines pendant la migration des éléments du modèle MVC conçus en respectant les heuristiques préconisant de séparation PIM et PSM se fait comme pour les composants du modèle d'architecture ARCH.

## Résumé

Nous avons présenté dans cette section 1.3.1 le modèle d'architecture de ARCH qui permet la migration des composants de l'UI sans modification des composants du NF. Le modèle d'architecture ARCH facilite aussi la prise en compte des guidelines pendant la migration. Par ailleurs, En considérant un modèle d'architecture MVC avec des composants qui respectent une séparation entre les éléments PSM et les éléments PIM, il est possible de migrer les composants d'UI sans modifier les composants du NF et aussi de prendre en compte les principes de conception pendant la migration. Le modèle MVC permet en une séparation entre les interactions en entrée décrites au niveau des contrôleurs et les interactions en sortie. Cette séparation facilite le **changement des interactions** en entrée.

### 1.3.2 Modèle d'interactions abstraites

La migration d'une UI desktop vers les tables interactives est un processus qui implique un changement de modalité d'interaction tel que nous l'avons montré à la section 1.2.1. Le **changement des modalités d'interactions** doit préserver les actions de l'UI de départ et l'adapter aux dispositifs d'interactions de l'UI d'arrivée. Le changement de modalités d'interactions est possible en décrivant des équivalences entre les différentes modalités d'interactions d'une UI. Ce qui peut se faire en se basant sur un modèle indépendant des dispositifs d'interactions.

Le modèle d'interactions abstraites permet de décrire des équivalences entre deux modalités d'interactions en jouant un rôle de langage pivot.

Dans cette section nous présentons deux modèles d'interactions abstraites et nous en déduirons quelques caractéristiques d'un modèle d'interactions abstraites pour la migration.

#### Modèle de Vlist

Vlist et al. [VNHF11] proposent les primitives d'interaction (*interaction primitives*) qui constituent les plus petits éléments d'interactions adressables ayant une relation significative avec l'interaction elle-même. Selon eux, il est possible de décrire les capacités d'interactions des dispositifs d'interactions à l'aide des primitives d'interactions et ensuite la relation entre les dispositifs et l'UI sera décrite en faisant un mapping sémantique. Les primitives d'interactions sont associées à chaque dispositif d'interaction, elles sont choisies, identifiées et associées aux dispositifs par le concepteur de l'UI. Par exemple les primitives d'interactions



Up, Down correspondent aux touches étiquetées '+' et '-', ces primitives d'interactions ont des types de données et des valeurs, elles sont mappées sur des composants graphiques tels que les contrôles de volume pour permettre l'utilisation de composants avec ces touches. L'objectif de ce mapping sémantique est de faciliter son adaptation en fonction des contextes.

Dans le cadre de la migration, la mise en correspondance des dispositifs d'interactions des plateformes de départ et d'arrivée semble être possible en utilisant une description sémantique des capacités des dispositifs d'interaction d'une plateforme par les primitives d'interactions. Pour cela les primitives d'interactions doivent pouvoir décrire tous les dispositifs d'interactions en se basant sur un seul vocabulaire. En effet Up et Down sont certes adaptés pour les touches '+' et '-' mais en considérant un écran tactile par exemple il faut décrire les différents types de contacts (simple, double, multiple, toucher et glisser, etc.).

### Modèle de Gellersen

Gellersen [GH95] quant à lui propose un modèle d'interactions abstraites qui a pour but de décrire les interactions en entrée et en sortie indépendamment des modalités d'interactions. Ce modèle est aussi indépendant d'un domaine ou d'une application. Il est présenté à la figure 1.14 et il décrit une hiérarchie des interactions en entrée et en sortie. Les interactions en entrée sont raffinées en deux catégories, les interactions d'entrée de données telles que *Editor*, *Valuator* (éditer du texte) et *Option* (sélectionner un élément d'une liste) qui sont des sous classes de *Entry* d'une part et les interactions de *Command*<sup>4</sup> et de *Signal*<sup>5</sup> d'autre part. Les interactions en sortie sont aussi de deux types : les messages (alertes, confirmation, etc.) et la vue qui permet l'affichage des données et des composants de l'UI.

Dans le cadre du changement de modalité d'interaction entraîné par la migration, ce modèle permet de décrire les interactions des différents composants graphiques de l'UI de manière indépendante des modalités d'interactions. Par exemple une fenêtre d'authentification comportant des champs de texte et des boutons, les champs de texte seront représentés par les *Editor* qui sont des interactions à la fois en entrée et en sortie et les boutons seront représentés par des *Command* dans un modèle abstrait.

Ce modèle identifie à la fois les interactions en entrée et en sortie de haut niveau et indépendamment des modalités d'interactions. Dans le cadre de la migration de l'UI CBA par exemple, les interactions telles que la rotation, le déplacement ou le redimensionnement qui sont liés aux guidelines de la table interactives doivent être interprétés comme des com-

---

4. les *Command* sont des interactions en entrée de contrôle avec des paramètres (exemple copier texte, coller texte, etc.)

5. Les *Signal* sont des interactions en entrée sans paramètres (exemple valider)



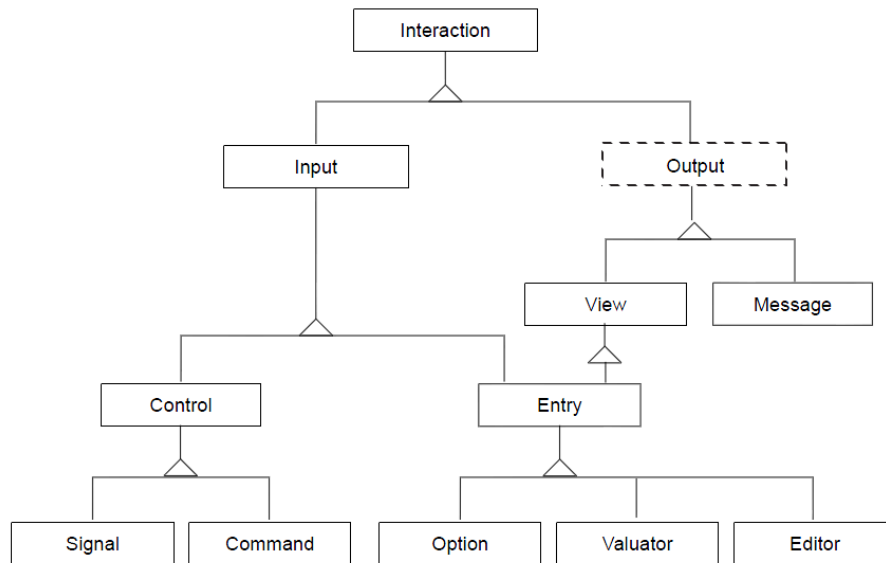


FIGURE 1.14 – Modèle d'interactions abstraites de Gellersen

mandes. Cette interprétation n'est pas générique et dépend du concepteur des interactions, en effet le déplacement peut être considéré comme une rotation en modifiant l'angle par exemple. De manière générale, le modèle de Gellersen est un modèle d'interactions abstraites qui nécessite la spécification de certaines interactions indépendantes des modalités d'interactions pendant la migration.

## Résumé

Deux caractéristiques sont essentielles pour les modèles d'interactions abstraites :

- leurs indépendances d'une modalité d'interaction
- et leur capacité à décrire toutes interactions du langage d'une modalité.

## 1.4 Conclusion

Dans ce chapitre nous avons étudié les problèmes liés à la migration d'UI vers une table interactive qui consiste à :

- transformer le layout de l'UI de départ conformément aux guidelines pour les UI collaboratives
- changer les modalités d'interactions de l'application de départ conformément aux guidelines pour les UI tangibles et tactiles

- adapter la taille et le style de l'UI de départ aux recommandations de la plateforme d'arrivée.

Dans le chapitre suivant, nous étudions quelques mécanismes de migration des UI en faisant ressortir les problématiques liées à la prise en compte des guidelines et au changement de modalités d'interactions de ces mécanismes de migration.

# Chapitre 2

## Approches de migration d'UI

### Sommaire

---

<b>2.1</b>	<b>Introduction</b>	<b>35</b>
<b>2.2</b>	<b>Approches spécifiques de migration d'UI</b>	<b>36</b>
2.2.1	Migration de l'application AgilePlanner vers une table interactive	36
2.2.2	Amener les applications existantes sur les tables interactives	39
<b>2.3</b>	<b>Approches de migration basée sur les modèles de l'UI</b>	<b>41</b>
2.3.1	Migration avec MORPH	42
2.3.2	Approches basées sur des serveurs de migration d'UI	46
<b>2.4</b>	<b>Synthèse et objectifs</b>	<b>50</b>
2.4.1	Synthèse	50
2.4.2	Objectifs	53

---

### 2.1 Introduction

La migration des applications est une activité de déplacement et d'adaptation d'un logiciel d'un environnement source vers un environnement cible. Elle est plus globale que le portage d'application [Moo95] car elle ne se limite pas qu'au changement de langages de programmation ou au changement des systèmes d'exploitation. La migration englobe les problématiques de ré engineering, de reverse engineering, de forward engineering et de portage d'applications. McClure et al. [McC92] définissent le ré engineering comme une amélioration d'un système existant pour le rendre conforme aux standards. Le reverse engineering consiste à analyser un système existant pour décrire la représentation d'origine de manière plus abstraite [McC92]. Le forward engineering est une concrétisation de la représentation abstraite

d'un système dans une implémentation.

Dans notre contexte, les SI à migrer ont une décomposition fonctionnelle minimale qui comprend une UI et un NF. Le mécanisme de migration d'UI d'une application tout en respectant les guidelines de la section 1.2.3 peut être manuel et spécifique dans le cas où le concepteur se charge de transformer l'UI de départ pour la plateforme cible. La migration peut aussi être automatisée par certains processus qui se basent sur des modèles et des principes génériques pour une classe d'applications ou de plateformes.

Dans ce chapitre, la section 2.2 présentent les approches de migration spécifiques à des applications. La section 2.3 présente les approches de migrations d'UI basées sur des modèles d'UI. La section 2.3 fait une synthèse des différentes approches de migrations d'UI et présente les objectifs de notre solution.

## 2.2 Approches spécifiques de migration d'UI

Dans cette section nous étudions les processus de migrations spécifiques à une application ou une boîte à outils. Nous présentons d'abord un processus qui permet de migrer manuellement une application spécifique (AgilePlanner) sur une table interactive, ce processus identifie des guidelines pour les UI collaboratives et s'appuie sur une démarche structurée. Ensuite nous présentons une famille de mécanismes de migration d'UI des applications existantes sur les tables interactives sans re conception de l'UI de départ.

### 2.2.1 Migration de l'application AgilePlanner vers une table interactive

C'est un processus manuel et ad hoc mis en place pour la migration de l'application AgilePlanner<sup>1</sup> vers une table interactive [WGM08]. Le processus est basé sur quatre phases. La première phase consiste à analyser l'UI de l'application à migrer, elle permet d'identifier les différentes zones (menu, légendes, zones d'interaction, espace de travail, etc.). La deuxième phase consiste à évaluer l'UI de l'application à migrer sur une table interactive, le but de cette évaluation est de ressortir les différences entre desktop et table interactive dans le but d'en déduire des recommandations qui seront des guidelines pour le concepteur. Il en ressort les 7 guidelines suivantes :

---

1. AgilePlanner est une application de planification et de gestion de projet suivant la méthode Agile. Sur une table interactive elle est utilisée pour faire du brainstorming pour le planning de projets

- Les composants d'UI de l'application doivent être déplaçable et utilisable en 360°
- Utiliser la reconnaissance gestuelle pour les interactions utilisateurs et éviter les menus traditionnels
- Utiliser l'écriture à main levée au lieu du clavier pour la saisie des textes
- Prendre en compte les interactions concurrentes pendant la conception de l'UI
- Les tailles des composants graphiques doivent être assez grandes pour faciliter les interactions tactiles
- Éviter l'utilisation des boîtes de dialogues pop up
- Permettre à l'UI de l'application de s'adapter aux différentes tailles des tables interactives.

Ces guidelines sont conformes aux guidelines pour les UI collaboratives et aux guidelines pour les interactions tactiles que nous avons présenté au chapitre précédent (cf. section 1.2.3).

La troisième phase consiste à appliquer les guidelines de la phase précédente pour concevoir à nouveau une UI de l'application AgilePlanner utilisable sur table interactive. Certaines des guidelines telles que la rotation et le déplacement des composants graphiques, l'écriture à main levée, les reconnaissances gestuelle et vocale sont fournies par l'environnement logiciel de certaines tables interactives.

La dernière phase du processus consiste à évaluer l'UI produit en demandant aux utilisateurs finaux d'évaluer l'utilisabilité de chaque fonctionnalité migrée de l'application AgilePlanner.

