Première partie

Domaine d’étude et état de l’art

CHAPITRE 1

Tables interactives et Migration d'UI

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 **??**) pour identifier les problématiques liées à la migration d'UI. Ensuite la section **??** 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 **??** quant à elle aborde des concepts utiles pour la migration des UI. Enfin la section **??** 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 **??** présente une répartition des applications conçues pour les tables interactives en 2011 [**?**] ; 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 **??**) 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 **??**) et enfin les questions soulevées par la migration d'une UI vers les tables interactives (au paragraphe **??**).

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 **??** 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.

1.1.2.1 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[**?**]. 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'enchainement des fenêtres et des activités de l'UI),
- et évaluer le respect des critères ergonomiques de l'UI générée.

1.1.2.2 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 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 de départ. Et 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 **??** peuvent être précisées d'avantages. En considérant l'application CBA (cf section **??**), 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 [**?**]. 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 **??**, 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, tangiles) 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 CoWord [**?**] 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 [**?**]. 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 **??** 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 **??**. 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 PixelSense, 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

1.2.1.1 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 DiamonTouch [**?**] qui est l'une des première table interactive utilisée dans un cadre non expérimental, elle fut industrialisée par MERL [**?**], 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 metaDesk [**?**] qui supporte que des objets tangibles comme moyen d'interaction. Enfin nous nous intéresserons aux tables Microsoft PixelSense 1.0 et 2.0 [**?**] 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 DiamonTouch sont une surface tactile non capacitif 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 utilisateus 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 connaitre le nombre d'utilisateurs de l'UI migrée car chaque utilisateur doit être assis sur une chaise pour utiliser la table(cf. figure **??**). 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 Diamond-Touch et permet à chaque utilisateur d'être mobile autour de la table. Cette Table permet de décrire des UI tangibles (TUI).

Ulmer et Ishii [**?**] 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 [**?**], les concepteurs associent une lentille à une fenêtre, un plateau à un menu, etc. comme l'indique la figure **??** d'instanciation physique des éléments GUI de metaDESK de la figure **??**. 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 interactions 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 [**?**] sont en générale conçues pour faire de la réalité augmentée pour des applications de cartographie par exemple.

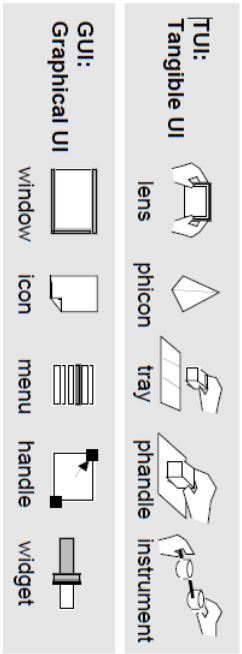

FIGURE 1.6 – Instanciation physique des éléments GUI dans TUI

Microsoft PixelSense Les moyens d'interaction de la table interactive Microsoft PixelSense 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 sup-

portent 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 [1]. 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.

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 PixelSense 1.0 et 2.0) ou de manière verticale (pour Microsoft PixelSense 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.

1.2.1.2 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 [**?**] 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

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

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 **??**) 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 [**?**] 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 [**?**] et XNA [**?**]. 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 **??**) définie 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 [**?**]. 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.

1.2.1.3 Modalités d'interactions

Nigay [**?**] propose une définition de la modalité d'interaction comme un couple <d, l> constitué d'un dispositif d'interaction et d'un langage d'interaction [**?**] : - 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 **??** les problèmes liées aux changements de modalités d'interactions pendant la migration. Et au paragraphe **??** 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 Tangibles, 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 [**?**] propose une boîte à outils décrivant des widgets multimodales et indépendantes des dispositifs d'interactions en sortie. Les widgets de Crease [**?**] 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.

1.2.2.1 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 [2] ou aux composants graphiques [3] 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.

1.2.2.2 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.

1.2.2.3 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 **??** 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 [**?**]. 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 [**?**].

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.

1.2.3.1 Corpus des principes de conception d'UI pour Microsoft PixelSense

Les principes de conception d'UI pour une Microsoft PixelSense sont décrits sous forme de guidelines (ou recommandations) dans le document *User Experience Design Guideline* [**?**]. 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 [**?**] 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.

1.2.3.2 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 **??** 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 **??**). 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 **??**, 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 **??** 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.

1.2.3.3 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 **??** caractérisant le nombre d'utilisateurs et à la propriété **??** qui caractérise leur répartition autour de la surface d'affichage. Elles sont aussi liées à la propriété **??** et à la propriété **??** 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 **??**), l'organisation de la surface de travail (qui est spécifiée par la guideline **??**) et l'accessibilité des éléments d'une UI collaborative (qui est spécifiée par la guideline **??**).

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

Elle est une spécification de la guideline **??**, car elle préconise dans le cas d'une UI multi utilisateurs :
– d'éliminer toute 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 **??** 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 [**?**] par exemple permet aux concepteurs d'associer un utilisateur à une zone de l'écran. Cependant la table Microsoft PixelSense quant à elle ne permet pas une division de l'écran en zones associées aux utilisateurs ou à des fonctionnalités. La figure **??** 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.

Guideline 7 *Utilisation 360 degré de l'UI*

Cette guideline permet de spécifier la guideline **??** 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 **??** illustre à un cas d'utilisation des composants graphiques utilisables à 360°.

