UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Yoann GABILLON

Thèse dirigée par **Gaëlle CALVARY**
et par **Humbert FIORINO**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
et de **École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Composition d'Interfaces Homme-Machine par planification automatique

Thèse soutenue publiquement le **14 octobre 2011**,
devant le jury composé de :

Mme Gaëlle CALVARY
Professeur, Grenoble INP, Directeur de thèse
M. Humbert FIORINO
Maître de conférences, UJF, Co-Directeur de thèse
M. Christophe KOLSKI
Professeur, Université de Valenciennes, Rapporteur
M. Jean VANDERDONCKT
Professeur, Université catholique de Louvain, Rapporteur
Mme Anne-Marie PINNA-DERY
Maître de conférences, École Polytechnique Universitaire de Sophia-Antipolis, Examinateur
M. Laurent D'ORAZIO
Maître de conférences, Université Blaise Pascal, Examinateur
M. Yves LEDRU
Professeur, UJF, Président

À mes parents et à Lucie.

Remerciements

Ce travail de thèse n'aurait pu voir le jour, sans le soutien de nombreuses personnes. Ces litotiques remerciements ne pourront exprimer l'importance de leur soutien.

Je souhaite tout d'abord remercier mes directeurs de thèse : **Gaëlle Calvary** et **Humbert Fiorino**. Vous avez tous deux apporté deux points de vue distincts indispensables à la réalisation de ce travail. Humbert, merci de m'avoir apporté formalisme ainsi qu'un recul sur l'IA et la recherche en général très enrichissant. Gaëlle, merci infiniment de m'avoir soutenu tout le long de la thèse. Grâce à ta compétence, ta patience et à tes qualités humaines, j'ai énormément appris durant cette thèse. Il n'y a pas de mots pour exprimer ma gratitude pour tout ce que tu as fait pour moi.

Je remercie particulièrement **Jean Vanderdonckt** et **Christophe Kolski** pour l'intérêt que vous voulez bien porter à mes travaux, illustré par l'honneur que vous m'avez fait d'accepter d'être rapporteurs de cette thèse, mais également par les échanges que nous avons eus, que nous avons, et que nous aurons. Merci également aux membres du jury qui ont porté attention à mon travail et accepté de rapporter ma thèse : **Anne-Marie Pinna-Dery**, **Yves Ledru** et **Laurent D'Orazio**. Ma soutenance de thèse restera un excellent souvenir et nourrira mes réflexions lors la suite de ma carrière.

Je remercie tout particulièrement **Alexandre Demeure** qui a réalisé la boite à outils COMETs et qui a participé à la réalisation du prototype et à mes réflexions sur la composition. **Nadine Mandran** a également été d'une aide indispensable pour mener l'étude sociologique si instructive et enrichissante. Le protocole, comme l'analyse des résultats, n'auraient pu être de cette qualité sans ses compétences en sociologie et en statistiques, ni sans sa gentillesse. Je remercie également **Sofia Zaidenberg** pour son apport lors de cette étude.

Je tiens également à remercier l'**équipe IIHM** et l'**équipe MAGMA** dans leur ensemble. En particulier, merci aux chefs d'équipes **Joëlle Coutaz** puis **Laurence Nigay** de m'avoir accueilli dans l'équipe IIHM, et

Yves Demazeau, puis **Julie Dugdade** et **Sylvie Pesty** de m'avoir accueilli dans l'équipe MAGMA. J'ai eu la chance d'évoluer dans deux équipes qui m'ont beaucoup apporté. Pour ne pas tous les citer, je remercie également : **Sophie Dupuis-Chessa**, **François Berard**, **Renaud Blanch**, **Celine Coutrix**, **Yann Laurillau**, **Frank Tarpin-Bernard**, **Eric Ceret**, **Alphonso Garcia**, **Jérémy Francone**, **Mickael Ortega**, **Jean Sébastien Sottet**, **Rémi Dupuis**, **Dimitri Masson**, **Thomas Vincent**, **Lionel Balme**, **Frédéric Jourdre**, **Adriano Scoditti**, **Audrey Serna**, **Emeric Fontaine**, de l'équipe IIHM ainsi que **Chatherine Garbay**, **Cyrille Martin**, **Carole Adam**, **Fabien Badeig**, **Jérémie Rivière**, **Laurent Lacaume**, **Robin Lamarche-Perrin**, sans oublier **Axelle** de l'équipe MAGMA, **Pauline**, **Aurélia**. J'ai pu y confirmer l'importance d'une équipe de recherche, de l'ambiance qui peut y régner et de l'impact que cela peut avoir sur le travail et le bien être de chacun.

Je remercie le **CLUSTER ISLE** qui à financé mes travaux de thèse à travers le **projet PRESENCE**.

La formation à la recherche est un long processus. La thèse représente l'aboutissement officiel de ce processus de formation mais je tiens particulièrement à remercier les personnes qui m'ont permis d'atteindre la thèse. Ma formation à la recherche a débuté en Master grâce à **Violaine Prince** que je remercie d'avoir accepté de me guider en stage volontaire en TALN. Il m'a permis de découvrir le métier de chercheur et de mettre en application ce désir que j'avais depuis plusieurs années. Dans la même période, je veux remercier **Jocelyne Nanard** et **Marc Nanard** pour les riches discussions et leur enseignement en IHM. C'est grâce à eux que j'ai découvert ce domaine. Je ne peux évoquer l'IHM sans parler de l'IA puisque mes attraits ont toujours porté sur les deux disciplines. Merci à **Jacques Ferber** et **Stéphano Cerri** qui ont été mes premiers inspirateurs en SMA. Le stage de M2R est le véritable pied à l'étrier. Merci à **Vincent Chapurlat** et **Daniel Diepp** pour m'avoir encadré et permis d'être prêt pour débuter ma thèse. J'ai beaucoup appris grâce à toutes ces personnes, et je n'oublie pas qu'elles ont participé à ma formation et que j'ai grandi grâce à eux.

Pour finir, je tiens tout particulièrement à remercier mes parents **Fabien** et **Yolande** qui m'ont soutenu durant toutes mes longues études. Ils ont fait preuve de patience et d'encouragement constants durant ces années. Je tiens également à remercier de tout cœur **Lucie** de m'avoir soutenu tout au long de cette thèse et en particulier lors de la rédaction en étant une relectrice assidue. Merci à **Clovis** d'être venu à ma soutenance.

Table des matières

Liste des figures

Liste des Algorithmes

Introduction 1

Publications associées : IHM'09 (Gabillon et al. 2009), RJCIHM'08 (Gabillon 2008)

1.1 Contexte des recherches

Mes travaux de recherche s'inscrivent dans le cadre de l'informatique ambiante telle que définie par Weiser (Weiser 1991; 1993) :

> *« Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. »*

Par opposition à l'informatique traditionnelle, souvent cantonnée à un ordinateur muni d'un clavier et d'une souris, l'informatique ambiante s'ouvre sur une diversité de dispositifs et d'usages (Weiser 1993) :

> *« I first created the idea of ubiquitous computing from contemplation of the place of today's computer in actual activities of everyday life. »*

La diversité concerne deux volets : le contexte d'usage et la tâche de l'utilisateur.

– Le **contexte d'usage** est défini selon trois dimensions (Calvary et al. 2003) :

  - L'*utilisateur* désigne l'utilisateur final d'un système interactif. Avec la diminution du coût des plates-formes, il provient d'un public large. Il est en général modélisé par ses capacités sensorielles, cognitives et motrices (Card et al. 1983).

  - La *plate-forme* est l'infrastructure matérielle et logicielle sous-tendant l'interaction. L'ordinateur portable est un exemple. Les plates-formes se sont diversifiées depuis une vingtaine d'années avec la diminution de leur coût. L'informatique est intégrée dans de nombreux appareils du quotidien : téléphone portable,

télévision, GPS, électroménager, etc. Parallèlement, leurs caractéristiques techniques (écrans tactiles, synthétiseurs vocaux, processeurs, etc.) ne cessent d'évoluer.

- L'*environnement* désigne l'environnement physique et social dans lequel prend place l'interaction. L'utilisateur n'est plus confiné à un lieu fixe (bureau, maison, etc.) ou à un environnement social particulier (seul, avec des collègues, avec des amis, etc.). Il interagit désormais dans la rue, dans les transports en commun, lors de fêtes familiales, de réunions de bureau, etc.