La migration d'une application est un processus qui comporte plusieurs phases. Dans cette section nous avons présenté les approches qui nous permettent d'identifier quatre phases importantes :

1. Analyser l'UI de départ pour identifier la structure des éléments qui la compose et ses fonctionnalités
2. Identifier les guidelines qui permettront la migration, cet identification se fait en se basant sur les différences entre la plateforme de départ et celle d'arrivée
3. Appliquer les guidelines pour migrer l'UI de départ
4. Évaluer le résultat obtenu

## Réutilisation du processus pour d'autres applications

Les étapes de migration de l'application AgilePlanner vers une table interactive et la mise en œuvre des guidelines identifiées constituent “un savoir faire” que nous souhaitons réutiliser pour d'autres applications. Par exemple le choix des composants graphiques utilisables en 360° dépend du concepteur.

Considérons les deux applications : l'UI de l'application CBA et l'UI d'une application agenda présenté à la figure 2.1. Les composants graphiques représentant la barre de menu, la barre de recherche, la liste de catégorie et le tableau de l'application agenda. Ils peuvent être tous déplaçables sur la table ou l'on peut considérer la liste des catégories et le tableau de contacts comme étant ensemble.

Dans le cas de l'application CBA, le choix des groupes utilisables à 360° peut se faire comme le propose la figure 1.3 par exemple.

L'utilisation des guidelines dans cette approche dépend de l'interprétation faite par les personnes en charge de la migration car ces guidelines ne précisent pas formellement comment on peut regrouper les composants graphiques déplaçables par exemple.

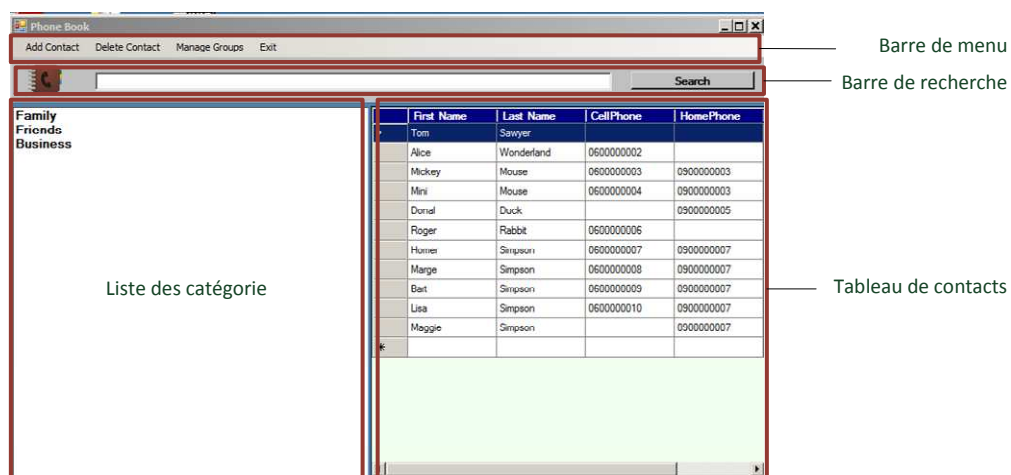


FIGURE 2.1 – Application de consultation des contacts

## Résumé

Même si ce processus n'est pas automatisé et qu'il est spécifique à une application, il permet d'avoir une idée sur les rôles des concepteurs en charge de la migration en présentant

une démarche de migration composée de plusieurs étapes. Cette démarche s'appuie sur les guidelines pour les UI collaboratives et les UI tactiles mais ne prend pas en compte les guidelines pour les UI tangibles par exemple.

Par ailleurs si l'on souhaite réutiliser ce mécanisme de migration, il est indispensable de formaliser les guidelines en s'appuyant sur des éléments qui constituent une UI (tels que menu, formulaire, etc.) et en décrivant des règles de transformations des aspects de l'UI concernés par la migration (layout, interactions, styles, etc.)

### 2.2.2 Amener les applications existantes sur les tables interactives

Cette approche a pour but de réutiliser des applications pour desktop sur des Diamond-Touch. L'approche regroupe des technologies qui prennent en compte des guidelines liées à la conception des UI collaborative. Besacier [Bes10] a caractérisé six technologies qui permettent la réutilisation des applications existantes sur une table interactive qui sont : la capture d'écran, les cartes graphiques virtuelles, les claviers et souris virtuelles, le langage de script, les API d'accessibilité numérique et la réécriture d'une boîte à outils d'interface homme machine. Ces technologies sont comparées à l'aide de cinq axes.

- Le premier axe est la structuration de données échangées entre l'application et la table interactive, ces données sont structurées lorsqu'elles décrivent les composants d'interfaces (fenêtres, boutons, etc.) et elle ne sont pas structurées si les composants d'interfaces sont représentés à l'aide d'images de type bitmap par exemple. La structuration de données concerne le système représentatif des **modalités d'interactions** (cf. section 1.2.1). Le niveau de structuration de données est un axe important car la migration est aussi un changement des modalités d'interactions d'une application sans concevoir à nouveau l'UI de départ.
- Le deuxième axe est la flexibilité des modifications possibles, les différentes manières d'utilisation des applications existantes sur une table impliquent par exemple une réorganisation libre des fenêtres de l'application à migrer ou pour permettre à plusieurs personnes l'interaction en même temps avec les différents éléments de l'UI. La flexibilité des modifications consiste à évaluer la **prise en compte les guidelines pour les UI collaboratives** par une technologie.
- Le troisième axe est la performance de l'ensemble du système. La performance est une caractéristique qui permet d'évaluer le résultat de la migration.
- Le quatrième axe est la compatibilité et la réutilisabilité. La compatibilité avec les applications détermine l'intérêt global d'une technologie. En effet une technologie doit permettre la migration de plusieurs applications. La réutilisabilité détermine l'utilité

d'un prototype. Une solution doit être **générique** et applicable pour plusieurs catégories d'applications.

- Le cinquième axe est la difficulté d'implémentation d'une solution de migration. Elle permet d'évaluer d'une solution par le concepteur avant de la choisir.

Ces cinq critères ont permis d'évaluer des technologies de réutilisation des applications existantes sur les tables interactives dont le résultat est synthétisé sur la figure 2.2. La solution préconisant la réécriture de la boîte à outils de l'interface homme machine(cf. section 2.2.2) est à la fois la plus réutilisable et celle qui permet une structuration de données élevée même si elle est difficile à mettre en place.

	Structuration des données	Flexibilité	Compatibilité et réutilisabilité	Performance	Difficulté d'implémentation
Capture d'écran (sortie graphique)	Bas	Très bas	Très élevé	Bas	Facile
Carte graphique virtuelle (sortie graphique) *	Bas	Bas	Très élevé	Élevé	Moyen
Clavier virtuel et souris virtuelle (entrée utilisateur)	Bas	Très bas à bas	Très élevé	Élevé	Facile
Langage de script (entrée et sortie) *	Très élevé	Élevé	Bas	Très élevé	Moyen
API d'accessibilité numérique (entrée et sortie)	Sortie : moyen Entrée : élevé	Moyen	Moyen	Moyen	Moyen
Réécriture d'une boîte à outils d'interface homme machine (entrée et sortie) *	Élevé	Sortie : moyen Entrée : élevé	Élevé	Très élevé	Difficile

FIGURE 2.2 – Évaluation des technologies de réutilisation des applications existantes sur les tables interactives

### Réécriture d'une boîte à outils d'interface homme machine

Elle consiste à utiliser la boîte à outils de la table interactive à la place de celle du desktop. Le remplacement se fait en interceptant tous les appels de fonction vers les objets de la boîte à outils d'IHM de départ et les rediriger vers les éléments de la table interactive. L'avantage de cette approche est d'utiliser des composants graphiques respectant les guidelines de la table interactive sans reconception de l'application de départ. Cette solution peut être implémentée en utilisant les wrappers [BCR10] par exemple, le wrapper du listing 2.1 permet de réécrire un *JComboBox* en *DSJComboBox*.

Cette solution couteuse en temps pour sa mise en place est spécifique pour les boîtes à



outils mais réutilisable pour les applications d'une boîte à outils.

Listing 2.1 – Exemple de wrapper

```
1 public class JComboBox {  
    public JComboBox() {...}  
3     public void addItem(Object anObject) { ...}  
    }  
5  
    //Wrapper de JComboBox  
7 public class JComboBox {  
    DSJComboBox adaptee;  
9     public JComboBox() { adaptee = new DSJComboBox(); }  
  
11    public void addItem(Object anObject) {  
        adaptee.addItem(anObject);  
13    }  
}
```

## Résumé

Il existe plusieurs solutions qui permettent d'amener des applications existantes sur les tables interactives sans re conception de l'UI. En effet la réécriture des boîtes à outils mise en œuvre dans [Bes10] considère les guidelines pour la conception des UI collaboratives (cf. section 1.2.1) et en prenant en compte la métaphore du papier [BRNB07].

La mise en œuvre de cette solution induit la problématique de la description des **équivalences entre les éléments des bibliothèques graphiques** sources et cibles tout en préservant la nature des interactions en entrée et en sortie.

## 2.3 Approches de migration basée sur les modèles de l'UI

Cette section présente des approches de migration basées sur des modèles pour être indépendante des applications et des plateformes.

### 2.3.1 Migration avec MORPH

MORPH (*Model Oriented Reengineering Process for HCI*) [MR97] est un processus de migration d'une UI textuelle vers une UI graphique en se basant sur des modèles abstraits d'UI et un support pour les transformations vers de nouvelles implémentations graphiques. Le processus de migration avec MORPH implique une reconception de l'UI textuelle en UI graphique car les UI graphiques de type WIMP [vD97] supportent les dispositifs d'interactions de manipulation directes comme une souris. Cette nouvelle conception est aussi un **changement de modalités d'interactions** et l'utilisation d'une boîte à outils graphique. Nous étudions cette approche de migration car comme la migration d'UI vers des tables interactives, les modalités d'interactions des plateformes d'arrivée sont différentes des plateformes de départ avec des nouveaux dispositifs d'interactions.

#### Processus de migration

Il est représenté par l'ensemble des mécanismes qui permettent l'extraction du modèle de l'UI, sa transformation et enfin sa génération pour la plateforme cible. Les modèles abstraits sont utilisés pour la représentation de l'UI.

MORPH décrit un processus (cf. figure 2.3) de migration en trois étapes : la détection, la représentation et la transformation.

- La **détection** est une activité de reverse engineering qui consiste à analyser le code source de l'application à migrer dans le but d'identifier les modèles abstraits. La **génération** est l'opération inverse de la détection qui consiste à produire le code source de l'application migrée à partir de modèles abstraits transformés.
- La **représentation** de l'UI de départ est un ensemble de modèles abstraits issu de la phase de détection.
- La **transformation** consiste à manipuler, augmenter, restructurer les modèles abstraits de l'UI source pour être utilisable dans l'environnement cible.

#### Les modèles abstraits d'UI

Ce sont des éléments clés du processus MORPH car ils représentent les différents aspects (layout, activités, etc.) de l'UI à migrer. Les modèles sont décrits suivant deux niveaux d'abstractions :

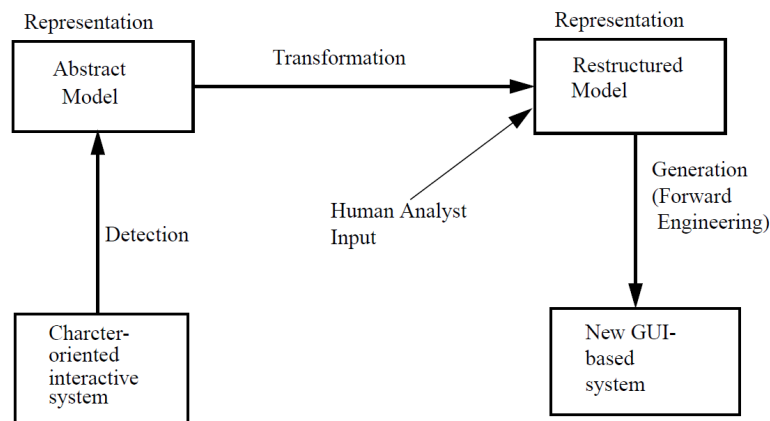


FIGURE 2.3 – Processus de migration avec MORPH

- le niveau des **tâches d'interactions** qui regroupe quatre interactions de base d'une UI [FvDFH90] qui sont : -la sélection dans une liste, -la quantification ou la saisie d'une donnée numérique, -l'indication de la position d'un élément sur l'écran et -la saisie d'une donnée textuelle. Ces tâches d'interactions sont identifiées pour la migration d'UI textuelle vers une UI graphique.
- et le niveau d'**objet d'interactions abstraites** qui représente les éléments abstraits indépendants d'une bibliothèque graphique tel que (Button, List, Menu, etc.).

Les tâches d'interactions et les objets d'interactions abstraites sont identifiés à partir d'UI textuelle en se basant sur l'architecture de l'application à migrer et sur des règles d'identifications [Moo96, RFJ08]. En considérant une application qui respecte une architecture ARCH et en analysant statiquement les codes sources du composant de présentation de cette application, les règles d'identification des objets d'interactions abstraites et des tâches d'interactions permettent de distinguer les menus des listes de sélection.