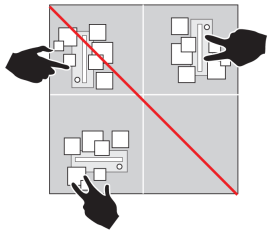

FIGURE 1.9 – Illustration de partage d'espace entre plusieurs utilisateurs

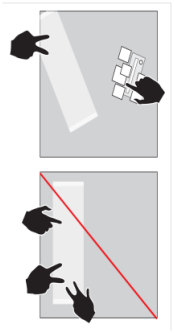

FIGURE 1.10 – Illustration de la propriété 360°

En considérant que l'UI de l'application CBA (cf. figure **??**) 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 **??** préconisant le partage de l'espace de travail et la guideline **??** préconisant l'utilisation des objets 360 degré permettent de décrire une UI sans layout avec des groupes d'éléments utilisables à 360°. La guideline **??** par exemple permettra de supprimer les composants graphiques bloquants tels que les boites de dialogues de l'UI de l'application CBA.

1.2.3.4 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 **??**) ne constituent pas le corpus des guidelines applicables pendant la migration d'une UI vers une table interactive. En effet les apects 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 [**?**] 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 **??** de tactibilité 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 styles - 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 diposition 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 **??**.

Formulaire de l'application de départ	Formulaire migré en prenant en compte les guidelines de styles

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

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

Le corpus de guidelines décrit dans la section **??** 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 **??** 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 apects 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 **??**). 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 [**?**] qui se base sur des composants logiciels spécifiques et indépendants des plateformes, ensuite le modèle MVC [**?**] .

1.3.1.1 ARCH

ARCH [**?**] est un modèle d'architecture qui se base sur des composants conceptuels du modèle de Seeheim [**?**]. Comme l'indique la figure **??**, 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'enchainement 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 [**?**] 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. [**?**] 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 [**?**]. 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 bibliothèques 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 [**?**], les guidelines identifiées à la section **??** 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.

1.3.1.2 MVC

MVC [**?**] 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 **??** 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.

1.3.1.3 Résumé

Nous avons présenté dans cette section **??** 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 **??**. 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.

1.3.2.1 Modèle de Vlist

Vlist et al. [**?**] 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.).

1.3.2.2 Modèle de Gellersen

Gellersen [**?**] 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 **??** 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*[4] et de *Signal*[5] 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.

FIGURE 1.14 – Modèle d'interactions abstraites de Gellersen

Dans le cadre du changement de modalité d'interaction entrainé 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 commandes. 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 nécéssite la spécification de certaines interactions indépendantes des modalités d'interactions pendant la migration.

1.3.2.3 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

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)

– 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

2.1 Introduction 5
2.2 Approches spécifiques de migration d'UI 6
2.2.1 Migration de l'application AgilePlanner vers une table interactive 6
2.2.2 Portage des applications existantes sur des tables interactives 8
2.3 Approches de migration basées sur les modèles de l'UI 10
2.3.1 Migration avec MORPH 11
2.3.2 Approches de migration automatiques d'UI 14
2.4 Synthèse et objectifs 18
2.4.1 Synthèse 18
2.4.2 Objectifs 19

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 [**?**] 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. [**?**] 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 [**?**]. Le forward engineering est une concrétisation de la représentation abstraite d'un système dans une implémentation. Une approche de migration d'UI vers une table interactive implique un ré engineering de l'UI source en impliquant d'abord une phase de reverse engineering de l'UI source ensuite une phase de forward engineering.

Dans notre contexte, les SI à migrer ont une décomposition fonctionnelle minimale qui comprend une UI et un NF. Cette décomposition facilite la migration de l'UI en réutilisant le NF de l'application de départ par exemple. Les solutions de migration d'UI d'une application qui prennent en compte les guidelines de la plateforme cible peuvent être manuelles et spécifiques dans le cas où le concepteur se charge de transformer l'UI de départ pour la plateforme cible. Ces solutions peuvent aussi être automatisées par des processus qui se basent sur des modèles et des principes génériques pour une classe d'applications ou de plateformes.

Nous présentons dans ce chapitre des approches de migration d'UI dans le but de savoir comment sont adaptés les différents aspects des UI par ces approches de migration. Pour répondre à cette question, nous nous basons sur trois critères importants pour définir une solution réutilisable pour la migration d'UI vers les tables interactives.

1. **Les équivalences** entre les dispositifs d'interactions, les bibliothèques graphiques des plateformes source et cible.
2. **La modélisation** des différents aspects des UI : Structure, Interactions, Comportement, Layout, Style.
3. La prise en compte des **Guidelines** par les mécanismes de migration d'UI pour garantir une UI conforme aux critères d'utilisabilité.

Dans ce chapitre, la section 2.2 présente les approches de migration spécifiques à des applications ou à des boîtes à outils. La section 2.3 présente les approches de migration d'UI basées sur des modèles d'UI. La section 2.3 fait une synthèse des différentes approches de migration 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 à 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 [1] vers une table interactive [**?**]. Le processus mis en place était basé sur quatre phases.

La première phase consistait à analyser l'UI de l'application à migrer, elle permettait d'identifier les différentes zones (menu, légendes, zones d'interaction, espace de travail, etc.).

La deuxième phase consistait à évaluer l'UI de l'application à migrer sur une table interactive, le but de cette évaluation était 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 est ressorti les 7 guidelines suivantes :

- Avoir des composants d'UI de l'application déplaçables et utilisables 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
- Pouvoir utiliser des composants graphiques de taille assez grande 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 présentées au chapitre précédent (cf. section **??**).

La troisième phase consistait 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

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 dans le cadre de plannings de projets par exemple

reconnaissances gestuelle et vocale étaient fournies par l'environnement logiciel de certaines tables interactives.

La dernière phase du processus consistait à évaluer l'UI produite en demandant aux utilisateurs finaux d'évaluer l'utilisabilité de chaque fonctionnalité migrée de l'application AgilePlanner. L'UI migrée a été modifiée par rapport aux remarques issues des évaluations. Enfin l'UI a été réévaluée à nouveau pour mesurer la satisfaction des testeurs. Les résultats de cette réévaluation ont montré que l'utilisabilité de l'écriture à main levée sur la table interactive peut être améliorée.