– La **tâche de l'utilisateur** est l'objectif que l'utilisateur doit accomplir, assorti d'une procédure. Par exemple, pour « prendre un rendez-vous », l'utilisateur doit ouvrir son agenda puis noter l'objet du rendez-vous, sa date, son heure, son lieu et ses participants. Avec l'informatique ambiante, les tâches de l'utilisateur ne se limitent plus à des tâches professionnelles. Elles se diversifient. L'utilisateur peut désormais accomplir des tâches quotidiennes comme « éteindre le chauffage le soir ».

En informatique ambiante, le contexte d'usage et la tâche de l'utilisateur peuvent non seulement être variables mais aussi imprévisibles. L'utilisateur étant mobile, il évolue dans un contexte d'usage dynamique propice à des objectifs émergents. De son Smartphone, l'utilisateur peut accéder à un bouquet de services (par exemple, les applications iPhone). L'offre est évolutive, fonction de la localisation de l'utilisateur. Il peut accomplir une variété de tâches de différentes manières. Par exemple, pour consulter son courrier électronique, il peut utiliser son webmail ou son client de messagerie.

En ingénierie de l'interaction Homme-Machine, la variabilité du contexte d'usage est étudiée pour des contextes d'usage prévus à la conception (Thevenin et Coutaz 1999, Thevenin 2001). La tâche de l'utilisateur est toujours supposée connue à la conception et constante à l'exécution. Cette hypothèse est remise en cause aujourd'hui par l'informatique ambiante. Ce constat motive la problématique de cette thèse : la composition dynamique de systèmes interactifs.

1.2 Sujet et cas d'étude

Un système interactif est constitué d'un noyau fonctionnel (NF) et d'une Interface Homme-Machine (IHM) (Figure 1.1). Le NF regroupe l'ensemble des traitements indépendamment de toute représentation à

l'utilisateur. L'IHM fait les choix de présentation compte tenu du contexte d'usage courant et des propriétés ergonomiques à satisfaire. Ma **thèse** traite de la composition d'IHM. Le noyau fonctionnel n'est pas considéré.

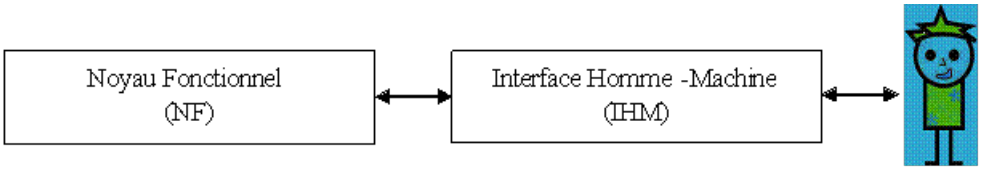


FIGURE 1.1 – *Décomposition fonctionnelle minimale d'un système interactif.*

Pour illustrer le sujet, considérons un **cas d'étude** « See a doctor ». Le cas d'étude met en scène Victor, un jeune étudiant qui vit à Grenoble. Il est en vacances dans un hôtel à Montpellier. Il est 23 heures. Soudain, Victor ne se sent pas bien. Il ressent le besoin d'une assistance médicale. Il est déjà tard. Il ne connaît personne dans cette ville. Il lance *Compose*, un logiciel de type « assistant personnel ». Après s'être identifié, il spécifie en langage naturel son objectif « Je veux aller voir un médecin » (Figure 1.2).

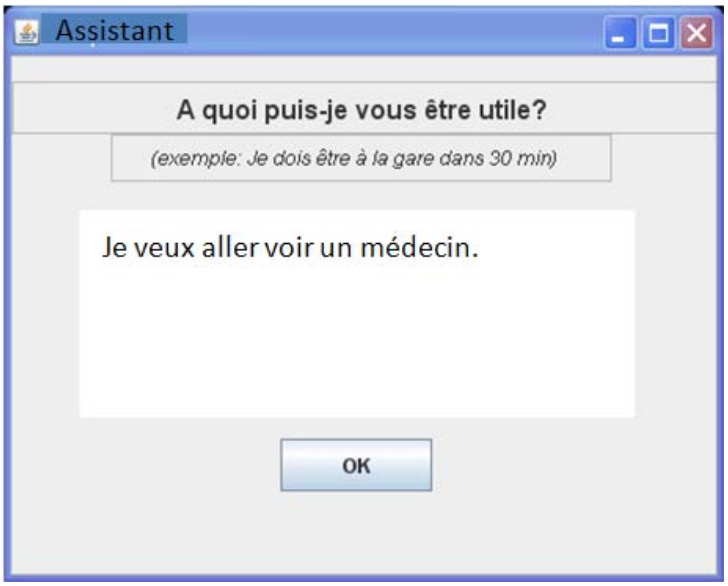
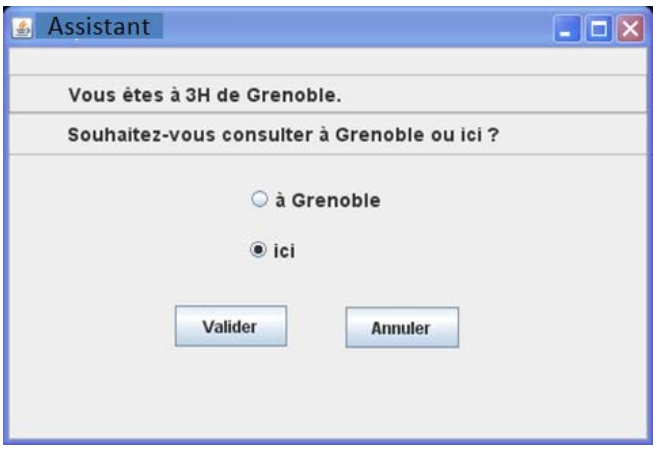


FIGURE 1.2 – *L'IHM pour spécifier l'objectif de l'utilisateur.*

Compose calcule à la volée des solutions médicales adaptées au contexte d'usage de Victor : il peut soit rentrer chez lui à Grenoble pour se faire soigner, soit rester sur place (Figure 1.3a). Victor préfère rester à Montpellier. *Compose* lui propose alors cinq possibilités (Figure 1.3b) : consulter le médecin de garde (Dr. Mabuse, à 10 minutes en voiture), appeler le SAMU, se rendre à l'hôpital le plus proche (Hôpital Dapi, à 35 minutes en voiture), appeler SOS médecin ou les pompiers. Victor

choisit de voir le médecin de garde.



a) Deux possibilités de lieu pour se soigner.

b) Cinq solutions d'assistance médicale.

FIGURE 1.3 – *Deux IHM composées en contexte pour affiner le besoin de l'utilisateur « voir un médecin ».*

Trois variantes d'IHM sont ensuite possibles selon que Victor utilise un mur numérique (version 1), un Smarthphone (version 2) ou un mur numérique couplé à un Smartphone sans GPS (version 3).

Version 1 : Mur numérique

Compose calcule à la volée une IHM (Figure 1.4) pour le mur numérique. Elle permet à Victor d'appeler le docteur Mabuse. Le numéro est pré-composé en haut de la fenêtre. L'IHM localise également le cabinet médical (sur la carte au milieu de la fenêtre) et fournit des informations complémentaires : ici, la pharmacie de garde et la station essence la plus proche (en bas de la fenêtre).