Dans le cadre de la migration, les tâches d'interactions permettent de décrire les interactions des objets d'interactions abstraits indépendamment du dispositif d'interaction de départ (clavier).

Les tâches d'interactions constituent un **modèle d'interactions abstraites** pour la migration d'une UI textuelle vers une UI graphique.

Le mapping entre les tâches d'interactions et les objets d'interaction abstraits est **représenté** à l'aide du langage de représentation des connaissances CLASSIC [RBB<sup>+</sup>95]. Par exemple une liste de sélections des tâches peut être transformée en menu si le nombre d'éléments est inférieur à 10. Le modèle abstrait décrit à la fois les interactions de l'utilisateur (la sélection, l'édition, etc.) et les composants graphiques (bouton, liste de sélection, menu etc.)

indépendamment des bibliothèques graphiques.

Composants Graphiques	Objets d'Interactions	Rôles
MORPH-RADIO-BUTTON	SELECTION-OBJECT	-action= Visible-State-Change -number-of-states=2 -variability = fixed -grouping= grouped
MORPH-BASIC-MENU	SELECTION-OBJECT	-action= Procedural-Action -number-of-states=(min=2, max=10) -variability = fixed -grouping= not-grouped
AWT-Choice	INTERACTION-OBJECT	-action=Procedural-Action -number-of-states=(min=2, max=15) -variability = fixed -grouping= not-grouped

TABLE 2.1 – Composant graphique et rôles

La **transformation** du modèle d'UI est constituée des règles d'inférence, elle permet d'établir des règles d'équivalences entre les modèles abstraits sources et cibles en se basant sur les rôles. Par exemple un AWT-Choice est équivalent à un MORPH-BASIC-MENU au nombre d'état près suivant le tableau 2.1 et ceci indépendamment des types des tâches d'interactions.

Le concepteur intervient manuellement sur les modèles d'UI après la transformation dans le but de placer les éléments de l'UI. En effet, l'UI textuelle de départ n'est pas structurée comme une UI graphique et les modèles de tâches d'interactions et d'objets abstraits d'interactions ne permettent pas de décrire le layout par exemple.

La transformation par inférence permet d'inclure les principes de conception d'UI pour la plateforme cible dans la base de connaissances [MR97].

### Réutilisation du processus MORPH pour la migration d'UI vers les tables interactives

Dans cette section, nous utilisons les modèles abstraits de l'approche MORPH pour représenter les interactions des éléments graphiques d'un artéfact de l'UI CBA cf. figure 2.4 :

- le menu principal est une SELECTION-OBJET avec les rôles (action=*Procedural-Action*, number-of-states=(2..10), variability=*fixed*, grouping= *not-grouped*)
- la liste déroulante (*ComboBox*) est une SELECTION-OBJET avec les rôles (action=*Procedural-Action*, number-of-states=(2..10), variability=*fixed*, grouping= *not-grouped*)

- la liste d'images est une SELECTION-OBJET avec les rôles ( $action=Procedural-Action$ ,  $number-of-states=(2..10)$ ,  $variability=fixed$ ,  $grouping=not-grouped$ )

En considérant comme règle de transformation que tous les objets d'interactions abstraites de type SELECTION-OBJET ayant les rôles  $number-of-states=(2..10)$  par exemple sont insérés dans des containers déplaçables et utilisables 360° comme sur la figure 2.4. Cette transformation pose un problème de regroupement des éléments graphiques migrés sur la table interactive. En effet sur l'exemple ci-dessous (cf figure 2.4), la liste déroulante et la liste d'images par exemple doivent être regroupés dans un même container car l'utilisation de ces deux composants se fait au même moment par un utilisateur.

La réutilisation de ce processus de migration dans le cadre de la migration vers une table interactive *DiamondTouch* par exemple nécessite un mapping entre les objets d'interactions abstraites et chaque élément de la bibliothèque graphique *DiamondSpin*.

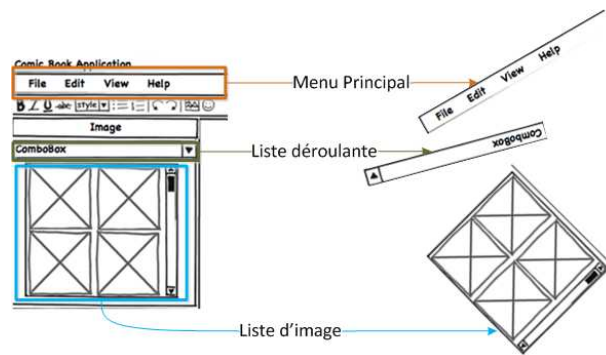


FIGURE 2.4 – Migration vers une table interactive avec MORPH

## Résumé

Cette approche propose un processus de migration à trois phases. La première est une **détection** qui permet la représentation de l'UI textuelle de départ dans un modèle abstrait qui comprend les objets d'interactions abstraits et les tâches d'interactions. La deuxième est une **transformation** du modèle abstrait prenant en compte des principes de conception qui guident le choix des composants graphiques. La troisième phase est la **génération** de l'UI pour la plateforme cible. Ce processus fait intervenir le concepteur pendant la migration pour placer et définir le layout de l'UI à générer.

Dans le cadre de la migration d'une UI desktop vers une table interactive, l'utilisation du processus MORPH requiert d'une part l'annotation des bibliothèques graphiques des plateformes source et cible, d'autre part la spécifications des principes de conceptions en règles de transformations qui prend en compte les éléments du modèle abstrait.

### 2.3.2 Approches basées sur des serveurs de migration d'UI

Dans cette section nous présentons les approches de migrations réutilisables pour différentes plateformes en s'appuyant sur des modèles abstraits d'UI. Ces solutions de migration d'UI génériques sont utilisées par des serveurs pour la migration des UI dans des contextes ubiquitaires comportant plusieurs types de plateformes. Le serveur de migration est chargé d'adapter à l'exécution l'UI d'une application pour une plateforme donnée. Nous étudions dans cette section les modèles d'UI et les mécanismes de ces approches pour la migration d'UI vers une table interactive.

#### Modèles d'UI

Les modèles abstraits permettent de décrire les UI comportant en général plusieurs niveaux d'abstractions dans l'objectif de décrire les différents aspects des UI. Le framework de référence CAMELEON [CCB<sup>+</sup>02] propose quatre niveaux d'abstraction pour décrire les UI : le niveau tâche, le niveau interface abstraite (AUI), le niveau interface concrète (CUI) et le niveau interface finale (FUI). Ils existent plusieurs implémentations pour chacun de ces niveaux d'interface. Paternò *et al.* [PSS09] proposent le langage MARIA XML qui décrit les UI aux niveaux (AUI et CUI) et USIXML (User Interface eXtensible Markup Language) [VLM<sup>+</sup>04] décrit aussi des modèles de CUI, AUI.

Le lien entre ces différents niveaux d'abstraction étant assuré par un modèle de mapping, le processus de développement consiste en un raffinement successif du niveau le plus abstrait pour atteindre une plateforme précise. En effet l'application est spécifiée d'abord sous forme de tâche et de concept qui seront raffinés en AUI, ensuite en CUI et enfin en FUI (cf. figure 2.5).

Les modèles des niveaux d'abstractions de tâches, d'AUI et de CUI sont utilisés par les mécanismes de migrations pour transformer l'UI de départ à la plateforme cible.

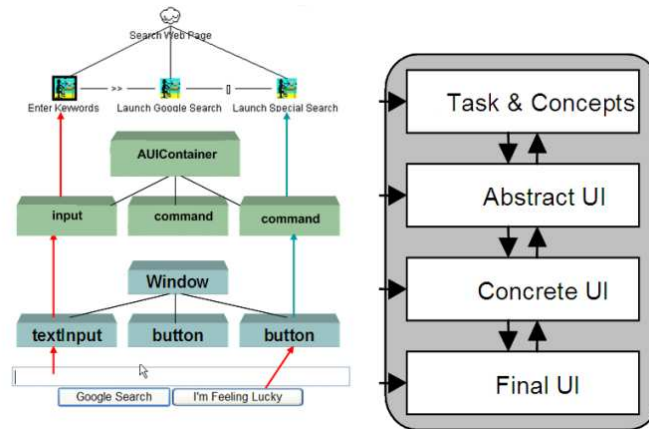


FIGURE 2.5 – Exemple de transformation USIXML

### Mécanismes de migration centrés sur les activités

Ces mécanismes ont pour objectif de faciliter le changement de modalités d'interactions de l'UI de départ. Kong et al. [KSM99a] proposent à cet effet un modèle de tâches<sup>2</sup> basé sur les *actions des utilisateurs*, sur les *écrans* et les transitions entre les différents écrans. Ce modèle identifie trois types d'actions :

- les actions d'entrée de donnée (*tell*),
- les actions d'affichage d'informations à un même endroit de l'écran (*ask-standard*<sup>3</sup>)
- et les actions d'affichage d'informations à différents endroits de l'écran (*task-select*).

Les trois types d'actions permettent d'identifier les différentes informations échangées entre l'application et l'utilisateur. Les activités permettent de représenter les actions possibles de l'utilisateur à partir de l'UI textuelle dans le but de déduire le modèle d'interface abstrait. Par exemple les actions de types *tell* qui manipulent les dates sont représentées par des objets graphiques qui peuvent être des calendriers.

Dans le cadre de la migration d'une UI desktop vers une table interactive, le modèle de tâches peut permettre la prise en compte de la guideline de "couplage tâches et utilisateurs" (G5) et se fait en identifiant et en supprimant les activités bloquantes pour les autres utilisateurs telles que l'affichage d'une boîte de dialogue par exemple.

Par ailleurs, les trois types d'actions identifiées ci-dessus ne permettent pas de représenter toutes les activités possibles sur une UI desktop. En effet, les interactions en entrée par

2. Ce modèle de tâches est utilisé par un processus de migration générique d'une UI textuelle vers une UI graphique.

3. Dans le cadre d'une UI textuelle, certaines zones de l'écran peuvent être utilisées pour présenter des informations de progression par exemple.

exemple peuvent être des saisies de données ou des commandes (déplacement, redimensionnement ou activation) avec ou sans paramètre.

Les modèles d'interactions abstraites permettant de représenter l'ensemble des activités possibles des plateformes sources et cibles et peuvent être utilisés pour représenter le modèle de tâches.

### Mécanismes de migration basés sur les modèles AUI et CUI

Les mécanismes de migration basés sur les modèles AUI et CUI sont utilisés par plusieurs solutions de migrations [MVL06, WCK<sup>+</sup>]. Par exemple le serveur de migration d'UI de la figure 2.6 [PZ10] est une solution qui a pour objectif d'adapter la taille des pages web conçues pour des écrans desktop aux écrans des smartphones. Ce serveur abstrait les pages HTML dans les modèles CUI et AUI du langage MARIA XML et transforme les modèles obtenus en fonction de ces principes :

- adapter les tableaux en les découpant en plusieurs tableaux pour les tableaux comportant plusieurs colonnes ou en réduisant les données qu'ils contiennent (si une cellule d'un tableau contient un nombre maximum de mots alors on crée un lien (détail...) pour afficher les contenus en trop)
- transformer les textes longs en liens (détail...) comme pour les contenus des tableaux
- transformer les images en les redimensionnant
- convertir les listes en menu drop down par exemple
- adapter la taille et la disposition des composants graphiques en les redimensionnant et en adoptant un layout vertical par exemple

Le serveur applique ces principes sur les modèles de CUI et d'AUI. Dans ce cas, la migration ne change pas de bibliothèque graphique car l'UI reste toujours en HTML. Cependant le layout, la taille des composants graphiques, les données et le type de certains composants graphiques sont transformés et remplacés pendant la migration. Les transformations des instances des modèles AUI et CUI de l'UI de départ sont horizontales car elle génèrent d'autres instances de ces modèles pour la plateforme d'arrivée.

### Réutilisation d'un serveur de migration d'UI desktop vers les tables interactives

Dans cette section nous étudions les adaptations nécessaires dans le cas d'une réutilisation du serveur de migration (cf. figure 2.6) pour migrer l'application CBA (cf. section 1.1.1) sur une table interactive (Microsoft Surface). Une adaptation du serveur de migration nécessite :



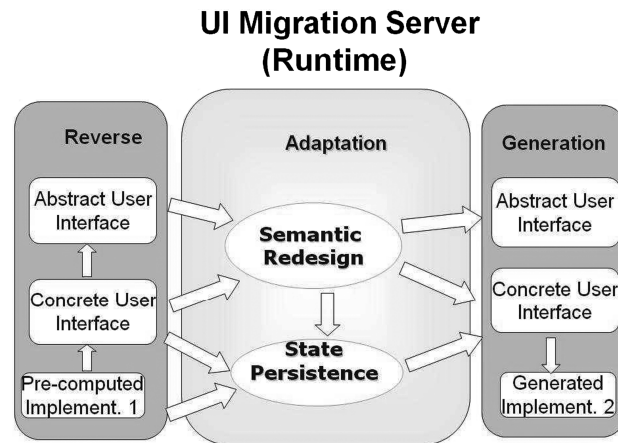


FIGURE 2.6 – Serveur de migration

- une table d'équivalence (tableau 2.2) entre les éléments du langage MARIA XML et l'API Java Swing pour abstraire et concrétiser l'UI de départ.
- une formalisation des guidelines sur les éléments du langage MARIA XML.