La migration manuelle d'une UI est un processus qui comporte plusieurs phases. Dans cette section nous avons présenté une approche qui nous permet 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 et améliorer le résultat obtenu.

2.2.1.1 Réutilisation du processus pour d'autres applications

Les étapes du processus de migration de l'application AgilePlanner vers une table interactive et la mise en œuvre des guidelines identifiées constituent des "savoir faire" qu'un développeur souhaiterais réutiliser pour d'autres applications.

Les mécanismes d'adaptation de l'application AgilePlanner ne peuvent pas être réutilisés pour d'autres applications car ils sont spécifiques à cette application. En effet, l'analyse et l'identification des différentes zones de l'UI ne s'appuient pas sur des **modèles indépendants de la plateforme**.

La mise en œuvre des guidelines identifiées dans ce processus est basée sur l'**intuition**. Nous avons considéré l'UI de deux applications : l'application CBA et l'UI d'une application agenda présentée à la figure 2.1. En appliquant les guidelines identifiées à la section 2.2.1 pour migrer ces deux UI. En ce qui concerne l'application agenda, les composants graphiques représentant la barre de menu, la barre de recherche, la liste de catégorie et le tableau peuvent être tous déplaçables sur la table ou bien 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 **??** par exemple.

Nous remarquons que l'utilisation des guidelines pour deux applications distinctes 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

2.2.1.2 Résumé

Les mécanismes de migration du processus décrits dans cette section, tels que les équivalences des composants graphiques ou des dispositifs d'interactions des plateformes source et cible sont manuels. Et les mécanismes de prise en compte de guidelines sont aussi manuels et basés sur les **connaissances** des personnes en charge de la migration.

Par ailleurs si l'on souhaite réutiliser ce mécanisme de migration d'UI, il est indispensable de formaliser les guidelines en s'appuyant sur des modèles indépendants des plateformes source et cible (tels qu'un menu, un formulaire, etc.) et en décrivant des règles de transformations des modèles décrivant les aspects de l'UI concernés par la migration (layout, interactions, styles, etc.)

2.2.2 Portage des applications existantes sur des tables interactives

Cette approche [**?**] a pour but de réutiliser des applications pour desktop sur une table interactive sans reconception de l'UI, l'objectif est d'adapter les différents moyens d'interactions (clavier et souris) des desktops sur une table interactive DiamondTouch en utilisant plusieurs types de technologies. L'approche regroupe des technologies qui prennent en compte des guidelines liées à la conception des UI collaboratives. Besacier [**?**] a caractérisé six technologies qui permettent la réutilisation des applications existantes sur une table interactive.

La capture d'écran est une technologie qui consiste à prendre une capture de l'écran de l'UI de départ à intervalle régulier, la capture d'écran enregistre une copie exacte de l'image affichée sur l'écran du desktop dans une zone mémoire où elle peut être modifiée pour être adaptée à l'environnement de la table interactive. Cette approche se situe dans un contexte de portage des applications existantes. Elle ne permet pas de porter des interactions en entrée car l'UI portée est constituée d'images capturées et ces images ne contiennent pas de méta données.

La carte graphique virtuelle est une amélioration de la technologie basée sur la capture d'écran. L'approche utilise le serveur Metisse [] qui stocke les fenêtres de l'application de départ sous forme d'images. Le serveur Metisse joue un rôle de carte graphique virtuelle car il redessine les fenêtres sur l'écran de la table interactive à travers un compositeur et il redirige les événements venant d'une souris ou d'un clavier vers l'application. Cette technologie permet de porter les fenêtres des applications desktops sur une table interactive en offrant la possibilité aux utilisateurs de les manipuler facilement. Cependant les composants graphiques des fenêtres portées ne sont pas interactifs car ils restent des images de l'UI de départ.

L'émulation du clavier et de la souris est une technologie qui améliore les deux technologies présentées ci-dessus. Elle permet de porter aussi les interactions en entrée (du clavier et de la souris). Cette technologie se base sur des techniques qui permettent d'utiliser plusieurs pointeurs avec applications Java existante []. Les problèmes liés aux passages d'une UI mono utilisateur à une UI multi utilisateurs sur la table interactive sont partiellement résolus par cette approche. En effet, la technique permet de porter plusieurs applications à la fois sur une table interactive et à chaque application est associée un utilisateur avec son clavier et sa souris virtuels. Dans le cas d'une UI mono utilisateur que l'on souhaiterait utiliser en multi utilisateurs, cette technique ne permet pas de multi touches simultanées car l'interaction d'un utilisateur verrouille la table à d'autres utilisateurs d'une même application.

Le langage de script est une technologie qui permet de porter des UI des applications existantes sur une table interactive en résolvant les limites liées aux multi touches et multi utilisateurs de la technologie précédente. Le langage de script est spécifique à une application et consiste à écrire dans un scénario un ensemble de scripts qui font appel chacun à une fonctionnalité de l'application à porter.

Les scripts sont appelés par une application de table interactive qui capture les événements en entrée de la table et les associes à un script précis. Par exemple pour afficher un fichier sur la table, l'application table interactive associe un script d'ouverture et de lecture de fichiers de l'application de départ à l'événement lancé après la pause d'une feuille de papier sur la table interactive. Cette technique n'est pas facilement réutilisable pour d'autres applications car les scripts sont écrits en fonction des applications. Les données échangées entre l'application à migrer et l'application table interactive sont décrites dans un modèle. Ce modèle de données permet de faire le mapping entre les données issues des composants graphiques de l'application à miger et les composants graphiques (DiamondSpin) de la table interactive. Les guidelines de la métaphore du papier [2] [**?**] sont prises en compte dans l'implémentation de cette approche qui considère la bibliothèque graphique DimaondSpin de la table DiamondTouch. Cette prise en compte des guidelines dépend fortement des connaissances et de l'intuition du concepteur en charge du portage.