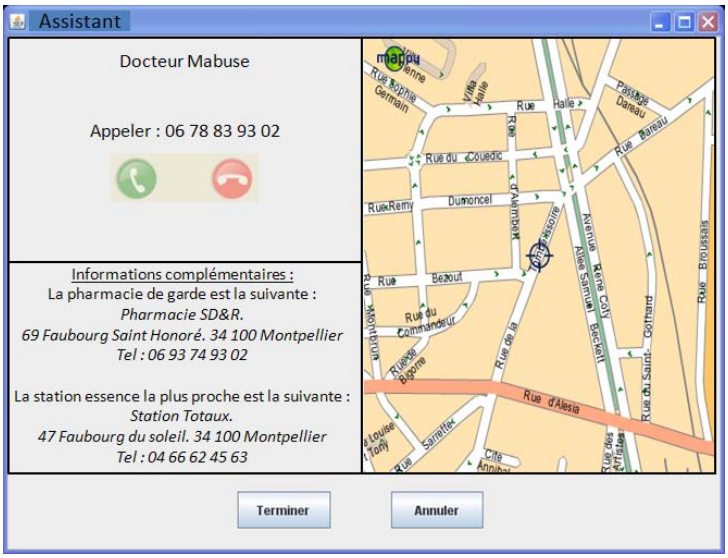
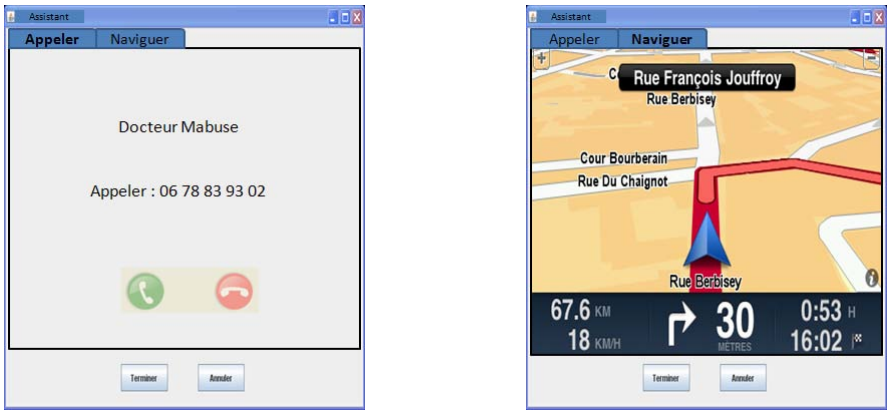


FIGURE 1.4 – *IHM composée pour un mur numérique.*

L'IHM produite par *Compose* est différente si Victor dispose seulement d'un SmartPhone.

Version 2 : Smartphone

Compose calcule à la volée une IHM (Figure 1.5) pour le Smartphone. Elle permet à Victor d'appeler le docteur Mabuse. L'IHM guide Victor vers le cabinet médical en utilisant un navigateur GPS. Dans ce cas, l'assistant présente ces deux parties de l'IHM dans un onglet et omet les informations complémentaires compte tenu de la petite taille de l'écran.

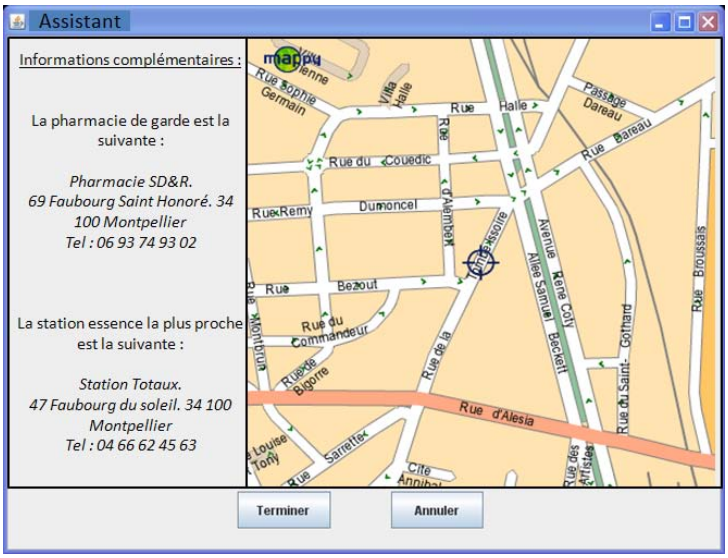


a) Onglet « Appeler ». *b) Onglet « Naviguer ».*

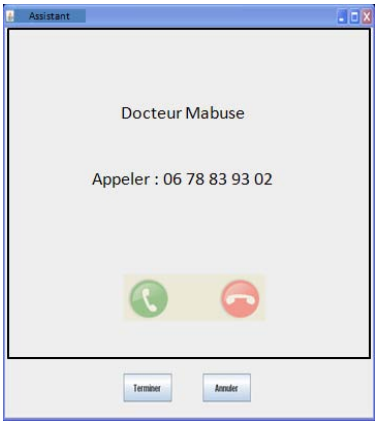
FIGURE 1.5 – *IHM composée pour un Smartphone.*

Version 3 : Mur numérique et Smarthphone sans GPS

L'assistant calcule à la volée une IHM (Figure 1.6) pour le mur numérique couplé à un Smartphone sans GPS. Elle permet à Victor d'appeler le docteur Mabuse sur le Smartphone. L'IHM localise également le cabinet médical sur une carte et fournit des informations complémentaires sur le mur numérique.



a) IHM composée pour un mur numérique.



b) IHM composée pour un Smartphone sans GPS.

FIGURE 1.6 – *IHM composée pour un mur et un Smartphone sans GPS.*

Mon **sujet** est l'étude de systèmes tels que Compose. Il s'agit de composer dynamiquement une IHM compte tenu d'un objectif utilisateur donné et des ressources disponibles dans le contexte d'usage courant (services, dispositifs d'interaction).

1.3 Difficultés

La composition dynamique d'IHM est un sujet difficile à plusieurs titres :

- **Sémantique :** la composition d'IHM n'est pas un problème de « surface ». Elle requiert de connaître la raison d'être de chaque élément de l'IHM : sa sémantique. Reprenons l'exemple des deux IHM comportant chacune une date et un bouton de validation. La composition de ces deux IHM en une seule fenêtre soulève deux problèmes :
  - les dates : représentent-elles le même concept ? Est-ce, par exemple, la date du jour ? Ou représentent-elles des concepts différents comme la date de naissance de l'individu et sa date d'aménagement ? Si ces dates représentent le même concept, comme sa date de naissance, alors pour minimiser la charge de travail de l'utilisateur, la date ne devrait être affichée qu'une seule fois dans l'IHM composée ;
  - les boutons de validation : correspondent-ils à la même tâche ? Si oui, ces boutons devraient être factorisés pour permettre à l'utilisateur de valider l'ensemble du formulaire en une fois.

  Les exemples peuvent être multipliés. Typiquement si deux IHM nécessitaient l'authentification de l'utilisateur, alors au regard de ce même critère de charge de travail, il serait maladroit de demander à l'utilisateur de s'authentifier deux fois dans l'IHM composée. Ainsi, composer une IHM ne se limite pas à une juxtaposition mécanique des IHM à assembler. Des ajustements peuvent être nécessaires comme la suppression de concepts et de tâches.

- **Combinatoire :** il y a plusieurs façons de composer deux IHM. Par exemple, dans la figure 1.7, les IHM permettant d'appeler un médecin et de téléphoner sont composées en séquence (c) ou en parallélisme (a). En (a), les IHM sont placées dans deux canevas différents et peuvent, en conséquence, être éventuellement exécutées sur deux plates-formes différentes. Par exemple, Victor peut appeler le médecin de son Smartphone pendant qu'il consulte l'itinéraire sur le mur numérique. Au contraire, en (d) et (e), elles sont assemblées en un même canevas. En (d), les IHM sont simplement juxtaposées verticalement. En (e), les éléments redondants (la date du jour et les boutons de navigation « Validate » et « Cancel ») ont été fusionnés pour réduire la charge de Travail de l'utilisateur. Ainsi est-il important que le système de composition sache, dans la combinatoire, sélectionner la « bonne » manière de composer pour produire une IHM de qualité. Cette observabilité est valable

pour le résultat de la composition mais aussi pour le processus de composition. Aussi toutes les propriétés classiques en IHM (prévisibilité, en particulier prévisibilité du calcul) s'appliquent-elles.