La table d'équivalence permet de décrire les correspondances entre les bibliothèques graphiques des plateformes sources et cibles. Le tableau 2.2 par exemple décrit une correspondance entre Java Swing et XAML en se basant sur le langage MARIA XML. Cette correspondance est statique et doit être établie pour l'ensemble des composants graphiques d'une bibliothèque graphique.

Par ailleurs, Silva et al. [SC12] proposent plusieurs critères pour la correspondance entre bibliothèques graphiques basées sur les langages XML. Le premier critère est le comportement des composants graphiques car il caractérise les actions utilisateurs indépendamment de la représentation du composant graphique. Les autres critères utilisables dans le cadre de la migration sont le style des UI, les balises des éléments graphiques.

TABLE 2.2 – Table d'équivalence

MARIA XML	Java Swing	XAML Surface
Activator	JButton	Button
Single choice	JCombox	ListBox
Text Edit	JTextField	TextBox
Object	Image	Image
Grouping	JPanel	ScatterViewItem, Grid

En considérant par ailleurs la guideline d'utilisation 360° (G7), elle peut être traduite sur les éléments du langage MARIA XML, qui par exemple donne la règle suivante : tous *Grouping* sont transformés en *ScatterViewItem* pour être conforme à la guideline d'utilisation 360° de l'UI.

## Résumé

Les serveurs de migration présentés dans cette section se basent sur des modèles d'UI pour leur permettre de décrire des mécanismes de migration génériques. Le modèle décrivant les actions utilisateurs et les transitions des écrans proposés par Kong et al. [KSM99a] peut être considéré aussi comme un modèle de tâche. Dans notre contexte de migration, ce modèle permet de déduire le AUI et permet aussi la prise en compte de guidelines pour la table interactive.

Les modèles décrivant la structure, le regroupement et le positionnement des éléments graphiques sont situés au niveau AUI et CUI. Dans le cadre de la migration vers les tables interactives, ces modèles sont transformés pour modifier le layout de l'application de départ et définir un nouveau regroupement pour les éléments graphiques conformément aux guidelines.

La concrétisation des modèles d'UI en applications finales se fait en se basant sur des tables d'équivalences statiques ou sur des règles d'inférences.

## 2.4 Synthèse et objectifs

### 2.4.1 Synthèse

Les approches de migration d'UI présentées dans ce chapitre nous montrent que les processus de migration peuvent être spécifiques ou réutilisables. Dans le but de proposer une solution de migration qui permet<sup>4</sup> le changement des modalités d'interactions conformément aux guidelines d'UI tactiles et tangibles, la transformation des layout et l'adaptation de la taille et du style de l'application de départ, nous évaluons les approches présentées ci-dessus suivant quatre critères :

- **Équivalences entre les plateformes source et cible** : ce critère permet de savoir comment une approche décrit les équivalences entre la plateforme source et cible. Dans les

---

4. conformément aux problématiques identifiées au chapitre précédent.

approches ci-dessus, les équivalences entre les plateformes se font en se basant sur les tâches, modèles AUI ou CUI et les composants graphiques. Nous avons aussi identifié deux types d'équivalence entre ces éléments des plateformes : - les équivalences statiques qui sont définies par les concepteurs (à l'aide d'une table d'équivalence par exemple) - et les équivalences dynamiques qui sont établies en se basant sur les caractéristiques des éléments à comparer (utiliser les inférences d'un modèle de connaissance par exemple)

- **Prise en compte des guidelines** : ce critère permet de savoir comment une approche permet la prise en compte des guidelines. Cette prise en compte peut être basée sur l'intuition du concepteur ou être décrite formellement dans des règles. Certaines approches ne permettent pas la prise en compte de toutes les guidelines que nous avons identifiées au chapitre précédent.
- **Modèle d'interactions de l'UI** : ce critère nous permet de savoir comment est décrit les interactions utilisées par une approche et si le modèle peut être réutilisé dans le cadre de la migration vers les tables interactives.
- **Modèle de structures de l'UI** : ce critère permet de savoir si les modèles de structures d'une approche permettent de transformer l'UI de départ conformément aux guidelines pour les UI collaboratives.

Le tableau 2.3 présente une évaluation des approches présentées dans ce chapitre avec les quatre critères que nous avons décrits ci-dessus. Le tableau 2.3 nous permet de remarquer que les modèles d'interactions utilisés par les approches MORPH et les serveurs de migration ne décrivent pas de manière exhaustive les interactions de chaque objet d'interactions indépendamment de modalités d'interactions car leurs objectifs sont de décrire les activités d'une UI donnée. Dans le cadre de la migration, une description des interactions de chaque composant graphique permet d'établir des équivalences entre les plateformes source et cible. Par ailleurs, les modèles de structures utilisés par les approches MORPH et les serveurs de migration peuvent être réutilisés dans le cadre de la migration vers les tables interactives.

### Limites des modèles d'UI

En réutilisant les différents modèles d'UI (interactions et structures) présentés dans les approches ci-dessus dans le cadre de la migration vers les tables interactives, nous identifions les limites suivantes.

- **Équivalences statiques** entre les plateformes ne sont pas exhaustives et ne facilitent pas la réutilisation de l'approche pour d'autres applications ou d'autres plateformes.
- **Prise en compte des guidelines** : ces approches ne permettent pas la prise en compte de toutes les guidelines pour les tables interactives car les modèles de structures et d'interactions n'expriment pas tous les aspects des UI pour les tables interactives.

<b>Approches</b>	<b>Équivalences entre plateformes</b>	<b>Prise en compte des guidelines</b>	<b>Modèle d'interactions de l'UI</b>	<b>Modèles de structures de l'UI</b>
Approches ad-hoc	Statique : définit par le concepteur en charge de la migration	Basée sur l'intuition du concepteur	Pas de modèle d'interactions	Pas de modèle de structures
Réécriture de la boîte à outils	Statique : basées sur les boîtes à outils graphiques et définit par le concepteur	Toutes les guidelines ne peuvent pas être prises en compte	Pas de modèle d'interactions	Pas de modèle de structures
MORPH	Dynamique : basée sur un modèle de connaissances et exprimé par des rôles	Règles de transformations basées sur les modèles abstraits	Basée sur les règles d'inférences sur les rôles	Décrit par des objets d'interactions abstraites n'exprimant pas le positionnement des éléments graphiques.
Serveurs de migration	Statique : tables d'équivalences statiques et mapping des modèles	Règles de transformations des modèles	Modèle de tâches et AUI	AUI et CUI

TABLE 2.3 – Synthèse des approches de migration d'UI

### **2.4.2 Objectifs**

La solution de migration d'UI que nous proposons se base sur des applications décrivant le modèle d'architecture MVC. Notre solution a pour objectifs de proposer un modèle d'interactions et de permettre la prise en compte des toutes les guidelines.

Dans la partie suivante, nous présentons le cœur de notre approche pour la migration des UI vers des tables interactives. En nous servant de l'étude précédente, nous proposons une approche basée sur un modèle d'interactions abstraites qui permet d'établir des équivalences dynamiques entre les plateformes et qui permet aussi la prise en compte des guidelines pour les tables interactives.

## **Deuxième partie**

### **Contributions**

# Chapitre 3

## Modélisation des interactions abstraites

### Sommaire

---

<b>3.1</b>	<b>Introduction</b>	<b>55</b>
<b>3.2</b>	<b>Primitives d'interactions</b>	<b>56</b>
3.2.1	Les primitives d'interaction en entrée	57
3.2.2	Les primitives d'interaction en sortie	61
3.2.3	Méta Modèle des primitives d'interaction	62
<b>3.3</b>	<b>Primitives d'interactions et Composants graphiques</b>	<b>63</b>
3.3.1	Un méta modèle de composant graphique pour identifier les primitives d'interaction intrinsèques	64
3.3.2	Règles d'identification des primitives d'interaction intrinsèques	66
3.3.3	Les limites des primitives d'interaction intrinsèque d'un composant graphique	69
<b>3.4</b>	<b>Représentation générique de la structure analysable d'une UI</b>	<b>71</b>
3.4.1	Primitives d'interaction effectives des interacteurs	72
3.4.2	Container	75
<b>3.5</b>	<b>Synthèse</b>	<b>77</b>

---

### 3.1 Introduction

Les processus de migration d'UI réutilisables que nous avons étudiés dans le chapitre précédent, de manière générale fonctionnent en transformant les modèles abstraits de l'UI. Ces modèles permettent de décrire les interactions entre l'UI et les utilisateurs [GH95, KSM99b], la structure et le positionnement des éléments d'une UI [PSS09, VLM<sup>+</sup>04].

Les interactions entre les utilisateurs et des systèmes interactifs peuvent être décrites par des activités de l'UI de manière globale sous forme de tâches. Par exemple une tâche d'authentification comprend les activités de saisie de l'identifiant et du mot de passe, ensuite la validation de la saisie et enfin l'authentification ou non de l'utilisateur par le système. Les différentes activités de l'utilisateur dans une tâche sont reliées à des composants graphiques, l'activité de saisie d'identifiant par exemple est possible que sur des composants graphiques pouvant recueillir des données en entrée.

Dans le cadre de la migration, l'un des objectifs est d'établir des équivalences entre les composants graphiques des plateformes source et cible. Pour atteindre cet objectif, nous nous basons sur un modèle d'interactions abstraites qui permet de décrire l'ensemble des activités possibles sur les différents composants graphiques et de manière indépendante des dispositifs d'interactions des plateformes source et cible.

Le processus de migration transforme aussi les éléments structurels d'une UI tels que la position et le regroupement des éléments d'une UI. Par exemple, la migration de l'application AgilePlanner [WGM08] par exemple les menus sont regroupés dans des panels utilisable en 360°.

Ce chapitre propose à la section 3.2 un modèle d'interactions abstraites qui décrit les activités atomiques possibles sur les composants graphiques. Ce modèle identifie l'ensemble des primitives d'interactions des composants graphiques, la section 3.3 présente les règles d'identifications des primitives d'interactions des composants graphiques. La section 3.4 propose un modèle représentation de la structure d'UI pour permettre les transformations conformément aux guidelines.

## **3.2 Primitives d'interactions : un modèle des interactions abstraites pour composants graphiques**

Les modèles des systèmes interactifs [NC94, PHR02, Weg97] distinguent deux types d'interactions [W3C03] entre les utilisateurs et une UI : les interactions en entrée et les interactions en sortie. Les interactions en entrée permettent de fournir des données ou d'invoquer des fonctionnalités du NF grâce à une UI et à des dispositifs d'entrée (souris, clavier, microphone, camera de reconnaissance gestuelle, écran tactile, accéléromètre, etc.). Les interactions en sortie permettent d'effectuer des rendus des données du NF à travers des dispositifs de sortie (tel un écran, des hauts parleurs, etc.). Les interactions (en entrée ou en sortie) se déclinent en plusieurs modalités d'interactions en fonction des langages et des disposi-



tifs d'entrée ou des dispositifs de sortie utilisés. Chaque modalité d'interaction est utilisée comme un canal de communication entre un utilisateur et une application pour transmettre ou acquérir des informations [Nig94].

Une **primitive d'interaction** est une décomposition atomique d'une interactions<sup>1</sup> sur les composants graphiques d'une UI.

Les interactions en entrée en modifient l'état d'un composant graphique soit à travers ses propriétés (données contenues, taille, position et représentation) ou par son comportement (tels les appels des fonctionnalités du NF et les interactions sur d'autres composants graphiques d'une UI, etc.). Les interactions en sortie quant à elles permettent d'afficher des données provenant du NF ou de changer les propriétés graphiques des composants graphiques de l'UI.

Les primitives d'interaction sont des interactions abstraites indépendantes des dispositifs physiques qui sont dérivés en interactions concrètes. Par exemple l'édition dans un champ de texte nécessite une sélection de ce champ de texte (**Widget Selection** ou **Navigation**) avec une souris ou un clavier puis l'édition (**Edition**) de son contenu.

Cette section présente une caractérisation des primitives d'interactions en fonction des deux types d'interactions (entrée et sortie) ainsi qu'une méta modélisation des primitives d'interaction pour le processus de migration.