Les API d'accessibilité numérique [3] permettent d'obtenir des informations sémantiques à propos des UI graphiques en cours d'exécution. Ces informations constituent des instances de modèles de structure d'une UI donnée (comprenant les composants graphiques, les données, les positions, etc.). Ce modèle permet de décrire une technique de migration qui segmente les images obtenues par capture d'écran de l'UI source. Les parties de l'UI de départ peuvent donc être dupliquées ou juxtaposées en se basant sur le modèle de structure et les images segmentées. Le modèle de structure et les images segmentées sont les éléments de l'UI source qui permettent de générer l'UI cible. Comparée aux technologies précédentes, cette technologie est moins réutilisable que la capture d'écran, la carte graphique virtuelle et l'émulation des dispositifs d'interactions.

La réécriture d'une boite à outils d'interface homme machine 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 de l'application vers les objets de la boîte à outils d'IHM de départ et en les redirigeant vers les éléments de la boîte à outils de la table interactive. L'avantage de cette approche est d'utiliser des composants graphiques respectant les guidelines de la table interactive sans une nouvelle conception de l'application de départ. Cette solution peut être implémentée en utilisant les wrappers [**?**] par exemple, le wrapper du listing **??** permet de réécrire un *JComboBox* en DSJComboBox. Cette approche ne permet pas le respect de l'ensemble des guidelines car la structure et le layout de l'UI par exemple n'est pas modifiée pendant le portage. Elle permet de choisir les composants graphiques qui ont des interactions ou des styles conformes à la table interactive.

2.2.2.1 Résumé

Il existe plusieurs solutions de portage des applications existantes sur les tables interactives. Les technologies présentées ci-dessus permettent la migration à l'exécution des applications sources. Ces approches sont limitées car elles sont faiblement réutilisables avec d'autres applications ou d'autres boîtes à outils.

Le tableau 2.1 présente une synthèse de ces technologies en prenant en compte les critères d'équivalences entre les éléments de la plateforme, les modèles utilisés et la prise en compte des guidelines. L'on que remarque les équivalences entre les dispositifs d'interactions sont décrites manuellement

2. déplacement, rotation, redimensionnement, changement d'échelle, pliage

3. Elles sont utilisées pour offrir des interfaces alternatives à des groupes d'utilisateurs spécifiques, par exemple des utilisateurs souffrant de déficience visuelle

Approches	Équivalences	Modélisations	Guidelines
Capture d'écran	Aucune équivalence	Aucune modélisation	Aucune prise en compte
Carte graphique virtuelle	Aucune équivalence	Aucune modélisation	Aucune prise en compte
Émulation du clavier et de la souris	Équivalences entre les dispositifs d'interactions en entrée des plateformes source et cible définies manuellement	Aucune modélisation	Aucune prise en compte
Langage de script	Équivalence manuelle des dispositifs d'interactions	Modélisation des données, approche non réutilisable	Prise en compte des guidelines
API d'accessibilité numérique	Manuelles	Modélisation de la structure	Prise en compte des guidelines
Réécriture d'une boite à outils	Manuelles	Aucune	Prise en compte partielle

TABLE 2.1 – Synthèse des approches de portage d'UI sur tables interactives

pendant la mise en œuvre de la solution. Et toutes ces approches n'utilisent une modélisation des interactions et seulement deux approches modélisent la structure de l'UI. Les prises en compte des guidelines dépendent du l'utilisation ou non de la boîte à outils cible. En effet le langage de script, les API d'accessibilité et la réécriture d'une boîte à outils permettent d'utiliser les éléments de la boîte à outils cible. Cependant dans le cas de la réécriture d'une boîte à outils, la prise en compte est partielle car la structure et le layout de l'UI de départ ne sont pas modifiés pendant le portage.

2.3 Approches de migration basées sur les modèles de l'UI

Cette section présente des approches de migration basées sur des modèles pour être indépendantes des applications et des plateformes. En effet les solutions de portage d'UI sur une table interactive présentées à la section 2.2.2 utilisent des modèles de données et de structure de l'UI de départ pour générer l'UI cible. Ces modèles comportent des informations sémantiques qui incluent pour chaque composant son nom, son rôle (bouton menu, case à cocher etc.), sa position sur l'écran et dans la hiérarchie en terme de contenant et de contenu. Cependant ces informations sont exploitées par des mécanismes non réutilisables et spécifiques à des applications car ces mécanismes sont constitués de scripts spécifiques à une fonctionnalité d'une UI.

Dans cette section nous étudions les approches de migration qui se basent sur des modèles qui décrivent les différents aspects de l'UI. D'abord l'approche MORPH *(Model Oriented Reengineering Process for HCI)* [**?**] qui modélise la structure et les interactions et qui décrit des mécanismes de transformations de ces modèles. Ensuite les approches basées sur les services de migration qui modélisent la structure, le layout, les interactions et le comportement des UI à migrer.

2.3.1 Migration avec MORPH

MORPH [**?**] est une solution 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 [**?**] supportent les dispositifs d'interactions de manipulations 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.

2.3.1.1 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 de migration en trois étapes : la détection, la représentation et la transformation (cf. figure 2.2).

- 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 de structure et d'interactions.
- 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 [4], augmenter [5], restructurer [6] les modèles abstraits de l'UI source pour être utilisables dans l'environnement cible.