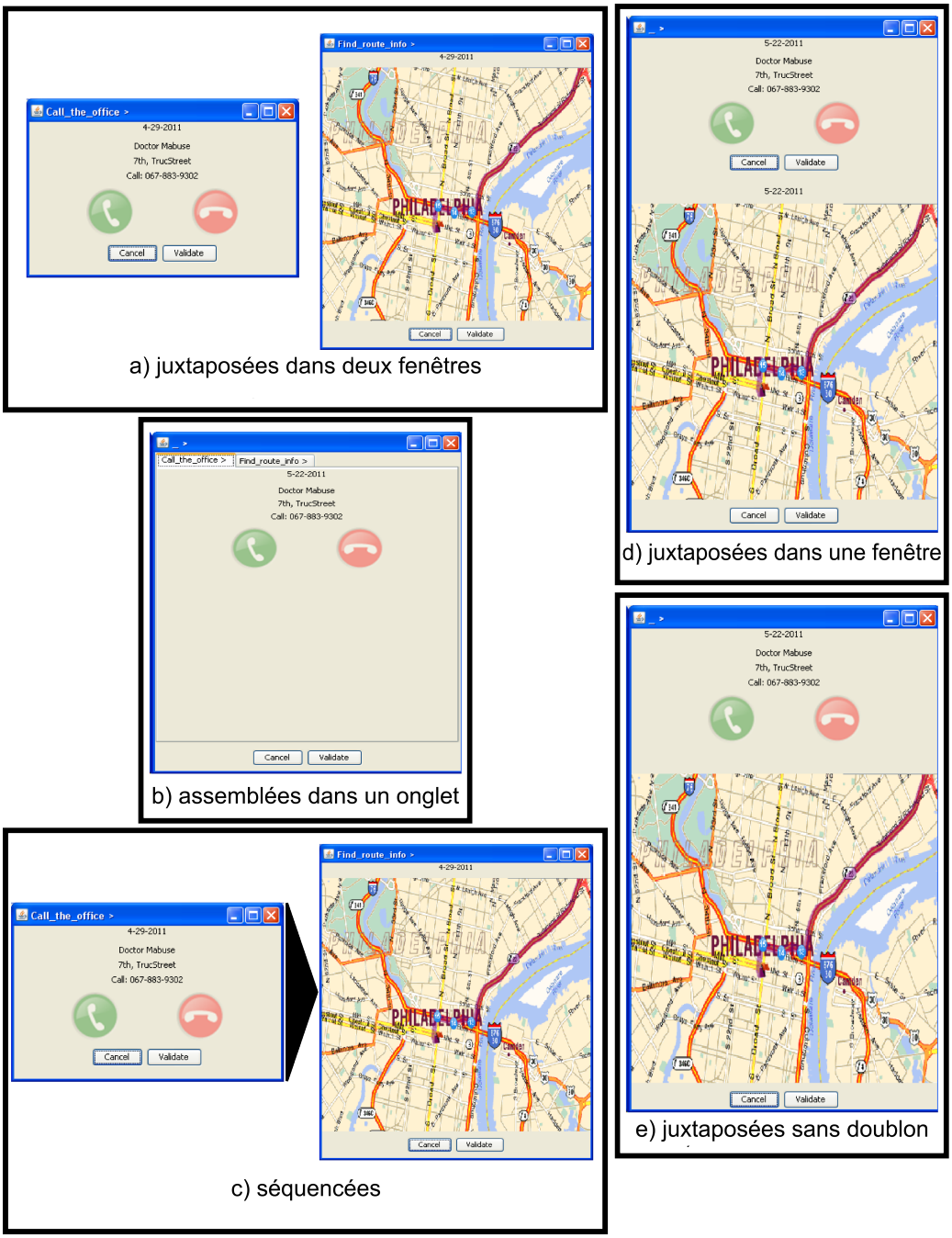


FIGURE 1.7 – *Exemples de compositions.*

– **Observabilité :** la qualité de la composition est « visible » à l'utilisateur. L'expertise du concepteur n'est plus dans la boucle de validation faisant, en conséquence, reposer la « qualité » de l'IHM

composée sur les performances du système de composition.

– **Sélection :** une façon de régler le problème de combinatoire et de satisfaire l'exigence de qualité est de mettre l'utilisateur dans la boucle via une méta-IHM de composition. Cette méta-IHM est chargée de rendre le processus de composition observable et contrôlable par l'utilisateur final (Coutaz 2006). Des choix de composition peuvent ainsi être laissés à l'utilisateur. Par exemple, Victor aurait pu choisir entre le GPS et la carte manuellement. Il aurait pu aussi choisir la manière de composer son IHM. Par exemple, Victor peut préférer composer horizontalement que verticalement.

– **Validité :** une IHM pour être utilisable requiert des ressources d'interaction (classiquement un écran, un clavier et une souris). Ces ressources peuvent être partagées et remettre en cause la validité de l'IHM en cas d'accès concurrent (typiquement le canal audio). Plus généralement, la non disponibilité d'un service ou d'une ressource peut rendre l'IHM inutilisable.

1.4 Objectifs, approche et démarche

Mes **objectifs** sont doubles :
– d'un point de vue conceptuel, il s'agit de comprendre les manières de composer dynamiquement des IHM et d'établir des patrons de composition.
– d'un point de vue logiciel, il s'agit d'appliquer et d'adapter les algorithmes de planification automatique à la composition d'IHM. Ces algorithmes doivent « raisonner » sur le contexte d'usage et sur la qualité de l'IHM composée pour permettre une composition dynamique.

Ainsi, mon **approche** s'inscrit dans l'Intelligence Artificielle (IA) sous l'angle de la planification automatique. Il s'agit d'explorer la pertinence de cette approche pour la composition dynamique d'IHM.

Ma **démarche** est analytique. Elle se structure en trois temps. Dans un premier temps, je décompose le problème de la composition dynamique d'IHM pour séparer les préoccupations et ainsi simplifier le problème. Selon cet espace problème, l'état de l'art en composition d'IHM est dressé et analysé dans un deuxième temps. Cette analyse souligne l'absence de travaux en IHM considérant la tâche de l'utilisateur variable et permettant ainsi la dynamicité de la composition à haut niveau d'abstraction. En conséquence, dans un troisième temps, je propose de

composer dynamiquement des IHM à partir de l'objectif de l'utilisateur. Cette composition nécessite de « raisonner » sur la composition d'IHM pour sélectionner les IHM à composer selon le contexte d'usage et des propriétés ergonomiques à privilégier.

1.5 Contributions

Mes **contributions** sont de deux types : conceptuel et logiciel.

Mes contributions conceptuelles sont :
- l'identification d'exigences utilisateur vis-à-vis d'un système tel que *Compose* ;
- l'identification d'exigences système suivant une décomposition fonctionnelle ;
- un espace problème de la composition d'IHM.

Mes contributions logicielles sont :
- la confrontation des algorithmes de planification au problème de composition d'IHM ;
- une adaptation de ces algorithmes pour calculer le modèle de tâches de l'IHM composée. Notre travail se concentre sur la recherche d'une solution au niveau tâches.

J'illustre mes contributions avec le démonstrateur *Compose* : mon système de composition d'IHM.

1.6 Plan du manuscrit

Le **plan** de cette thèse se structure en deux parties divisées en six chapitres (figure 1.8) :

- Dans la première partie, j'analyse le problème et l'existant en composition d'IHM :

  - Au chapitre 2, j'étudie la composition d'IHM du point de vue de l'utilisateur. Cette étude menée avec une sociologue confirme d'une part, la pertinence du sujet et permet, d'autre part, de recueillir des exigences et/ou préférences utilisateur de nature à orienter le développement du système de composition.

- Au chapitre 3, je présente les concepts fondamentaux en IHM sur lesquels repose ce travail.

- Au chapitre 4, je dresse l'espace problème de la composition dynamique d'IHM. En particulier, je propose une définition des concepts de la composition, une taxonomie des opérateurs de composition et un espace problème du processus de composition d'IHM.

- Au chapitre 5, j'analyse l'état de l'art en composition de noyau fonctionnel et d'IHM. Cette analyse s'appuie sur l'espace problème de la composition proposé au chapitre 4. Elle met en avant les manques de la littérature pour traiter l'imprévisibilité de la tâche de l'utilisateur : composer dynamiquement des IHM.

– Dans la deuxième partie, j'expose ma solution par planification :

- Au chapitre 6, je présente mes contributions pour traiter cette imprévisibilité par des algorithmes de planification. Ce chapitre s'organise en trois parties. Dans la première, je présente les concepts fondamentaux en planification automatique. Dans la deuxième, j'étudie la composition d'IHM par des algorithmes de planification hiérarchique (Hierarchical Task Network (Erol 1995), HTN) existants. Cette étude m'amène à formaliser le problème d'IHM en un problème de planification et à proposer une procédure pour résoudre ce problème dans la troisième partie.

- Au chapitre 7, je décris *Compose*, un prototype illustrant l'utilisation de la planification automatique pour la composition d'IHM. Ce prototype inclut le *planificateur Compose* adapté à la composition d'IHM.

Je termine par un rappel des contributions et l'ouverture de nombreuses perspectives.