### 3.2.1 Les primitives d'interaction en entrée

Elles décrivent de manière abstraite toutes les actions des utilisateurs ou des autres composants graphiques qui permettent de modifier l'état d'un composant graphique. Nous avons identifiés comme interactions en entrée sur les contenus l'édition (**Data Edition**), la sélection (**Data Selection**) et le déplacement (**Data Move In** et **Data Move Out**) de contenus. Elles sont indépendantes des dispositifs d'interaction en entrée et sont effectuées avec des dispositifs de manipulation directe (souris, écran tactile, reconnaissance gestuelle, etc.) ou en combinant plusieurs mode d'interaction (clavier et souris).

#### **Primitive d'interaction 1 : Data Edition**

Elle consiste à modifier les données d'un composants graphiques qui ont des contenus.

La modification du contenu d'un composant graphique peut se faire avec un clavier, une reconnaissance vocale, gestuelle ou de forme en fonction des plateformes. Dans le cadre des tables interactives, l'édition se fait avec un clavier virtuel ou à main levée sur un écran tactile.

**Primitive d'interaction 2 : Data Selection**

Elle permet d'accéder aux contenus d'un composant graphique.

Les composants graphiques peuvent contenir plusieurs données, dans le cas des listes par exemple elle permet la sélection d'un contenu précis. Elle se fait à l'aide d'un clavier, d'une souris, d'un écran tactile, d'un dispositif de reconnaissance gestuelle, etc.

**Primitive d'interaction 3 : Data Move In**

Elle permet de recevoir le contenu d'un autre composant graphique de type compatible.

Les composants graphiques peuvent s'échanger des contenus, le drag et le drop permettent de déplacer un contenu à l'aide d'une souris ou d'un geste tactile. Cette interaction n'est pas que liée à une souris, elle peut être migrée sur une table interactive en utilisant l'écran tactile ou émulée avec un objet tangible.

**Primitive d'interaction 4 : Data Move Out**

Elle permet d'exporter le contenu vers un autre composant graphique.

Cette primitive d'interaction correspond à la première action d'un drag-drop. Elle est indépendante des dispositifs physiques et peut être émulée sur une table interactive.

Les trois primitives d'interactions en entrée sur le(s) contenu(s) des composants graphiques décrit de manière exhaustive l'ensemble des actions possibles qu'un utilisateur peut faire sur un élément graphique contenant des données utilisables par le noyau fonctionnel. En effet, si nous considérons le(s) contenu(s) d'un composant graphique comme une donnée, ces primitives d'interactions permettent la création, la modification, l'accès et la suppression des données (CRUD [D. 06]) d'un composant graphique.

Par ailleurs, les actions utilisateurs sur une UI<sup>2</sup> ne concernent pas que le contenu des composants graphiques, leurs propriétés graphiques sont modifiées aussi par soit un déplacement (**Widget Move**), soit une rotation sans changement de position (**Widget Rotation**), soit un redimensionnement (**Widget Resize**). Un composant graphique peut aussi être sélectionné de manière directe (**Widget Selection**) ou en naviguant à travers un container (**Navigation**). Les primitives d'interactions sur ces propriétés graphiques sont indépendantes des dispositifs d'interaction en entrée. En effet elles peuvent être effectuées avec des dispositifs de manipulation directe (souris, écran tactile, reconnaissance gestuelle, etc.) ou en combinant plusieurs mode d'interaction (clavier et souris).

**Primitive d'interaction 5 : Widget Move**

Elle permet d'exprimer le changement de la position d'un composant graphique.

La position des composants graphiques peuvent être changée par un utilisateur à l'aide d'un dispositifs de manipulations directes ou des raccourcis d'un clavier. Cette primitive d'interaction est définie pour les composants graphiques déplaçable à l'aide.

**Primitive d'interaction 6 : Widget Rotation**

Elle permet d'exprimer le changement d'orientation d'un composant graphique.

L'orientation des composants graphiques est une caractéristique importante dans le cadre des UI collaboratives et co localisée. Nous considérons le changement d'orientation comme une primitive d'interaction car elle peut être définie pour des écran utilisables par plusieurs personnes tels qu'une table interactive, une tablette tactile, un ordinateur portable avec écran pliable, etc. La rotation de composant graphique n'est pas définie pour les composants graphiques de la plateforme desktop. Dans le cadre de la migration vers des plateformes multi utilisateurs cette primitive d'interaction est indispensable.

**Primitive d'interaction 7 : Widget Resize**

Elle permet de modifier les dimensions d'un composant graphique.

---

2. Ces actions concerne les UI en 2D telles les GUI classiques

Les dimensions d'un composant graphique peuvent être redéfinies par un utilisateur d'une UI par une interaction. Cette action peut être effectuée avec un clavier ou une souris sur un desktop et à l'aide d'un geste tactile sur une table interactive.

**Primitive d'interaction 8 : Widget Selection**

Elle permet d'exprimer la sélection immédiate de composants graphiques avec un dispositif de manipulation directe.

**Primitive d'interaction 9 : Navigation**

Elle permet d'exprimer la sélection d'un composant graphique de manière séquentielle

Les primitives d'interactions **Widget Selection** et **Navigation** décrivent les différents types de sélections d'un composant graphique par un utilisateur.

Les primitives d'interactions ci-dessus<sup>3</sup> caractérisent les actions utilisateurs sur les propriétés graphiques (position, orientation, taille et représentation) d'un composant graphique.

**Primitive d'interaction 10 : Activation**

Elle permet de décrire les interactions des composants graphiques avec le NF d'une application

L'activation représente le lien entre la structure d'une UI et les fonctionnalités. Concrètement, dans le cadre d'une architecture MVC, cette primitive permet de représenter un appel de méthode du contrôleur par un élément de la vue.

**Résumé**

Les primitives d'interactions en entrée caractérisent les actions possibles des utilisateurs sur le(s) contenu(s), la taille, la position, l'orientation, et la représentation des composants graphiques d'une UI.

---

3. Widget Move, Widget Rotation, Widget Resize, Widget Selection et Navigation

En considérant l'artéfact d'UI de l'application CBA 1 par exemple, la liste d'images permet aux dessinateurs de BD de sélectionner des images prédéfinies et de l'ajouter à sa BD. Les éléments de cette liste peuvent être ajoutés au canevas de dessin par une interactions de glisser-déposer (drag-drop). Le tableau 3.1 présente les primitives d'interactions de la liste d'images et de la liste déroulante (ComboBox) de la figure 3.1

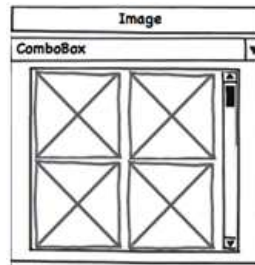


FIGURE 3.1 – Artéfact d'UI

Composants graphiques	Primitives d'interaction en entrée
Liste d'images	Widget Selection, Navigation, Widget Display, Data Selection, Data Move Out, Activation
Liste déroulante	Widget Selection, Navigation, Data Selection, Data Display, Activation

TABLE 3.1 – Exemple de primitives d'interaction en entrée

### 3.2.2 Les primitives d'interaction en sortie

Elles décrivent de manière abstraite toutes les actions du modèle ou du contrôleur sur un composant graphique à travers les interactions de type *updateView* (Réf chap.) sur les composants de type *UIBehaviourComponent* d'une vue. Elles permettent de modifier les contenus et les propriétés graphiques des composants graphiques.

#### Primitive d'interaction 11 : Data Display

Les composants graphiques sont utilisés pour afficher des données (**Data Display**) provenant du Modèle. Ces interactions concernent les mises à jour des données de la vue par le modèle ou le contrôleur.

**Primitive d'interaction 12 : Widget Display**

Les composants graphiques d'une UI sont affichés, cachés, déplacés, redimensionnés d'une vue par des interactions en sortie. La primitive d'interactions **Widget Display** représente la mise à jour des propriétés graphiques (visibilité, position, taille) d'un composant graphique de la vue.

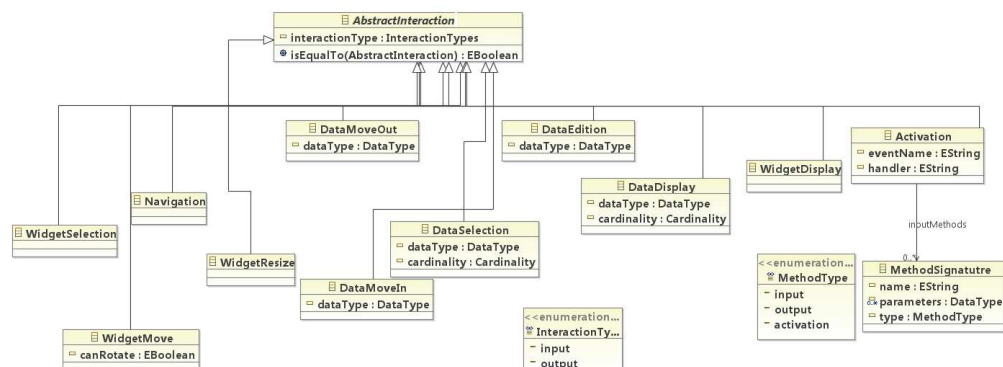
**3.2.3 Méta Modèle des primitives d'interaction**

FIGURE 3.2 – Méta Modèle des primitives d'interaction

La figure 3.2 décrit chaque primitive d'interaction comme des classes avec des attributs. La classe abstraite *AbstractInteraction* décrit une interaction abstraite, elle peut être en entrée (*InputInteraction*) ou en sortie (*OutputInteraction*). Les primitives d'interaction sont des classes dérivées de la classe abstraite *AbstractInteraction*.

Les classes *Navigation*, *WidgetSelection*, *WidgetResize*, *WidgetMove*, *WidgetRotation*, *DataMoveIn*, *DataMoveOut*, *DataSelection*, *DataEdition* et *Activation* sont des interactions en entrée. Elles représentent respectivement les primitives d'interaction *Navigation*, *Widget Selection*, *Widget Resize*, *Widget Move*, *Widget Rotation*, *Data Move In*, *Data Move Out*, *Data Selection*, *Data Edition* et *Activation*.

Les classes *DataDisplay*, *WidgetDisplay* et *WidgetHide* sont des interactions en sortie et elles représentent respectivement les primitives d'interaction *Data Display*, *Widget Display* et *WidgetHide*.

La classe *WidgetMove* décrit les différents mouvements d'un composant graphique.

La classe *WidgetRotation* permet de préciser si un composant graphique permet une rotation uniquement.

Les classes *DataSelection* et *DataDisplay* ont pour attributs les types (*dataType*) et la cardinalité (*cardinality*) des données sélectionnées et affichées.

Les classes *DataMoveIn* et *DataMoveOut* ont pour attribut le type de données concerné par

le déplacement.

La classe *DataEdition* décrit la primitive d'interaction *Data Edition* avec l'attribut *dataType* représentant le type de donnée éditée.

L'énumération *DataTypes* décrit les types de données. Les valeurs de l'attribut *dataType* sont des propriétés intrinsèques aux composants graphiques, elles ne changent pas pendant leur utilisation effective. Les types de données et la cardinalité permettent de discriminer deux primitives d'interaction de même de la même classe.

La classe *Activation* décrit la primitive d'interaction du même nom avec les attributs :

Les attributs *handler* et *inputMethods* de la primitive d'interaction *Activation* seront utilisés par le processus de migration pendant l'étape de concrétisation pour la génération des nouveaux Handlers et pour relier la vue aux contrôleurs de l'application à migrer.

La méthode **isEqualTo** : *AbstractInteraction*  $\rightarrow$  *EBoolean* de *AbstractInteraction* permet de comparer deux primitives d'interaction en prenant en compte les instances de la classe abstraite *AbstractInteraction*. Cette comparaison prend aussi en compte les différents attributs de toutes les primitives d'interaction sauf dans le cas de la primitive *Activation*. En effet les valeurs des attributs (*handler*, *inputMethods*) de la primitive d'interaction *Activation* dépendent d'une instance d'UI décrit dans *UIBehaviour* et *UIStructure*. Cette méthode est utilisée par le processus de migration pour permettre la comparaison des primitives d'interaction décrivant les composants graphiques. En effet les primitives d'interaction décrivent les interactions intrinsèques qui permettent le changement d'état des composants graphiques.

### 3.3 Les Primitives d'interaction intrinsèques aux composants graphiques

Les composants graphiques sont caractérisés par les données qu'ils contiennent, leurs propriétés graphiques et leurs comportements. Ils appartiennent à des bibliothèques graphiques qui permettent de décrire des UI graphiques 2D, des images de synthèse en 3D, jeu vidéo, des graphiques de données [BCW<sup>+</sup>06, Lon10, SWND03], etc. Dans notre processus de migration, les bibliothèques graphiques seront considérées comme des bibliothèques logicielles qui offrent des composants graphiques d'interface et qui sont spécifiques à des dispositifs d'interactions en entrée. Par exemple AWT, Swing et SWT sont spécifiques à l'environnement Java avec un clavier et (ou) une souris. Les bibliothèques graphiques offrent des API pour l'utilisation des dispositifs d'interactions (clavier, souris, écran tactile, etc.) à travers des événements tel que *Click*, *MouseDown*, *ContactDown*, *TouchDown*, etc. Les éléments d'une structure analysable d'une UI à migrer (*UIStructure*) sont des widgets d'une bibliothèque

graphique.