FIGURE 2.2 – Processus de migration avec MORPH

2.3.1.2 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 :

- le niveau des **tâches d'interactions** qui regroupe quatre interactions de base d'une UI [**?**] qui sont : -la sélection dans une liste, -la quantification ou la saisie d'une donnée numérique, -l'indication de la position [7] 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.), ce modèle est spécifique aux UI graphiques.

4. Elle consiste à modifier les aspects visuels de l'UI à travers les modèles
5. Elle consiste à ajouter des composants graphiques ou des fonctionnalités à l'UI de départ
6. Elle consiste a modifier les types de données ou la structure hiérarchique de l'UI
7. Sur une UI textuelle la position est exprimée en termes de ligne et de colonne pour positionner le curseur du clavier par exemple

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

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

Les objets d'interactions abstraits sont raffinées à partir des attributs les différentes tâches d'interactions et en se basant sur le langage de représentation des connaissances CLASSIC [**?**].

Extraction des modèles abstraits

Les tâches d'interactions et les objets d'interactions abstraites sont identifiés à partir d'UI textuelles en se basant sur l'architecture de l'application à migrer et sur des **règles d'identifications** [**?**, **?**]. Pour identifier les tâches d'interactions Moore et al. [**?**] décrivent les règles d'identifications sous la forme :

Si **Condition** *est vraie*; *Alors identifier une* **Tâche d'interactions**

où **Condition** correspond à une situation d'une instance d'UI qui représente une **Tâche d'interactions** du modèle abstrait. Cette forme de règle nécessite que l'ensemble des situation soit identifiées par un mapping entre la boîte à outils de l'UI source et les interactions abstraites.

Il existe d'autres approches pour extraire un modèle à partir d'une UI existante, par exemple Ratiu et al. [**?**] proposent une ontologie pour analyser et comprendre des API spécifiques à un domaine comme les bibliothèques graphiques. Le principe de l'approche consiste d'abord à construire une ontologie capable de représenter les concepts d'une bibliothèque graphique que l'on souhaite utiliser pour la migration ; dans notre cas ce sont la structure et les interactions abstraites d'une UI. Dans notre cas l'ontologie décrirait les liens de contenance entre les composants graphiques, les données et les tâches interactions en entrée (sélection, quantification, position, édition).

Ensuite, à partir d'une UI décrite à l'aide des instances des éléments d'une bibliothèque graphique, l'on extrait grâce à l'ontologie les objets d'interactions abstraites. Cette extraction est possible grâce à un algorithme d'extraction décrit dans [**?**]. Les objets d'interactions abstraites de l'UI source sont utilisés comme pivot pour décrire l'UI cible.

Cette approche d'extraction présente un avantage par rapport aux règles d'identifications de [**?**] car elle ne s'appuie pas sur des situations décrivant des cas possibles dans une UI mais elle se base sur les types des éléments utilisés pour décrire l'UI et elle est plus générique. Cependant le choix de l'ontologie doit être exhaustive pour une API et pour les concepts qui sont représentés.

Mécanismes de transformations et de génération de l'UI

Une fois l'UI source décrite par les tâches d'interactions et ensuite raffinée en objets d'interactions abtraites, la représentation abstraite de l'UI de départ est transformée en modifiant manuellement le modèle d'UI.

Le **transformation** consiste d'une part à remplacer des objets d'interactions de l'UI source. Par exemple une tâche de sélections dans une liste qui est raffinée en une liste à choix unique peut être remplacée par un menu si le nombre d'éléments est inférieur à 10.

D'autre part cette phase fait intervenir un utilisateur humain pour définir la position, la taille ou pour modifier l'UI. 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 sélection des éléments de la bibliothèque graphique cible se fait en recherchant dans une bibliothèque graphique décrite par une ontologie des éléments qui correspondent le plus à un objet abstrait du modèle restructuré.

Le mapping entre objets abstraits et les éléments d'une bibliothèque graphique est fait **dynamiquement en se basant sur les ontologies**. En effet les correspondances entre les objets abstraits et les éléments de la bibliothèque graphique ne sont pas définies manuellement et de manière exhaustives à la conception de la solution, mais elles sont établies en se basant sur les attributs de chaque éléments.

Les mécanismes de transformations et générations utilisés par MORPH sont basés aussi sur des ontologies. Dans le cadre de la transformation par exemple il est possible d'introduire des guidelines dans la base de connaissances, en préférant remplacer une liste en menu si elle contient moins de 10 éléments par exemple.

La transformation utilisée par MORPH permet d'inclure les principes de conception d'UI pour la plateforme cible dans la base de connaissances [**?**].

2.3.1.3 Comment adapter MORPH pour la migration d'UI vers les tables interactives ?

La réponse à cette question passe d'abord par l'ajout de la bibliothèque graphique de la table interactive à la base de connaissances pour permettre l'extraction et la génération de l'UI finale. Ensuite, il faut décrire les modèles abstraits. Par exemple, 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 **??** :

- le menu principal a une tâche d'interactions de type SELECTION-OBJET avec les rôles (action = *Procedural-Action*, number-of-states = (2..10), variability = *fixed*, grouping= *not-grouped*)
- la liste déroulante (*ComboBox*) a une tâche d'interactions de type SELECTION-OBJET avec les rôles (action= *Procedural-Action*, number-of-states = (2..10), variability = *fixed*, grouping = *not-grouped*)
- la liste d'images a une tâche d'interactions de type SELECTION-OBJET avec les rôles (action = *Procedural-Action*, number-of-states = (2..10), variability = *fixed*, grouping = *not-grouped*)

Enfin, il faut mettre en place des règles de transformations des modèles abstraits en prenant en compte les guidelines. Par exemple, la prise en compte de la guideline **??** d'utilisation en 306° des éléments graphiques consiste à remplacer les composants graphiques contenant d'autres avec ceux qui ont ces interactions. Les modèles abstraits doivent être capables de décrire et de sélectionner les composants graphiques conformes à une guideline.