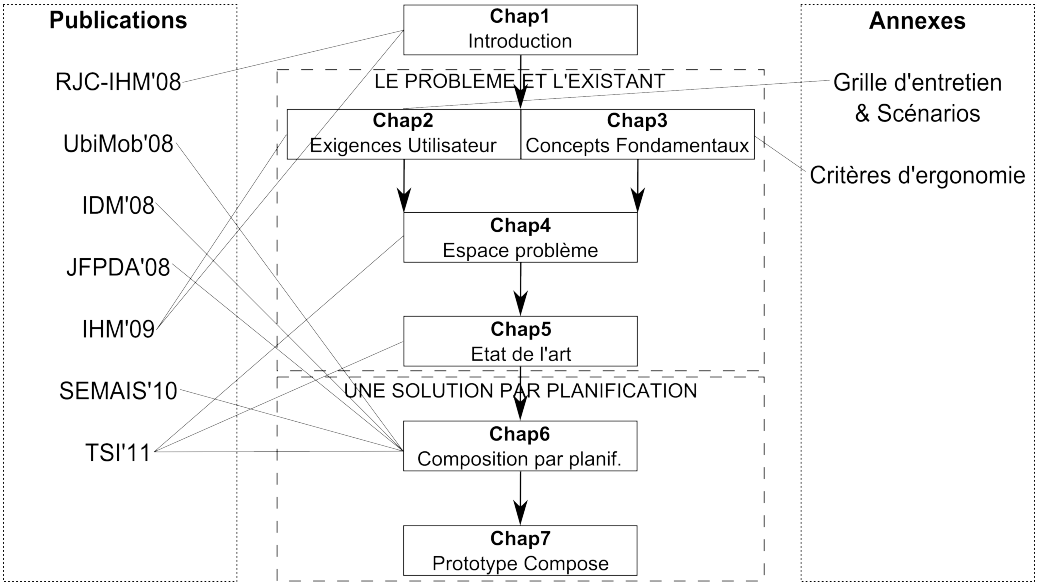


FIGURE 1.8 – *Plan du manuscrit.*

Première partie

LE PROBLÈME ET L'EXISTANT

Exigences utilisateur

2

Résumé

Ce chapitre traite de la composition d'IHM du point de vue de l'utilisateur. L'étude de terrain a été menée avec une sociologue. Elle confirme la pertinence du sujet et permet de recueillir des exigences et/ou préférences utilisateur de nature à orienter le développement du système de composition. La capitalisation des IHM composées est particulièrement pointée comme importante par la stabilité du système. Le mode d'interaction préféré est le langage naturel. *Compose* est plébiscité dans des situations d'urgence et au quotidien.

Publication associée : IHM'09 (Gabillon et al. 2009)

Sommaire

Introduction

Ce chapitre traite du système de composition d'IHM *Compose* du point de vue de l'utilisateur. Il présente l'étude terrain menée avec une sociologue pour d'une part, mesurer la pertinence du sujet et d'autre part, recueillir des exigences et/ou préférences utilisateur de nature à orienter nos développements logiciels.

L'étude a porté sur un public large en termes d'âge, de sexe, de situation socioprofessionnelle et de type d'habitat dans le but de couvrir tous les types d'habitudes en informatique : des novices aux experts. Le protocole s'articule en deux phases : une enquête qualitative a été menée dans un premier temps pour identifier les profils utilisateurs les plus intéressés et faire émerger des situations clé pour *Compose*. Des groupes de discussion ont ensuite été animés pour identifier les modes d'interaction favoris et recueillir des exigences complémentaires telles que les fonctions appréciées ou la qualité attendue. Pour chaque étude, la suite décrit les protocoles, puis les enseignements.

2.1 Entretien

L'entretien semi directif est une méthode qualitative (Dey et al. 2006, Hindus et al. 2001) qui permet des discussions ouvertes et libres. Il favorise l'émergence d'idées, opinions ou habitudes indépendamment de leur fréquence d'apparition. Le but n'est pas d'énumérer de manière exhaustive l'ensemble des comportements ou des besoins des usagers mais de mettre en évidence un ensemble d'idées. L'intérêt est de permettre aux sujets de s'exprimer librement tout en étant canalisés par des questions. La sociologie recommande 20 entretiens au moins. La pratique montre qu'au delà de 20 entretiens, les idées originales deviennent rares.

2.1.1 Protocole

Les interviews ont été menées auprès de 26 personnes de profils différents :

- Age : 9 personnes de 18 à 25 ans ; 7 personnes de 26 à 40 ans ; 7 personnes de 40 à 60 ans ; 3 personnes de plus de 60 ans.
- Sexe : 12 femmes ; 14 hommes.
- Statut professionnel : 14 actifs ; 9 inactifs ; 3 retraités.
- Catégorie professionnelle : 1 artiste ; 8 professions supérieures ; 4

professions intermédiaires ; 2 employés ; 2 ouvriers ; 9 étudiants et inactifs.
– Zone d'habitation : 3 rurales ; 5 rurbaines ; 18 urbaines.

En moyenne, les interviews ont duré une heure par personne. Elles commençaient par mesurer le niveau de connaissance du participant vis-à-vis des nouvelles technologies, ses habitudes vis-à-vis de l'informatique, d'Internet et de l'informatique ambiante. Selon la technique de l'entonnoir, les discussions se resserraient ensuite progressivement vers *Compose*. En particulier, les participants étaient invités à se remémorer des situations délicates pour lesquelles ils auraient apprécié l'aide d'un assistant. Un diaporama (Figures 1.3 et 1.4) du cas d'étude était finalement présenté pour concrétiser le concept d'assistant. Les participants s'exprimaient sur la pertinence du système, les fonctions attendues, les modes d'interaction favoris ainsi que la qualité attendue.

La grille d'entretien complète est présentée en annexe A.1.

2.1.2 Résultats

L'étude révèle qu'un système tel que *Compose* est globalement bien apprécié quel que soit le profil des utilisateurs (23 sujets sur 26 sont positifs à l'égard de *Compose*). Le système s'avère utile aussi bien pour les novices que les experts en informatique. Il n'existe pas de différence sensible entre ces deux profils. Toutefois, on peut noter que les moins aguerris en informatique attendraient volontiers de *Compose* une simplification de la vie au quotidien : une aide en informatique par exemple.

Le cas d'étude est apprécié par la majorité des utilisateurs :

> *« C'est le bazar pour trouver un médecin de garde. C'est bien, si ça marche ».*

D'autres cas d'étude sont suggérés comme la panne d'essence ou un problème à l'étranger. Cependant, contrairement à notre attente, *Compose* est également plébiscité pour des situations banales sans caractère d'urgence : démarches administratives, préparation des loisirs, aide technique, etc.

> *« J'utilise mal un logiciel. J'aurais une aide pour me guider. [...] Mon disque dur est saturé. Il va me suggérer d'éliminer tel fichier. [...] Un fichier son ou vidéo est trop volumineux. Je ne sais pas comment le compresser. Je ne sais pas juger de la qualité. Il faudrait qu'il prenne en jeu les paramètres, qu'il repère que j'ai des logiciels de compression, qu'il m'ouvre directement le logiciel et qu'il me pose les bonnes questions. Ensuite, je ferai mon choix ».*

L'étude montre que les utilisateurs préfèrent le mode d'interaction écrit (24 sujets sur 26). Seulement quelques uns sont favorables à l'oral ou à un mode de pré-interrogation.

On apprend par ailleurs que les sujets seraient moyennement tolérants aux erreurs d'interprétation ou incapacités à répondre. Les assistants personnels ne font pas l'unanimité auprès des sujets dès lors qu'ils font appel à des systèmes externes : la réponse obtenue ne correspond pas toujours à la question posée. La formulation de la requête peut être longue et fastidieuse.

> *« Tous les assistants auxquels j'ai eu à faire étaient minables. Ca ne marche jamais. Je ne comprends rien ».*
>
> *« Ce n'est jamais ce problème que j'ai. Il y a des questions type, mais ce n'est jamais le bon truc ».*

En revanche, si le système répond correctement aux questions, les interviewés se disent prêts à répondre à plusieurs questions pour obtenir un résultat de qualité.

> *« Je veux bien répondre à 20 questions, si j'ai la bonne réponse, si je ne passe pas du temps à trier l'information derrière ».*

Les participants expriment une méfiance vis-à-vis des systèmes commerciaux. Ils souhaitent savoir si les services utilisés sont à but lucratif ou non. Ils souhaitent connaître leur qualité. Les sujets émettent l'hypothèse d'un label qualité certifiant la qualité des services :

> « *une certification de certains sites comme pour le paiement, une certification du caractère sérieux, fiable des sites ».*
>
> *« L'information doit être fiable pour accorder un crédit à ce système ».*

Ils souhaiteraient pouvoir filtrer l'utilisation des services selon ces labels qualité.