Les primitives d'interaction présentées à ci-dessus(cf. sections 3.2.1 et 3.2.2 ) sont utilisées pour décrire l'ensemble des interactions possibles sur les composants graphiques. Elles sont intrinsèques aux composants graphiques car elles sont identifiées à partir des comportements et des propriétés qui les caractérisent. Cette section présente un méta modèle de composant graphique que nous appelons aussi *Widget*, ainsi que les règles qui permettent d'identifier les primitives d'interaction intrinsèques à chaque composant graphique et les limites d'une représentation des primitives d'interaction intrinsèques aux composants graphiques d'une bibliothèque graphique dans le cas d'une instance d'UI.

### 3.3.1 Un méta modèle de composant graphique pour identifier les primitives d'interaction intrinsèques

Nous proposons dans cette section une formalisation des différentes caractéristiques d'un composant graphique dans un méta modèle (Figure 4-2), ceci dans le but de faciliter la définition des règles d'identification des primitives d'interaction des composants graphiques d'une UI à migrer.

*“Widget is a user interface object which defines specific interaction behaviour and a model of information presented to the user” [Cre01]*

Nous complétons cette définition en prenant en compte les caractéristiques graphiques telles que la taille, la position, l'orientation et la structure du Widget. Nous ne considérons pas les caractéristiques liées aux styles de présentation ou au layout car notre processus de migration permet aux développeurs de personnaliser l'UI pendant la migration.

**Widget** Le méta modèle de la figure 4-2 décrit un composant graphique par la classe *Widget* avec son nom (*name*) et le nombre d'éléments qu'il peut contenir (*cardinality*). Le champ *name* permet de l'identifier de manière unique dans une bibliothèque graphique. La cardinalité permet quand à elle de préciser le nombre de données ou de *Widget* maximal qu'il peut contenir. Un composant graphique nécessite dans certains cas d'autres composants graphiques pour ses contenus, l'association *containedWidget* représente les widgets utilisés comme item. Par exemple en XAML [Mic12b] les *ListBox* nécessitent des *ListBoxItem*. La classe *Widget* correspond à tous les composants graphiques d'une bibliothèque graphique.



**Behaviour** Le méta modèle de la figure 4-2 considère que les comportements des composants graphiques consistent à modifier ou à sélectionner son contenu, sa taille, sa position ou sa structure. Il considère aussi qu'un comportement fait aussi appel à des fonctionnalités. La classe *Behaviour* est un attribut de *Widget*, qui est caractérisée par les types de comportements (*type*) et les propriétés qu'elle impacte (*targetProperty*). Nous identifions trois types de comportements (*BehaviorType*) parmi les caractéristiques d'un *Widget* :

1. la modification (*Change*) de valeurs du contenu (*Content*), des propriétés graphiques (*Size, Position, Orientation*)
2. la sélection (*Select*) d'un contenu ou du *Widget* lui-même (*WidgetStructure*)
3. et l'appel d'une fonctionnalité (*Call*).

L'attribut *targetProperty* de la classe *Behaviour* permet de préciser le type de propriété modifiée par un comportement, dans le cas d'un comportement de type *Call* alors la propriété cible est nulle. Si un comportement est de type *Select* alors la propriété cible est de type *WidgetStructure* ou *Content*. En effet la sélection s'effectue uniquement sur un composant graphique ou sur son contenu. Si un comportement est de type *Change* alors la propriété cible est de type *Size, Position, Orientation* ou *Content*. En effet, le contenu, la taille, la position ou l'orientation d'un composant graphique sont modifiés.

Un comportement peut combiner plusieurs types par exemple une liste déroulante permet de sélectionner un item de la liste et aussi de déclencher une fonctionnalité. Elle aura un attribut *Behaviour* avec les types *Change* et *Call* sur la propriété de type *Content*. Un *Widget* peut avoir plusieurs comportements pour exprimer toutes ses interactions intrinsèques, en effet les comportements décrivent l'ensemble des interactions possibles d'un composant graphique, ces comportements sont spécifiques à une bibliothèque graphique.

La classe *Behaviour* est implémentée de différentes manières par les bibliothèques graphiques. Les comportements de type *Change* et *Select* sont identifiés en fonction des événements, des méthodes ou des propriétés des composants graphiques. Par exemple en Java Swing, la propriété *isEditable* permet de dire qu'un composant graphique définit de manière intrinsèque un comportement pour changer son contenu. La méthode *addActionListener* permet de dire qu'un composant graphique définit de manière intrinsèque un comportement pour faire appel à des fonctionnalités et la méthode *getSelectedItem* permet de dire qu'un composant graphique définit de manière intrinsèque un comportement pour sélectionner son contenu. Les comportements de la bibliothèque graphique XAML sont identifiés à partir des événements et des propriétés ; par exemple l'événement *SelectedText* permet d'exprimer le comportement de sélection d'un contenu, et la propriété *AllowDrop* permet d'exprimer que le contenu peut être changé.

Pour chaque bibliothèque graphique, une table décrivant les correspondances entre les types de comportement et les caractéristiques des composants graphiques qui permet de les identifier de manière intrinsèque est fournie, des exemples de cette table sont décrits de manière exhaustive pour les bibliothèques graphiques XAML et XAML Surface à l'annexe (réf).

**Property** La classe *Property* (4-2) permet de décrire les attributs statiques des *Widget* qui sont impactés par les comportements. Les types des propriétés sont identifiés de manière spécifique à chaque bibliothèque graphique en décrivant une correspondance entre les propriétés des composants graphiques et les types de propriétés génériques du méta modèle de composant graphique. Pour chaque bibliothèque graphique, il faut fournir les différents noms des propriétés des composants graphiques correspondant aux différents types (*Content*, *Size*, *Position*, *Orientation*, *WidgetStructure*).

Les contenus d'un composant graphique sont de type booléen(*Boolean*), entier (*Integer*), chaîne de caractère(*String*), image (*Image*), son ou vidéo (*MediaElement*), des classes indépendantes de la bibliothèque graphique (*Object*), des composants graphiques (*Widget*). Tout autre type n'est pas pris en compte par ce modèle de composant graphique. L'identification des types est faite à l'aide d'une table de correspondances entre les types spécifiques aux bibliothèques graphique et les types décrit dans notre modèle. L'annexe (réf) présente des exemples de cette table de manière exhaustive pour les bibliothèques graphiques XAML et XAML Surface.

Le nom d'une propriété (tel que décrit dans la bibliothèque graphique) est conservé pendant la migration (*name*), pour permettre à l'UI migré de préserver les ressources (étiquettes des widgets, etc.) de l'UI de départ, ceci permet de retrouver dans la structure analysable (*UIStructure*) la valeur de la propriété à partir de son nom.

### 3.3.2 Règles d'identification des primitives d'interaction intrinsèques

La méthode *getIntrinsicInteractions* :  $\{Attributs\} \longrightarrow \{AbstractInteraction\}$  du méta modèle (4-2) permet d'identifier les primitives d'interaction d'intrinsèque d'un *Widget*. Cette méthode se base sur des règles pour identifier chaque primitive d'interaction.

#### **Règle 1 : Widget Selection et Navigation**

**Widget Selection** et **Navigation** sont définies de manière intrinsèque pour tous les widgets.

**Règle 2 : Widget Resize**

**Widget Resize** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de changement des propriétés de type *Size*.

$$\exists prop \in Widget.attributs, prop.type = Size$$

$$\wedge$$

$$\exists beh \in Widget.attributs, beh.type = Change$$

$$\wedge$$

$$beh.targetProperty = Size$$

**Règle 3 : Widget Move**

**Widget Move** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de changement des propriétés de type *Move*.

$$\exists prop \in Widget.attributs, prop.type = Position$$

$$\wedge$$

$$\exists beh \in Widget.attributs, beh.type = Change \wedge beh.targetProperty = Position$$

**Règle 4 : Widget Rotation**

**Widget Rotation** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de changement des propriétés de type *Orientation*.

$$\exists prop \in Widget.attributs, prop.type = Orientation$$

$$\wedge$$

$$\exists beh \in Widget.attributs, beh.type = Change \wedge beh.targetProperty = Orientation$$

zz

**Règle 5 : Widget Display**

**Widget Display** est définie de manière intrinsèque pour tous les widgets.

**Règle 6 : Data Edition**

**Data Edition** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de changement des propriétés de type *Content* et dont le type de données contenu n'est pas un *Widget*.

$$\begin{aligned}
 & \exists prop \in Widget.attributs / prop.type = Content \\
 & \quad \wedge \\
 & \quad prop.dataType \notin \{Widget, Null\} \\
 & \quad \wedge \\
 & \exists beh \in Widget.attributs / beh.type = Change \wedge beh.targetProperty = Content
 \end{aligned}$$

**Règle 7 : Data Selection**

**Data Selection** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de sélection des propriétés de type *Content*

$$\begin{aligned}
 & \exists prop \in Widget.attributs, prop.type = Content \\
 & \quad \vee \\
 & \exists beh \in Widget.attributs / beh.type = Select \wedge beh.targetProperty = Content
 \end{aligned}$$

**Règle 8 : Data Move In**

**Data Move In** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de changement des propriétés de type *Content*

$$\begin{aligned}
 & \exists prop \in Widget.attributs / prop.type = Content \\
 & \quad \wedge \\
 & \exists beh \in Widget.attributs / beh.type = Change \wedge beh.targetProperty = Content
 \end{aligned}$$

**Règle 9 : Data Move Out**

**Data Move Out** est définie de manière intrinsèque pour tous les widgets qui ont un comportement de sélection et de changement des propriétés de type *Content*.

$$\begin{aligned}
 & \left\{ \begin{array}{l} \exists prop1 \in Widget.attributs / prop1.type = Content \\ \wedge \\ \exists beh1 \in Widget.attribut / beh1.type = Select \\ \wedge beh1.targetProperty = Content \end{array} \right\} \\
 & \wedge \\
 & \left\{ \begin{array}{l} \exists prop2 \in Widget.attributs, prop2.type = Content \\ \wedge \\ \exists beh2 \in Widget.attributs / beh2.type = Change \\ \wedge beh2.targetProperty = Content \end{array} \right\} \\
 & \wedge \\
 & prop1.valueDataType = prop2.valueDataType
 \end{aligned}$$

**Règle 10 : Data Display**

**Data Display** est définie de manière intrinsèque pour tous les widgets qui ont des propriétés de type *Content*.

$$\exists prop \in Widget.attributs / prop.type = Content$$

**Règle 11 : Activation**

**Activation** est définie de manière intrinsèque pour tous les widgets qui ont des comportements de type *Call*

$$\exists evt \in Widget.attributs / beh.type = Call$$

### 3.3.3 Les limites des primitives d'interaction intrinsèque d'un composant graphique

Considérons l'artéfact d'UI décrit à la figure 4-3 qui permet de parcourir une liste d'image par catégorie, les catégories sont décrites dans la liste déroulante (*ComboBox*) et les images sont affichées dans une liste (*ListBox*). On considère aussi que :

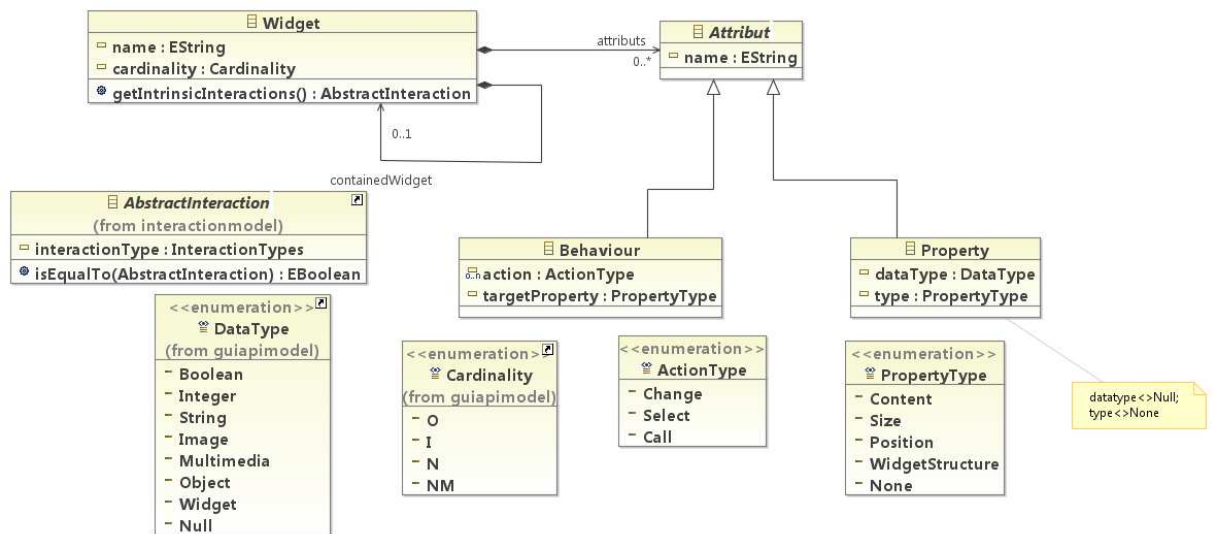


FIGURE 3.3 – Méta modèle de composant graphique

1. la liste déroulante ne permet pas de faire du drag ou du drop,
2. la liste est utilisée uniquement pour présenter les images et aucune autre interaction n'est implémentée (drag, drop, clic,...),
3. et enfin les autres widgets (Grid, Label) de l'artéfact n'implémentent aucun événement.