2.3.1.4 Résumé

Cette approche propose une solution de migration réutilisable qui prend en compte une phase reverse engineering (**détection**), une phase de re engineering (**transformation**) et une phase de forward engineering (**génération**). Le tableau 2.2 présente l'approche MORPH suivant les caractéristiques d'équivalences des éléments des plateformes, de modélisation et de prise en compte des guidelines.

L'approche est basée sur des modèles abstraits qui décrivent la structure et les interactions (et les comportements) des UI à migrer. Le layout et le style des UI ne sont pas modélisés par cette approche, leur migration est effectué manuellement par un utilisateur humain. Cependant les modèles abstraits proposés par MORPH ne sont pas adaptés pour être utilisés pour une plateforme comme une table interactive. Le modèle de tâche d'interactions par exemple ne prend pas en compte les mouvements (rotation, déplacement, etc) des composants graphiques des tables interactives. Le modèle de tâche

d'interactions permet d'établir des équivalences entre dispositifs d'interactions en se basant sur des tâches de haut niveau d'abstraction et indépendantes des plateformes. Cependant ce modèle n'est pas exhaustif.

En ce qui concerne les équivalences entre les bibliothèques graphiques par exemple, elles sont basées sur les modèles de connaissances et elles sont établies dynamiquement.

Les équivalences dynamiques entre les bibliothèques graphiques permettent de sélectionner les composants graphiques appropriés pour chaque cas en fonction de leurs caractéristiques d'utilisation.

Par ailleurs cette approche permet la prise en compte des guidelines à travers les transformations (remplacement des objets abstraits ou sélection des composants graphiques spécifiques). Cependant l'identification des guidelines et leur prise en compte pendant la transformation sont à décrire par le concepteur de la solution de migration sur une plateforme comme les tables interactives.

	Équivalences	**Modélisations**	**Prise en compte des guidelines**
MORPH	Dynamique des bibliothèques graphiques basées sur des modèles de connaissances	Modélisation des interactions abstraites, Modélisation de la structure	Prise en compte par les règles de transformations des modèles abstraits

TABLE 2.2 – Récapitulatif de MORPH

2.3.2 Approches de migration automatiques d'UI

La solution MORPH ne modélise pas tous les aspects d'une UI, ce qui implique une intervention de l'utilisateur pour la migrations des aspects non gérés. Dans cette section nous présentons les approches de migration réutilisables pour différentes plateformes en s'appuyant sur des modèles abstraits d'UI et qui ne font pas appel à des processus manuels.

Ces solutions de migration d'UI sont génériques et sont utilisées par des services de migration des UI dans des contextes ubiquitaires comportant plusieurs types de plateformes. Les services de migration d'UI sont chargés d'adapter une UI pendant son exécution [8] ou entre deux sessions de son utilisation [9] 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.

2.3.2.1 Modèles d'UI

Les modèles abstraits permettent de décrire les UI comportant en général plusieurs niveaux d'abstraction dans l'objectif de décrire les différents aspects des UI. Le framework de référence CAMELEON (CRF) [**?**] propose quatre niveaux d'abstraction pour décrire les UI : le niveau des tâches et concepts, le niveau interface abstraite (AUI), le niveau interface concrète (CUI) et le niveau interface finale (FUI).

8. C'est une migration à la volée d'une UI d'une application où l'état courant de l'UI est sauvegardé par des modèles abstraits d'UI et adapté sur une autre plateforme en permettant à l'utilisateur de continuer une tâche sans interruption [**?**]

9. C'est une migration qui implique que l'utilisateur quitte l'application, sauvegarde son état et redémarre l'application sur la nouvelle plateforme [**?**]

Modèle de tâches et concepts Ce modèle exprime les tâches et les concepts d'UI dans un contexte précis tel que défini par le concepteur. Ce modèle exprime les interactions de l'UI avec les utilisateurs ou le système, le comportement et les différents états des composants graphiques. Dans le cadre de la migration des UI à la volée, le modèle de tâches permet de conserver l'état d'une UI en cours d'exécution par exemple.

Dans notre cas, la migration ne se fait pas à l'exécution, mais elle se fait de manière statique et elle passe par une re conception de l'UI de départ. Le modèle de tâches peut permettre d'exprimer les interactions entre les utilisateurs et l'UI indépendamment des modalités d'interactions.

Modèle AUI Ce modèle permet de représenter les éléments d'une UI indépendamment des modalités d'interactions en exprimant leur fonctionnalité essentielle. Ce modèle concrétise les éléments du domaine utilisé par le modèle de tâches, par exemple une tâche de saisie de données correspond à un élément de type Input dans le modèle AUI (cf. figure 2.3). Les éléments du modèle AUI permettent en général d'exprimer les différents types d'interactions tels que l'entrée de données (Input), l'affichage d'informations (Output), l'activation d'une commande (Command). Ce modèle exprime aussi la structure d'une UI à travers les regroupements (Container) ou les types des données.

Dans notre cas, le modèle d'AUI exprime les interactions de haut niveau entre UI et NF et indépendamment des dispositifs d'interactions. Cependant ce modèle n'exprime pas les interactions sur les propriétés visuelles des composants graphiques (redimensionnement, déplacement, rotation, etc.). En effet le modèle AUI est indépendant de toute modalité, il peut être concrétisé en UI graphique, vocale, multimodale, etc.