Les interviews ont permis de faire émerger des cas d'étude dans lesquels *Compose* serait particulièrement apprécié. Ces scénarios décrits en annexes sont utilisés dans le protocole des groupes de discussion que nous avons menés.

2.2 Groupe de discussion

Un groupe de discussion (Bruseberg et McDonagh-Philp 2002) est une forme d'étude qualitative. Les sujets sont questionnés sur leurs

perceptions, leurs opinions, leurs croyances et leurs attitudes vis-à-vis de *Compose*. Les questions sont posées à des groupes. Chaque participant peut cependant librement échanger avec des membres d'un autre groupe. Les groupes de discussion permettent de faire émerger des besoins utilisateurs pour *Compose*.

2.2.1 Protocole

Trois groupes de discussion ont été organisés : un auprès d'étudiants en informatique ; deux auprès du grand public. Le premier groupe était constitué de 9 étudiants : 2 femmes et 7 hommes ; 22 ans de moyenne d'âge ; 5 d'entre eux ayant fait du développement de sites web. Le deuxième groupe était constitué de 7 personnes : 4 femmes et 3 hommes ; 36 ans de moyenne d'âge ; 2 d'entre eux avaient fait du développement de sites web. Le troisième groupe était constitué de 8 personnes : 4 femmes et 4 hommes ; 43 ans de moyenne d'âge ; aucun n'avait d'expérience de développement logiciel.

Dans les trois groupes, les participants utilisent l'informatique pour des raisons professionnelles et privées. Seulement 4 ne l'utilisent que pour des raisons privées. Tous sont familiers du courrier électronique. Le premier groupe (celui des étudiants en informatique) utilise largement Internet. Le grand public utilise principalement Internet pour le courrier électronique et la recherche d'information généraliste, administrative ou touristique. Les discussions et forums semblent être propres aux plus jeunes (groupe des étudiants).

La durée de chaque groupe de discussion était de 2h30. Dans un premier temps, il était demandé aux participants de raconter comment ils organiseraient un déménagement précipité : rien n'est encore fait et le déménagement est prévu dans quinze jours. Dans un deuxième temps, les participants, groupés en binômes, recevaient un scénario les exposant à un problème (présenté en annexe A.2). Ils devaient énumérer les informations nécessaires pour formuler leur requête puis réfléchir aux résultats attendus. Ils esquissaient ensuite leur IHM favorite permettant de spécifier la requête et consulter les résultats. Chaque groupe présentait son scénario et sa maquette de façon collective. Les discussions étaient animées et ont fait émerger des besoins inattendus.

A la fin, quatre types d'interaction étaient présentés aux sujets (Figure 2.1) : langage naturel (a) ; langage naturel contraint (b) ; croquis (c, extrait de (Coyette et Vanderdonckt 2005)) ; iGoogle (d). Les résultats sont éloquents.

a) langage naturel.

b) langage naturel contraint.

c) croquis (Coyette et Vanderdonckt 2005).

d) iGoogle.

FIGURE 2.1 – *IHM présentées aux sujets pour animer les discussions quant à l'interaction souhaitée dans Compose.*

2.2.2 Résultats

La figure 2.2 présente quatre exemples de maquettes d'IHM dessinées pour le cas d'étude « See a doctor » par les sujets. La première est une maquette ressemblant à celle du cas d'étude présenté dans cette thèse. La deuxième minimise les services proposés car elle est consultable sur une borne publique. La troisième est une IHM permettant de sélectionner la rapidité et le coût du service. La dernière met en scène un avatar proposant les services à l'utilisateur.

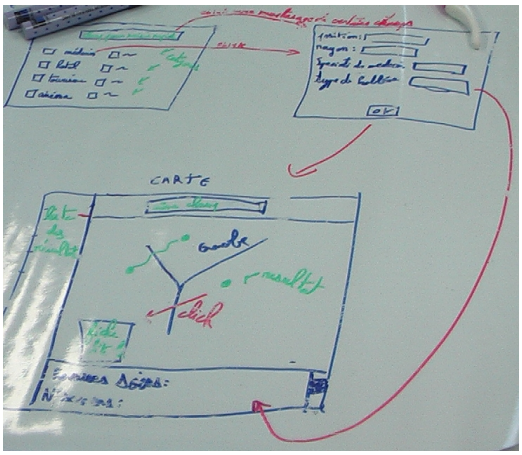
Les discussions montrent que la rapidité de calcul est la qualité première attendue par les sujets. La plupart voit en cet outil un gain de temps et une aide précieuse (13/24). Dix sont tout à fait convaincus de la valeur ajoutée d'un tel système. *Compose* est essentiellement perçu comme un outil simplificateur. Il doit être simple d'utilisation et en adéquation avec les besoins des utilisateurs (3 des sujets). Les scénarios privilégiés sont ceux du quotidien pour des recherches d'informations. Ce résultat confirme l'enquête qualitative. Le frein le plus important est l'originalité de *Compose* car les sujets n'ont pas d'exemple de système équivalent qui fonctionne. On retrouve également le frein classique de l'assistanat, à savoir le risque de dépendance et de rupture sociale.

Les dispositifs d'interaction plébiscités sont les dispositifs classiques, comme l'écran et le clavier (figure 2.3). Les dispositifs sonores sont sélectionnés uniquement par une personne sur trois. Les modes d'interaction originaux sont peu cités (un peu par les 20-25 ans) par manque d'« habitude » disent-ils.

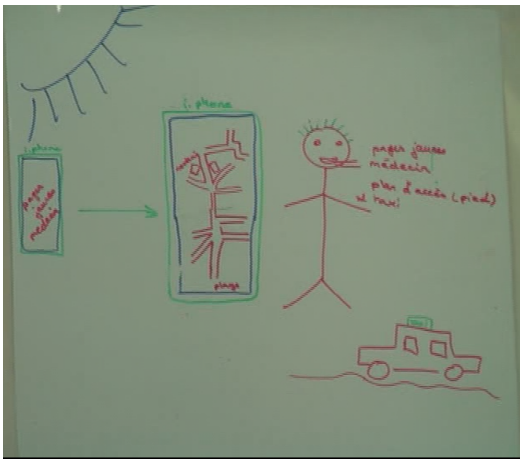
Les sujets optent massivement pour le langage naturel. 23 personnes sur 24 lui attribuent une note supérieure ou égale à 5 sur 10. La sélection manuelle de services à la iGoogle (principe des Mashups) est également plébiscitée (20/24). Le langage naturel est apprécié par une majorité pour affiner la requête et, en conséquence, préciser la réponse. Le nombre moyen de questions tolérées pour préciser une demande est deux fois plus élevé en situation nominale qu'en situation d'urgence (5.2 questions versus 2.5). Toutefois, des questions successives sont, pour certains, signe d'une qualité insuffisante du système :

> « *un système bien conçu ne doit pas poser de questions supplémentaires. Si question, c'est que le message que j'émets n'est pas bien reçu. L'interface est mal conçue* ».