On considère aussi que l'UI est décrite en XAML et que les widgets *Grid*, *Label*, *ComboBox* et *ListBox* ont les primitives d'interaction intrinsèques décrites dans le tableau 4-1. La figure 5-4 présente la structure analysable de l'artéfact d'UI décrite à la figure 4-3 sous forme d'arbre.

Widgets	Primitives d'interaction intrinsèques
Grid	Widget Selection, Navigation, Widget Display, Activation
Label	Widget Selection, Navigation, Widget Display, Data Display, Activation
ComboBox	Widget Selection, Navigation, Widget Display, Data Selection, Data Move In, Data Move Out, Data Display, Activation
ListBox	Widget Selection, Navigation, Widget Display, Data Selection, Data Move In, Data Move Out, Data Display, Activation

#### 4-1 Exemple de primitives d'interaction intrinsèques

#### Limites des primitives d'interaction intrinsèques.

On remarque que les primitives d'interaction intrinsèques ne permettent pas d'exprimer le fait que *ListBox* et *ComboBox* n'implémentent pas certaines primitives d'interaction. Pour les widgets appartenant à des instances d'UI, seules les primitives d'interaction effectives doivent être exprimées. La figure ci-dessous présente les primitives d'interaction effectives de l'exemple d'UI présenté ci-dessus.

Les widgets tels que définis dans la section 5.3.1 permettent d'exprimer les comportements et les propriétés statiques des composants graphiques mais elle n'exprime pas dans le cadre des instances d'UI les liens de contenances qui permettent d'identifier les groupes de widgets.

### 3.4 Représentation générique de la structure analysable d'une UI

L'utilisation des différents modèles [Dem07, VLM<sup>+</sup>04] pour la conception et l'adaptation [Fav08, KSM99b] des UI a pour objectif de les décrire indépendamment des plate-

formes [Bra10] (dispositifs d'interaction en environnement logiciel). Ceci évite aux concepteurs d'UI d'aborder très tôt des questions liées à l'implémentation telles que le choix des bibliothèques graphiques, des widgets ou du style de présentation. Dans notre contexte, la migration concerne les UI de modalité graphique et permet d'adapter les UI des applications existantes à d'autres bibliothèques graphiques. Les modèles d'interface concrète présentés au chapitre [réf état de l'art] décrivent des interacteurs qui permettent la représentation des UI de la modalité graphique indépendamment des bibliothèques graphiques.

Le processus de migration proposé dans ce manuscrit (réf processus) a pour objectif d'assister les développeurs en charge d'adapter les UI sur des plateformes ayant des guidelines différents. La structure analysable fournie en entrée de ce processus (*UIStructure*) est un arbre qui décrit l'UI à l'aide des composants graphiques spécifiques à une bibliothèque graphique. Dans l'objectif d'adapter l'UI de départ, nous avons extrait de la structure fournie une représentation indépendante de la bibliothèque graphique qui décrit la structure de l'UI à l'aide des composants graphiques instanciés que nous appelons interacteurs []. Cette représentation nous permet de préciser pour chaque interacteur les primitives d'interaction réellement utilisées (primitives d'interaction effectives), les différents groupes d'interacteurs et les ressources de l'UI à migrer.

Cette section présente une représentation de la structure d'UI indépendante des bibliothèques graphiques qui décrit les primitives d'interaction effectives des interacteurs et les différents types de container pour décrire les groupes d'interacteur à migrer.

### 3.4.1 Primitives d'interaction effectives des interacteurs

Les primitives d'interaction réellement utilisées par chaque interacteur sont identifiées à partir de la structure analysable fournie. Cependant pour préserver l'assemblage des composants graphiques de l'UI à migrer, on extrait au moyen d'interacteurs cette structure arborescente à l'aide du méta modèle *AUIStructure*. Il décrit la structure de l'UI comme un ensemble d'arbre dont les racines sont des fenêtres, les nœuds des interacteurs et les arcs la relation de contenance entre les interacteurs. Chaque interacteur a une méthode qui permet d'identifier les primitives d'interaction effectives en fonction du nœud correspondant dans la structure analysable de départ et du modèle de *Widget*.



## Les interacteurs pour la migration d'UI vers une table interactive

Les interacteurs représentés par la classe abstraite *Interactor* (5.4) sont caractérisés par leur identifiant unique pour une instance d'UI et leur nom qui correspond au nom du composant graphique dans *UIStructure*. Ils préservent (*Content*) les valeurs des données structurales de l'UI à migrer, par exemple les étiquettes des labels, des boutons, des images, etc. Ils ont une liste des widgets équivalents pour la bibliothèque graphique d'arrivée, cette liste est calculée pendant la phase de transformation (réf chapitre transformation). On distingue deux types d'interacteurs :

1. *UIComponent* représentant l'ensemble des composants graphiques ne pouvant pas contenir de composant graphique. Ils sont identifiés à partir des widgets dont les contenus ne sont pas d'autres widgets,
2. et *Container* représentant l'ensemble des composants graphiques pouvant contenir d'autres interacteurs. Ils sont identifiés à partir des widgets pouvant contenir d'autres widgets.

## Règles d'identification des primitives d'interaction effectives

On considère que la structure analysable *UIStructure* est un ensemble d'arbres qui décrit les différentes fenêtres de l'UI à migrer.

$$UIStructure = \bigcup_{i=0}^N Tree_i$$

$$Tree_i = \langle root, Node, Contains \rangle$$

La méthode *getEffectiveInteractions*: *WidgetNode*  $\rightarrow$   $\{AbstractInteraction\}$  du méta modèle (4-6) permet d'identifier les primitives d'interaction utilisées par un interacteur. Cette méthode se base sur des règles effectives suivantes.

### Règle 12 : Widget Selection

**Widget Selection** et **Navigation** sont définies de manière effective pour les interacteurs accessible dont les propriétés de type *WidgetStructure* sont de type booléen et avec une valeur différente de false.

$$\exists prop \in Widget.attributs, prop.type = WidgetStructure \wedge \exists att \in Node / prop.dataType = Boolean \wedge att.name = prop.name \wedge node.value \neq False$$

**Règle 13 : Widget Resize**

**Widget Resize** est définie de manière effective pour les interacteurs redimensionnables dont les propriétés de type *Size* sont de type booléen et avec une valeur différente de false.

$$\exists prop \in Widget.attributes, prop.type = Size \wedge \exists att \in Node / prop.dataType = Boolean \wedge att.name = prop.name \wedge node.value \neq False$$

**Règle 14 : Widget Mode**

**Widget Move** est définie de manière effective pour les interacteurs déplaçables dont les propriétés de type *Move* sont de type booléen et avec une valeur différente de false.

$$\exists prop \in Widget.attributes, prop.type = Move \wedge \exists att \in Node / prop.dataType = Boolean \wedge att.name = prop.name \wedge node.value \neq False$$

**Règle 15 : Widget Rotation**

**Widget Rotation** est définie de manière effective pour les interacteurs orientables dont les propriétés de type *Orientation* sont de type booléen et avec une valeur différente de false.

$$\exists prop \in Widget.attributes, prop.type = Orientation \wedge \exists att \in Node / prop.dataType = Boolean \wedge att.name = prop.name \wedge node.value \neq False$$

**Règle 16 : Widget Display**

**Widget Display** est définie de manière effective pour tous les interacteurs.

Ddd

**Règle 17 : Data Edition**

**Data Edition** est définie de manière effective pour tous les interacteurs accessibles dont les widgets pour lesquels la primitive est définie de manière intrinsèque.

**Règle 18 : Data Selection**

**Data Selection** est définie de manière effective pour tous les interacteurs accessibles dont les widgets pour lesquels la primitive est définie de manière intrinsèque.

**Règle 19 : Data Move In**

**Data Move In** est définie de manière effective pour tous les interacteurs accessibles dont les widgets pour lesquels la primitive est définie de manière intrinsèque

**Règle 20 : Data Move Out**

**Data Move Out** est définie de manière effective pour tous les interacteurs qui définissent de manière effective **Data Selection** et **Data Edition**.

**Règle 21 : Data Display**

**Data Display** est définie de manière effective pour tous les interacteurs qui ont un contenu ou dont la propriété de contenu est modifiée dans une méthode.

**Règle 22 : Activation**

**Activation** est définie de manière effective pour tous les interacteurs qui font appel à une méthode de type « *inputController* » (cf. Chapitre 4)

### 3.4.2 Container

Le processus de migration consiste à déterminer les correspondances entre les composants graphiques de la plateforme de départ et celle d'arrivée. La recherche d'équivalent considère les groupes des widgets contenu dans un container pour établir les équivalences. Dans cette section nous nous intéressons à caractériser les différents types de containers dans l'objectif de les identifier plus facilement pendant la migration.

En considérant qu'un *Container* peut contenir des interacteurs de type *UIComponents* et *Containers*, alors l'on identifie d'abord les containers ne contenant que des interacteurs de type *UIComponent*, ceux ne contenant que des interacteurs de type *Container* et ceux contenant les deux types à la fois. En se basant sur le type de contenu des interacteurs nous caractérisons dans le tableau ci-dessous quatre types de container. Les types de containers sont identifiés à partir de la structure analysable de manière récursive en identifiant d'abord les types des interacteurs fils.

#### Les containers de type Window

Les *Container* racines d'une structure d'UI sont de type racine d'une structure analysable. Ils peuvent contenir tous les autres types d'interacteurs.

Eléments du container	Types de container			
	Table	Panel	Simple	Window
$\{UIComponent\}$	Si tous ses interacteurs ont des contenus de même type.	Si tous les interacteurs n'ont pas de contenu Ou Si les contenus ne sont pas de même type	Le container n'est pas de type Simple	Si le container est l'élément racine d'une UI
$\{Container\}$	Le container n'est pas de type Table	Le container n'est pas de type Panel	Si tous ses interacteurs sont des Container	
$\{UIComponent\}$  $\cup$  $\{Container\}$		Si tous les interacteurs ne sont pas des Container mais des <i>UIComponent</i> aussi	Le container n'est pas de type Simple	

TABLE 3.2 – Illustration des types de container

### Les containers de type Simple

Ce sont des containers pouvant contenir uniquement d'autres containers. Ce groupe de widgets permet d'identifier un regroupement de plusieurs containers sur le quel on pourra appliquer des règles de recommandations sur leur disposition à l'écran par exemple. Un container de type *Simple* ne peut pas contenir des interacteurs de types *UIComponent*. Le container vert de la figure 4-5 est une illustration d'un *Container* de type *Simple*.

### Les containers de type Panel

Ce sont des containers pouvant contenir des containers de type *Table*, ou des *UIComponent*. Ce groupe d'interacteurs permet d'identifier les groupes d'interacteurs pouvant contenir des données non structurées et qui sont interprétés de diverses manières pendant la migration. Le container rouge de la figure 4-5 est une illustration d'un *Container* de type *Panel*.

### Les containers de type Table

Ce sont des containers contenant uniquement des interacteurs de type *UIComponent* dont chaque interacteur a des données du même type. Un container de ce type permet de définir un groupe d'interacteurs de même type (liste, tableau, menu, etc.). Le container bleu de la figure 4-5 est une illustration d'un *Container* de type *Table*.

## 3.5 Synthèse

Les primitives d'interaction présentées dans ce chapitre permettent la représentation des interactions intrinsèques aux composants graphiques et effectives aux interacteurs. Cette représentation offre des éléments de comparaison entre interacteurs de l'UI à migrer et les composants graphiques de la plateforme d'arrivée. Les algorithmes d'équivalence utilisés par le processus de migration s'appuient sur les primitives d'équivalence pour la sélection des widgets correspondants aux interacteurs à migrer. Les interacteurs permettent de décrire les instances d'UI et leurs interactions effectives indépendamment des bibliothèques graphiques tout en conservant leurs ressources et les liens entre eux. La structure analysable *UIStructure* fournie en entrée est représentée de manière abstraite par un ensemble d'arbre d'interacteurs.