Modèle CUI Ce modèle permet de décrire une représentation concrète de l'UI suivant une modalité. Dans le cadre des UI graphiques qui nous concernent par exemple, des composants graphiques sont choisies pour raffiner les éléments du modèle AUI. Ce modèle exprime la structure, le positionnement (layout) et même le style des éléments graphiques. Les composants graphiques de ce modèle sont exprimés dans un langage indépendant des bibliothèques graphiques pour garantir leur réutilisabilité.

Dans notre cas, le modèle de CUI exprime des UI de modalité graphique et l'on peut considérer la migration d'UI desktop vers une table interactive comme un changement de bibliothèque graphique qui nécessite une adaptation des interactions, de la structure, du layout et du style de l'UI de départ.

Il existe plusieurs implémentations pour chacun de ces niveaux de modèle. Paternò *et al.* [**?**] proposent le langage MARIA XML qui décrit les UI aux niveaux (AUI et CUI) et USIXML (User Interface eXtensible Markup Language) [**?**] décrit aussi des modèles de CUI, AUI.

FIGURE 2.3 – Exemple de transformation USIXML

Les différents modèles du CRF permettent de concevoir des UI multi plateformes, dans une approche de conception top down qui consiste à partir du modèle le plus abstrait (modèle de tâches et concepts), le concepteur génère les UI par raffinements successifs des modèles du CRF. Dans le cadre de la migration d'UI, ces modèles sont utilisés soit par des mécanismes de génération des UI finales pour la nouvelle plateforme à partir des modèles abstraits [**?**], soit par reverse engineering de l'UI existante [**?**].

2.3.2.2 Mécanismes de migration d'UI top down

Nous appelons les mécanismes de migration d'UI basés sur le raffinement des modèles abtraits du CRF en migration d'UI top down. En effet, les concepteurs génèrent par exemple le modèle CUI spécifique à chaque plateforme à partir des modèles AUI et tâches. Cette approche permet de migrer les états d'UI à l'exécution, MigriXML [**?**] par exemple définie grâce à ces mécanismes un environnement virtuel comprenant différentes plateformes (desktops avec différentes tailles d'écran, smartphones, etc.)

La génération des modèles les moins abstraits dans cette approche est effectuée à la conception. L'approche permet d'étendre une application à d'autres plateformes en réutilisant les modèles de tâches et d'AUI dans le cas d'une nouvelle plateforme ayant des modalités différentes de la plateforme de départ. Pour la migration d'une application existante par ces mécanismes, il est indispensable d'avoir les modèles du CRF associés. Cette contrainte exclue toutes les UI des applications qui ne sont pas conçues suivant le CRF.

2.3.2.3 Mécanismes de migration d'UI bottom up

Ces mécanismes permettent la migration d'UI en utilisant les modèles du CRF et en les abstrayant à partir d'UI existantes. Paternò *et al*. [**?**] proposent un service d'adaptation d'une page web à un téléphone portable 2.4 basé sur le reverse engineering et l'adaptation d'une UI existante. Ce service 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 ceux 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 redimenssionnant et en adoptant un layout vertical par exemple

Le concepteur définie des mécanismes de transformations [**?**] pour appliquer 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.

L'avantage de cette approche est que seuls les modèles indispensables pour la migration sont abstraits à partir de l'UI de départ. En effet, pour adaptation du layout par exemple, le modèle de tâche n'est pas indispensable. Dans le cadre de la migration d'UI vers les tables interactives qui implique un changement de dispositifs d'interactions, il est indispensable d'exprimer les interactions de manière abstraite par les modèles du CRF (Tâches et AUI). Cependant les modèles de tâches et AUI du CRF n'expriment pas toutes les interactions des tables interactives (cf. section 2.3.2.1). Ces mécanismes basées sur le reverse engineering nécessitent aussi que les applications respectent des architectures prônant la séparation entre UI et NF (telles que MVC ou ARCH).

FIGURE 2.4 – Service de migration d'UI

2.3.2.4 Service de migration d'UI desktop vers les tables interactives

Dans cette section nous étudions les adaptations nécessaires dans le cas d'une réutilisation du service de migration d'UI (cf. figure 2.4). Nous choisissons d'adapter cette solution car elle prend en compte des UI sources en retrouvant les modèles abstraits nécessaires. Nous considérons toujours notre exemple fil rouge :l'application CBA (cf. section **??**) à migrer sur une table interactive (Microsoft PixelSense). Une adaptation du service de migration d'UI nécessite :

- une table d'équivalences (tableau 2.3) entre les éléments du langage MARIA XML, XAML et l'API Java Swing pour abstraire et concrétiser l'UI de départ
- une formalisation des guidelines sur les éléments du langage MARIA XML.

TABLE 2.3 – Table d'équivalences

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

La table d'équivalence permet de décrire les correspondances entre les bibliothèques graphiques des plateformes sources et cibles. Le tableau 2.3 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.* [**?**] 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 et les balises des éléments graphiques.

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

2.3.2.5 Résumé

Cette section présente des mécanismes de migration d'UI basés sur les modèles de CRF. Le modèle de CUI graphique permet de décrire la structure hiérarchique, les données et le positionnement d'une UI. Dans un mécanisme de migration d'UI vers la table interactive, le modèle de CUI est transformé pour rendre l'UI de départ conforme aux guidelines.

Le modèle d'AUI décrit à la fois le regroupement des éléments abstraits d'UI et les interactions (en entrée ou en sortie) entre l'utilisateur et le système (NF). Les modèles de tâches et de concepts décrivent les activités d'UI et les comportements de l'UI de façon globale dans langage de haut niveau.

Les interactions décrites par les modèles de tâches et AUI expriment les interactions sur une UI indépendamment des dispositifs d'interactions, mais elles n'expriment pas l'ensemble des comportements des composants graphiques nécessaires pour établir des équivalences.