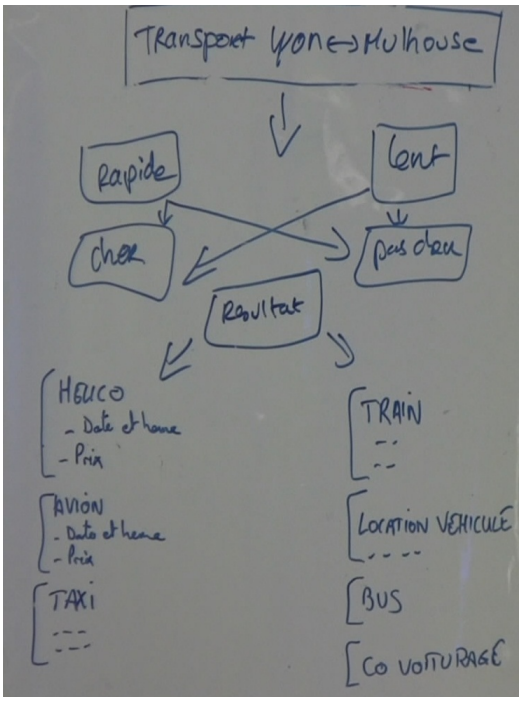
Au-delà des modalités d'interaction, les sujets ont exprimé des besoins fonctionnels. L'explication de la composition est, par exemple, massivement demandée : les sujets veulent comprendre comment *Compose* a



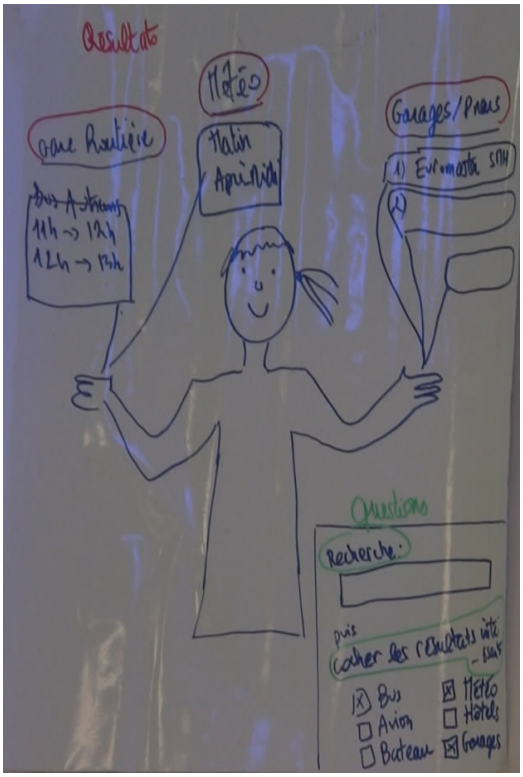
a) IHM ressemblant à notre maquette.



b) IHM consultable sur une borne.



c) IHM permettant de sélectionner la rapidité et le coût du service.



d) IHM avec un avatar proposant les services à l'utilisateur.

FIGURE 2.2 – *Exemples de maquettes d'IHM réalisées pour le cas d'étude « See a doctor ».*

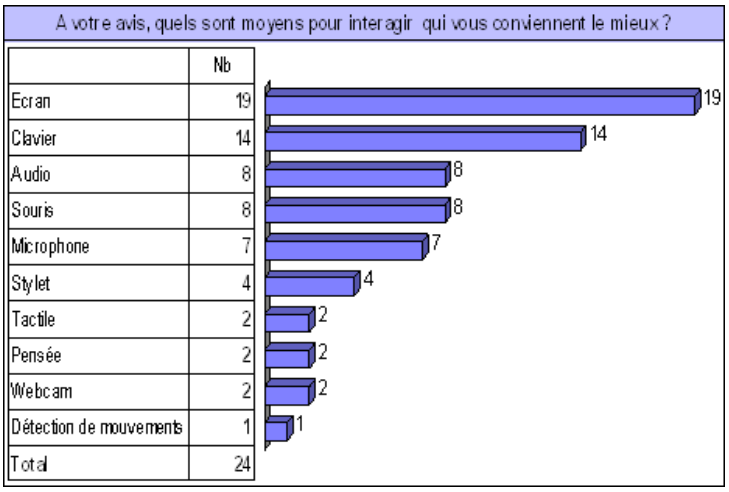


FIGURE 2.3 – *Dispositifs d'interaction favoris.*

procédé. Ils souhaitent aussi pouvoir sauvegarder le système interactif composé de façon à le réutiliser dans d'autres occasions et assurer ainsi la stabilité de l'interface.

2.3 CONCLUSION

L'étude que nous avons menée précise les cas d'étude préférés des utilisateurs et nous a permis de choisir le cas d'étude « See a doctor ». Elle démontre l'utilité de *Compose* dans des situations de vie quotidienne et d'urgence. Cette étude permet aussi d'orienter l'ingénierie du système de composition. Elle a, en particulier, une incidence directe sur la décomposition fonctionnelle de la fonction de composition : une capitalisation des systèmes interactifs est prévue conformément à la demande des utilisateurs. La langue naturelle est le moyen préféré de l'utilisateur pour spécifier son objectif. Les services composés devront être méta-décrits pour, en particulier, exprimer leur caractère commercial ou non.

Concepts fondamentaux 3

Résumé

Nous exposons ici les concepts fondamentaux en IHM puis en mathématiques sur lesquels se fonde notre travail. Nous introduisons la notion de « qualité » et les critères ergonomiques permettant de l'évaluer.
En IHM, nous présentons les niveaux d'abstraction classiques en modélisation des IHM. Ces niveaux ont été repris en plasticité des IHM pour le développement d'IHM douées d'adaptation à leur contexte d'usage dans le respect de leur qualité. Nous présentons les propriétés CARE qui permettent de raisonner sur les systèmes multimodaux.
En mathématiques, la composition est étudiée selon différentes approches. Nous retenons les relations topologiques temporelles entre intervalles de temps, et les relations spatiales entre régions.

Sommaire

Introduction

Ce chapitre présente les concepts fondamentaux en IHM sur lesquels s'appuie la thèse. La couverture de ce chapitre n'a donc pas pour objectif d'être exhaustive, mais de présenter les concepts utilisés dans la suite du manuscrit.

Ce chapitre se structure en cinq temps. Dans un premier temps, nous exposons les concepts en ergonomie des logiciels pour évaluer la qualité d'une IHM. Dans un deuxième temps, nous présentons les différents niveaux d'abstraction décrivant une IHM. Dans un troisième temps, nous présentons une manière de modéliser une IHM pour lui conférer la capacité de s'adapter à son contexte d'usage. Dans un quatrième temps, nous exposons les propriétés CARE qui permettent de raisonner sur les systèmes multimodaux. Pour finir, nous présentons les relations topologiques temporelles et spatiales.

3.1 Qualité d'une IHM

On utilise couramment le terme de *qualité* comme attribut des produits que nous fabriquons ou utilisons. Cette *qualité* devient une arme comparative entre produits (Daassi 2007). Mais qu'est-ce que la *qualité* ? Pour (Kitchenham et Pfleeger 1996), « la qualité est difficile à définir, impossible à mesurer et facile à connaître ».

En Interaction Homme-Machine, on a toujours parlé de l'*utilisabilité* (Nielsen 1993) des systèmes interactifs. L'utilisabilité a été introduite pour essayer de comprendre quels facteurs contribuent à une interaction réussie entre l'homme et la machine. Pour concevoir des systèmes utilisables, il fallait avoir des critères mesurables à référencer tout au long du processus de développement.

Plusieurs référentiels ont été proposés, provenant aussi bien d'ergonomes que d'informaticiens. Les terminologies utilisées divergent : on parle de facteurs, critères et caractéristiques. Les décompositions diffèrent elles aussi. La figure 3.1 collecte cinq référentiels (Scapin & Bastien (Scapin et Bastien 1997), Shackel (Shackel 2009), Nielsen (Nielsen 1993), IFIP (Gram et Cockton 1996), ISO (ISO/IEC CD 25000.2) sans les détailler et les comparer. La juxtaposition montre clairement l'absence de consensus. C'est l'*apprentissage* qui se retrouve dans le plus de travaux. Mais il n'est cité que trois fois sur cinq (Daassi 2007).

En raison de l'absence de consensus, nous choisissons les critères

Bastien & Scapin	**Shackel**	**Nielsen**	**IFIP**	**ISO**
Charge de travail	Efficacité	Apprentissage	Flexibilité	Compréhensibilité
Guidage	Apprentissage	Efficacité	Robustesse	Apprentissage
Contrôle explicite	Flexibilité	Mémorabilité		Opérabilité
Adaptabilité	Attitude	Erreurs		Attractivité
Gestion des erreurs		Satisfaction		Complétude
Homogénéité / Cohérence				
Signifiance des codes et dénominations				
Compatibilité				

FIGURE 3.1 – *Quelques référentiels pour raisonner sur la qualité d'une IHM (Daassi 2007).*

d'ergonomie de **Scapin & Bastien** comme support à l'évaluation de la *qualité* d'une IHM. Scapin et Bastien (1997) ont établi une liste de 18 dimensions à satisfaire réparties en 8 critères. Ils sont présentés de manière exhaustive en Annexes A.3.