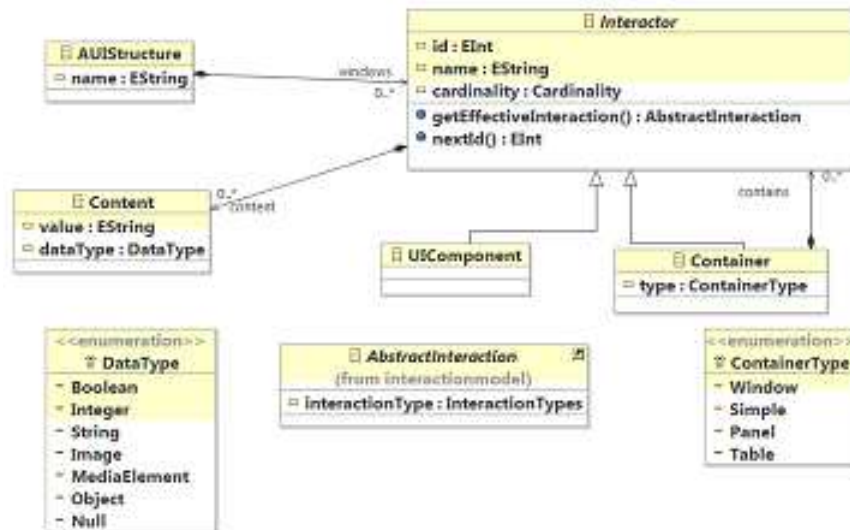


FIGURE 3.4 – Méta modèle d’interacteur

Cette structure abstraite est utilisée par le processus de migration pour établir les équivalences avec les widgets de la plateforme cible et pour identifier et transformer les groupes d’interacteurs en fonction des guidelines. Par ailleurs cette modélisation des interactions et de la structure des éléments d’une UI ne prend pas en compte toutes les préoccupations liées à la conception des UI adaptables aux contextes d’usage telles que définies par CRF. En effet, les modélisations pour l’adaptation des styles de présentation de l’UI [réf style], du layout [réf layout] des composants graphiques ou des comportements de l’UI de façon globale [réf tâche] ne sont pas prises en compte dans la modélisation proposée dans ce chapitre. Le processus de migration que nous proposons permet aussi d’assister les développeurs pendant le choix des styles et présentation ou du layout suivant les guidelines de la plateforme d’arrivée.

## **Chapitre 4**

### **Prise en compte des guidelines**

# **Chapitre 5**

## **Prototype**



## **Troisième partie**

### **Conclusion et Perspectives**

## **Chapitre 6**

## **Conclusion**

# Bibliographie

- [App95] Apple Computer Inc. *Macintosh Human Interface Guidelines*. Addison-wesley Publishing, 1995.
- [BCR10] Thiago Tonelli Bartolomei, Krzysztof Czarnecki, and L Ralf. Swing to SWT and Back : Patterns for API Migration by Wrapping. 2010.
- [BCW<sup>+</sup>06] Doug A Bowman, Jian Chen, Chadwick A Wingrave, John Lucas, Andrew Ray, Nicholas F Polys, Qing Li, Yonca Haciahetoglu, Seonho Kim, Robert Boehringer, and Tao Ni. New Directions in 3D User Interfaces. *The International Journal of Virtual Reality*, (0106) :1–30, 2006.
- [Bes10] Guillaume Besacier. *Interactions post-WIMP et applications existantes sur une table interactive*. PhD thesis, UNIVERSITÉ PARIS-SUD 11, 2010.
- [BL00] Michel Beaudouin-Lafon. Instrumental interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, pages 446–453, New York, New York, USA, April 2000. ACM Press.
- [Bra10] Giorgio Brajnik. Is the UML appropriate for Interaction Design ? 2010.
- [BRNB07] Guillaume Besacier, Gaétan Rey, Marianne Najm, and Stéphanie Buisine. Paper Metaphor for Tabletop Interaction Design. In *HCII'07 Human Computer Interaction International*, pages 758–767, 2007.
- [CCB<sup>+</sup>02] Gaëlle Calvary, Joëlle Coutaz, Laurent Bouillon, Murielle Florins, Quentin Limbourg, L. Marucci, Fabio Paternò, Carmen Santoro, N. Souchon, David Thevenin, and Jean Vanderdonckt. CAMELEON Project. Technical report, 2002.
- [Cre01] Murray Crease. *A Toolkit Of Resource-Sensitive, Multimodal Widgets Murray Crease*. PhD thesis, 2001.
- [D. 06] D. Heinemeier Hansson. World of Resource, 2006.
- [Dem07] Alexandre Demeure. Modèles et outils pour la conception et l'exécution d'Interfaces Homme-Machine Plastiques. 2007.
- [DL01] Paul Dietz and Darren Leigh. DiamondTouch. In *Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01*, page 219, New York, New York, USA, November 2001. ACM Press.

- [Fav08] Jean-marie Favre. A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces. *Ifip International Federation For Information Processing*, pages 140–157, 2008.
- [FBBN07] Franck Fleurey, Erwan Breton, Benoit Baudry, and Alain Nicolas. Model-Driven Engineering for Software Migration in a Large Industrial Context. pages 482–497, 2007.
- [FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. Computer graphics : principles and practice (2nd ed.). July 1990.
- [GH95] Hans-w Gellersen and Hans-W. Gellersen. Modality Abstraction : Capturing Logical Interaction Design as Abstraction from “User Interfaces for All”. In *1st ERCIM Workshop on “User Interfaces for All”*. ERCIM, 1995.
- [IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 234–241, New York, New York, USA, March 1997. ACM Press.
- [KKM03] M. Kassoff, D. Kato, and W. Mohsin. Creating GUIs for web services. *IEEE Internet Computing*, 7(5) :66–73, September 2003.
- [KLL<sup>+</sup>09] Sébastien Kubicki, Sophie Lepreux, Yoann Lebrun, Philippe Dos Santos, Christophe Kolski, and Jean Caelen. *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*, volume 5612 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [KP88] Glenn E Krasner and Stephen T Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. Technical report, 1988.
- [KSM99a] L Kong, E Stroulia, and B Matichuk. Legacy interface migration : A task-centered approach. ... of the 8th International Conference on ... , 1999.
- [KSM99b] Lanyan Kong, Eleni Stroulia, and Bruce Matichuk. Legacy Interface Migration : A Task “ Centered Approach. 1999.
- [Kub11] S  bastien Kubicki. *Contribution   la prise en consid  ration du contexte dans la conception de tables interactives sous l’angle de l ’IHM , application   des contextes impliquant table interactive RFID et objets tangibles*. PhD thesis, Universit   de Valenciennes, 2011.
- [Lon10] Nguyen Hoang Long. *Web Visualization of Trajectory Data using Web Open Source Visualization Libraries*. PhD thesis, INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION ENSCHEDE, THE NETHERLANDS, 2010.
- [McC92] Carma McClure. The three Rs of software automation : re-engineering, repository, reusability. March 1992.

- [Mic11] Microsoft. Microsoft Surface 2 Design and Interaction Guide. Technical Report July, 2011.
- [Mic12a] Microsoft. MSDN XAML, 2012.
- [Mic12b] Microsoft WPF. WPF, 2012.
- [Mic12c] Microsoft XNA. XNA, 2012.
- [Mit] Mitsubishi Electric Research Laboratoriesb. MERL .
- [Moo95] James D. Mooney. Portability and reusability. In *Proceedings of the 1995 ACM 23rd annual conference on Computer science - CSC '95*, pages 150–156, New York, New York, USA, February 1995. ACM Press.
- [Moo96] Melody M. Moore. Rule-Based Detection for Reverse Engineering User Interfaces. page 42, November 1996.
- [MR97] Melody Moore and Spencer Rugaber. Using Knowledge Representation to Understand Interactive Systems. pages 60–67, May 1997.
- [MVL06] José Pascual Molina Massó, Jean Vanderdonckt, and Pascual González López. Direct manipulation of user interfaces for migration. In *Proceedings of the 11th international conference on Intelligent user interfaces - IUI '06*, page 140, New York, New York, USA, January 2006. ACM Press.
- [NC94] Laurence Nigay and Joëlle Coutaz. *Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales = Software design and implementation of interactive systems : a case study of multimodal interfaces*. PhD thesis, 1994.
- [Nig94] Laurence Nigay. *Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales*. PhD thesis, UNIVERSITÉ JOSEPH FOURIER - GRENOBLE 1, 1994.
- [Pfa85] G. E. Pfaff. User Interface Management Systems. June 1985.
- [PHR02] Jenny Preece, Sharp Helen, and Yvonne Rogers. *Interaction Design : Beyond Human-Computer Interaction*. Chichester edition, 2002.
- [PSS09] Fabio Paternò, Carmen Santoro, and Lucio Davide Spano. MARIA : A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. *ACM Transactions on Computer-Human Interaction*, 16(4) :1–30, November 2009.
- [PZ10] Fabio Paternò and Giuseppe Zichittella. Desktop-to-Mobile Web Adaptation through Customizable Two-Dimensional Semantic Redesign. pages 79–94, 2010.
- [RBB<sup>+</sup>95] Lori Alperin Resnick, Alex Borgida, Ronald J Brachman, Charles L Isbell, Deborah L Mcguinness, Peter F Patel-schneider, and Kevin C Zalondek. CLASSIC Description and Reference Manual For the COMMON LISP Implementation. Technical report, 1995.

- [RFJ08] Daniel Ratiu, Martin Feilkas, and Jan Jurjens. Extracting Domain Ontologies from Domain Specific APIs. *2008 12th European Conference on Software Maintenance and Reengineering*, 1 :203–212, 2008.
- [SC12] Carlos Eduardo Silva and José Creissac Campos. Can GUI Implementation Markup Languages Be Used for Modelling ? In *Human-Centered Software Engineering*, pages 112–129, 2012.
- [Ste12] Gideon Steinberg. Natural User Interfaces. 2012.
- [SVFR04] Chia Shen, Frédéric D. Vernier, Clifton Forlines, and Meredith Ringel. DiamondSpin : an extensible toolkit for around-the-table interaction. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*, pages 167–174, New York, New York, USA, April 2004. ACM Press.
- [SWND03] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL programming guide : the official guide to learning OpenGL, version 1.4*. 4th ed edition, 2003.
- [TC02] David Thevenin and Joëlle Coutaz. Adaptation des IHM : Taxonomies et Archi. Logicielle. In *IHM 2002*, pages 207–210, 2002.
- [TKB78] Andrew Tanenbaum S., Paul Klint, and Wim Bohm. Guidelines for Software Portability. *Software-Practice And Experience*, 8(6) :681–698, November 1978.
- [UI97] Brygg Ullmer and Hiroshi Ishii. The metaDESK : Models and Prototypes for Tangible User Interfaces. In *UIST '97 Proceedings of the 10th annual ACM symposium on User interface software and technolog*, pages 223 – 232, 1997.
- [UIM92] UIMS. A METAMODEL FOR THE RUNTIM E ARCHITECTURE OF AN INTERACTIVE SYSTE M The UIMS Tool Developers Workshop \*. In *SIG-CHI Bulletin*, number January, 1992.
- [Val89] Valdès. A Virtual Toolkit for Windows and the Mac. *Byte*, 11(3) :209–216., 1989.
- [Van97] Jean Vanderdonckt. Conception assistée de la présentation d’une interface homme-machine ergonomique pour une application de gestion hautement interactive. 1997.
- [vD97] Andries van Dam. Post-WIMP user interfaces. *Communications of the ACM*, 40(2) :63–67, February 1997.
- [VLM<sup>+</sup>04] Jean Vanderdonckt, Quentin Limbourg, Benjamin Michotte, Laurent Bouillon, Daniela Trevisan, and Murielle Florins. USIXML : a User Interface Description Language for Specifying Multimodal User Interfaces The Reference Framework used for Multi-Directional UI Development. *Language*, pages 19–20, 2004.
- [VNHF11] Bram J J Van Der Vlist, Gerrit Niezen, Jun Hu, and Loe M G Feijs. Interaction Primitives : Describing Interaction Capabilities of Smart Objects in Ubiquitous Computing Environments. Number September, pages 13–15, 2011.

- [W3C03] W3C. W3C Multimodal Interaction Framework, 2003.
- [WCK<sup>+</sup>] Candy Wong, Hao-hua Chu, Masaji Katagiri, Seamless Experience, and San Jose. GUI Migration across Heterogeneous Java Profiles. *Architecture*.
- [Weg97] Peter Wegner. Why Interaction Is More Powerful Than Algorithms, 1997.
- [WGM08] Xin Wang, Yaser Ghanam, and Frank Maurer. From Desktop to Tabletop : Migrating the User Interface of AgilePlanner. In *Engineering Interactive Systems 2008*, pages 263–270, 2008.
- [XSS<sup>+</sup>04] Steven Xia, David Sun, Chengzheng Sun, David Chen, Q Xia, D Sun, C Sun, and D Chen. Leveraging Single-user Applications for Multi-user Collaboration : the CoWord Approach Categories and Subject Descriptors. In *CSCW 2004*, pages 162–171, 2004.