Équivalences	Modélisations	Prise en compte des guidelines
Statique des bibliothèques graphiques table d'équivalences	CUI : Structure et Layout AUI et Tâches : Interactions utilisateurs de haut niveau	Définie dans les règles de transformation des modèles

2.4 Synthèse et objectifs

2.4.1 Synthèse

Nous avons présenté dans le tableau 2.4 le récapitulatif des approches de migration d'UI étudiées suivant les critères d'évaluation décrits à la section 2.1. Ces approches de migration d'UI nous montrent que les solutions de migration peuvent être spécifiques à une application ou à une bibliothèque graphique. Et ces solutions peuvent aussi être réutilisables en se basant sur une modélisation des différents aspects d'une UI.

Nous avons raffiné les critères d'évaluation des approches présentées. Nous avons identifié deux types d'équivalences entre les éléments des plateformes :

- les **équivalences statiques** entre les bibliothèques graphiques 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 connaissances par exemple)

En ce qui concerne la prise en compte des guidelines, nous avons constaté que le concepteur peut se baser sur ses **connaissances** dans une approche manuelle pour décrire les mécanismes de transformations et d'équivalences. Pour des approches qui modélisent l'UI à migrer, les guidelines à considérer sont **traduites en règles** de transformation des différents aspects et les guidelines permettent aussi d'établir des équivalences entre les bibliothèques graphiques par exemple.

Les modélisations d'UI utilisées par les différentes approches présentées nous permettent d'affirmer que :

- les **interactions** dans un processus de migration d'UI peuvent être modélisées partiellement par les modèles de tâches et AUI [], les tâches d'interactions [**?**, **?**].
- la **structure d'une UI** comprend à la fois les données de l'UI et les relations hiérarchiques entre les différents composants de l'UI. Cet aspect de l'UI peut être modélisé par des **méta données** [**?**] ou des modèles d'UI [**?**, **?**] pour décrire les composants graphiques et les données d'UI.
- le **positionnement des éléments d'UI graphique** peut aussi être décrit dans un modèle indépendant d'une plateforme. Les langages de description d'UI (UIDL) tels que USIXML, MARIA, UIML permettent de décrire le layout.
- le **style** d'UI peut être modéliser au travers d'UIDL comme UIML [10].

2.4.1.1 Les points forts des approches présentées

- Les solutions de migration d'UI [**?**, **?**] faisant intervenir les utilisateurs humains permettent de générer des UI migrées proche de l'attente des utilisateurs finaux. En effet ces approches permettent une re conception manuelle pendant la migration et les prises en compte des guidelines sont plus fines pour les utilisateur ayant une bonne connaissance des plateformes cibles.
- Les solutions de migration basées sur les modèles permettent de décrire des mécanismes de transformations et d'équivalences réutilisables pour des applications respectant une architecture

10. UIML : An Appliance-Independent XML User Interface Language

TABLE 2.4 – Synthèse des approches de migration d'UI

Approches	Équivalences	Prise en compte des guidelines	Modèles
Approches ad-hoc	Équivalences statiques : – des bibliothèques graphiques, – des dispositifs d'interactions	Basée sur les connaissances du concepteur	Aucun modèle d'UI
Portage d'UI sur table interactive	Équivalences statiques : basées sur les boîtes à outils graphiques et définies par le concepteur	Toutes les guidelines ne peuvent pas être prises en compte	Modèle de structure d'UI (méta données)
MORPH	Dynamique : basée sur un modèle de connaissances et exprimée par des rôles	Règles de transformations basées sur les modèles abstraits	Modèle de structure d'UI & Modèle d'interactions abstraites
Service de migration d'UI	Statique : tables d'équivalences statiques et mapping des modèles	Règles de transformations des modèles	Tâches, AUI et CUI

définie.

2.4.1.2 Les limites

La réutilisation des approches basées sur des modèles d'UI dans le cadre des tables interactives nous permet d'identifier les limites ci-dessous :

- les **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
- la **prise en compte des guidelines** par toutes les approches décrites.

2.4.2 Objectifs

L'objectif principal de cette thèse est de faire de la migration des UI existantes en prenant en compte les guidelines de plateforme d'arrivée. Pour atteindre cet objectif nous choisissons de passer par le reverse engineering des UI d'abord en l'abstrayant dans des modèles abstraits. Ensuite, notre solution transforme les modèles abstraits tout en faisant intervenir le concepteur pour personnaliser les aspects non modélisés. Et enfin nous générerons l'UI finale à partir des modèles abstraits transformés.

La solution de migration d'UI que nous proposons se base aussi sur des applications décrivant le modèle d'architecture MVC.

En considérant cet objectif principal, l'étude des différentes approches de migration d'UI de façon globale et aussi particulièrement la migration sur les tables interactives nous montre des modèles d'interactions qui ne facilitent pas les équivalences entre les plateformes. Pour atteindre notre objectif, notre contribution dans cette thèse est de :

- proposer un modèle d'interactions qui permet le changement de modalité d'interactions et la préservation des interactions de l'UI de départ tout en prenant en compte celles de la plateforme d'arrivée
- 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

CHAPITRE 4

Prise en compte des guidelines

CHAPITRE 5

Prototype

Troisième partie

Conclusion et Perspectives

CHAPITRE 6

Conclusion

Table des figures

2.1 Application de consultation des contacts 7
2.2 Processus de migration avec MORPH 11
2.3 Exemple de transformation USIXML 15
2.4 Service de migration d'UI 16

Liste des tableaux

2.1 Synthèse des approches de portage d'UI sur tables interactives 10
2.2 Récapitulatif de MORPH 14
2.3 Table d'équivalences 17
2.4 Synthèse des approches de migration d'UI 19