Par exemple, le critère de « Guidage », dans sa dimension « Groupement/Distinction entre items » préconise de véhiculer dans la localisation et/ou le format l'appartenance ou non d'items à une même unité de présentation. Ainsi, à la lumière de cette recommandation, la figure 3.2 est structurée en trois unités, aussi appelées espaces de travail, distinguant les informations permettant d'appeler le docteur Mabuse, de se déplacer ou d'accéder à des services complémentaires comme la pharmacie de garde ou les stations services. De même, l'écriture en italique est utilisée pour distinguer le caractère obligatoire versus complémentaire de l'information. C'est ici l'application de ce critère dans sa dimension de format. L'utilisation de l'italique nuit par contre à la lisibilité de l'information, une autre dimension du guidage.

Les critères ergonomiques n'ont pas tous la même répercussion sur l'IHM. Par exemple, la lisibilité d'une IHM peut être améliorée en augmentant la taille des polices. Par contre, l'amélioration du « Groupement/Distinction entre items » peut entraîner une restructuration complète de l'IHM.

Le critère « compatibilité » se réfère à l'accord pouvant exister entre les caractéristiques des utilisateurs et des tâches d'une part, et l'organisation des sorties, des entrées et du dialogue d'une application

FIGURE 3.2 – *Dimensions Groupement/Distinction par la localisation et Lisibilité respectivement satisfaite et transgressée.*

d'autre part. En particulier, l'efficacité d'une IHM est accrue lorsque les procédures et les tâches sont organisées de manière à respecter les attentes/habitudes des utilisateurs. Ce critère peut entraîner le remplacement/suppression/ajout d'une partie de IHM. Par exemple, l'itinéraire peut être remplacé par le GPS si Victor est au volant. Comme le dévoilent ces exemples, les critères d'ergonomie ont des répercutions de profondeur différente.

3.2 Modélisation d'une IHM

Le projet européen CAMELEON (Calvary et al. 2003) a identifié des niveaux d'abstraction clés en conception d'IHM. Ces niveaux s'appuient sur l'architecture générale des Model-Based Interface Design Environment (MB-IDE) (Szekely 1996). Ils distinguent le niveau domaine (*tâches utilisateur* et *concepts*), les interfaces utilisateur *abstraite*, *concrète* et *finale* (figure 3.3).

Un **modèle de tâches** est une description récursive des tâches utilisateur en sous-tâches jusqu'à obtenir des tâches élémentaires (i.e, des tâches qui ne seraient décomposables qu'en actions physiques). Les tâches sont reliées entre elles par des *opérateurs entre tâches* exprimant

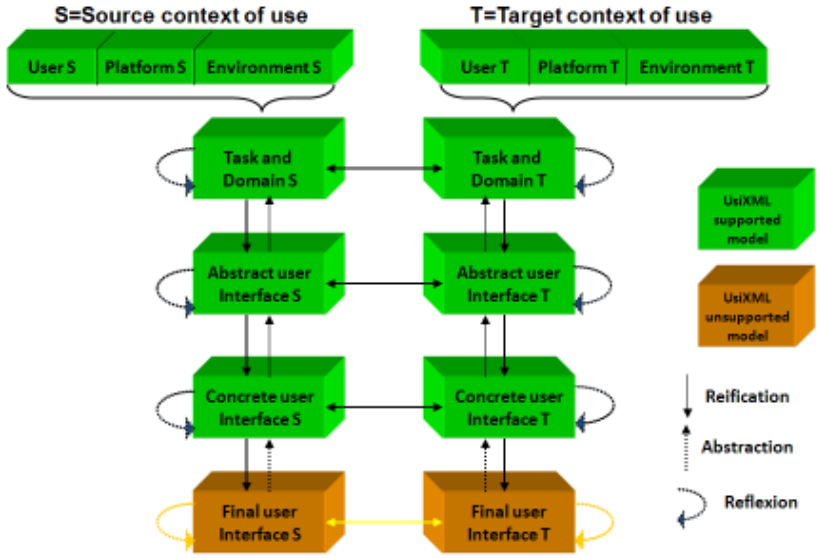


FIGURE 3.3 – *Niveaux d'abstraction d'une IHM (Lepreux et al. 2007).*

des relations logiques et/ou temporelles entre les sous-tâches.

La figure 3.4 montre un exemple de modèle de tâches décrit avec CTTe (Mori et al. 2002). Il représente les IHM des figures 1.3 et 1.4. Obtenir une assistance médicale (*Get medical assistance*), c'est choisir une ville (« Choose the city ») puis (>>) un médecin pour se soigner (« Choose the doctor ») puis (>>) le contacter (« Contact the doctor »). L'*opérateur entre tâches séquence* (notée « >> ») signifie que ces tâches sont exécutées l'une après l'autre. Contacter un médecin s'affine récursivement en appeler le cabinet médical (« Call the office ») et (|[]|) s'y rendre (« Find route information ») et (|[]|) trouver les informations sur le pharmacien le plus proche (« Find nearest chemist »). L'opérateur entre tâches *concurrence* (notée « |[]| ») signifie que ces tâches peuvent être exécutées en même temps.

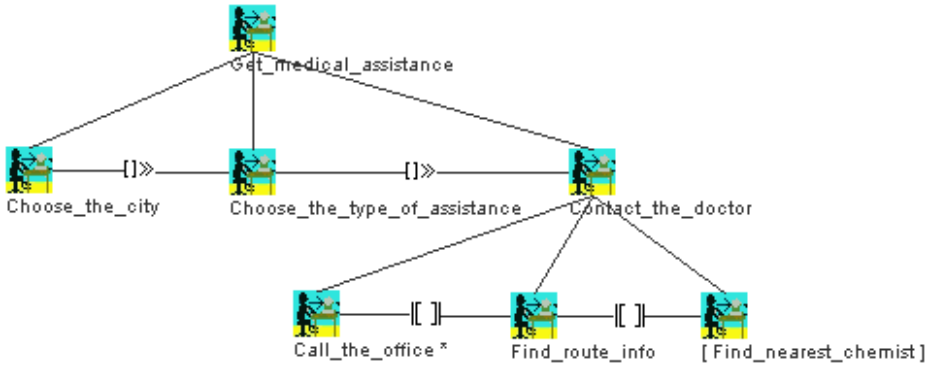


FIGURE 3.4 – *Exemple de modèle de tâches en CTTe.*

Typiquement, le critère de compatibilité entre tâches a un impact direct sur la décomposition de la tâche utilisateur. Par exemple, dans le cas où Victor n'aurait pas le permis, un taxi lui aurait été proposé au lieu de l'itinéraire.

En modélisation des tâches, en plus de la *séquence* et de la *concurrence* décrites précédemment, on dispose principalement des opérateurs entre tâches suivants :

- un *ordre indépendant* (noté « $|=|$ ») : les sous-tâches sont partiellement ordonnées et sans concurrence. Par exemple, les tâches « Call the office » et « Find route information » ne pourraient pas être exécutées en même temps mais l'ordre serait libre.

- un *choix* (noté « [] ») : une seule sous-tâche est possible pour l'utilisateur. Par exemple, la tâche « s'identifier » peut être faite par adresse email ou login. L'utilisateur peut effectuer l'une ou l'autre, mais pas les deux.

Les *tâches* peuvent être décorées de propriétés. Dans l'exemple précédent, la tâche « Call the office » est décorée d'une *itération*. L'*itération* signifie que la tâche peut être réalisée plusieurs fois. Typiquement, si l'utilisateur se perd en route ou prend du retard dans les embouteillages, il apprécierait de pouvoir en avertir le médecin. La tâche « Find more information » est décorée d'une *optionalité*. L'*optionalité* signifie que la réalisation de la tâche n'est pas obligatoire : elle est facultative, laissée à l'appréciation de l'utilisateur.

En plus de ces décorations, une tâche est classiquement décorée des *Concepts*. Par exemple, la tâche permettant d'appeler un médecin « Call the office » manipule le concept de médecin « Doctor ».

Les concepts sont décrits dans un **modèle de concepts**. Ce modèle est classiquement représenté par un diagramme de classe UML. La figure 3.5 illustre le concept de « Doctor ». Un médecin a un nom (name). Il travaille dans au moins un bureau (Office). Il soigne des patients. Les patients ont un nom et peuvent se rendre chez un pharmacien (Chemist). La pharmacie, le bureau du médecin et le patient ont une adresse comportant une rue.

L'**Interface Utilisateur Abstraite** (IUA) structure l'IHM en *espaces de travail* (aussi nommés espaces de dialogue ou unités de présentation) et fixe la navigation entre ces espaces. Un espace de travail est un « lieu d'activité virtuel offrant les éléments nécessaires à la réalisation d'une

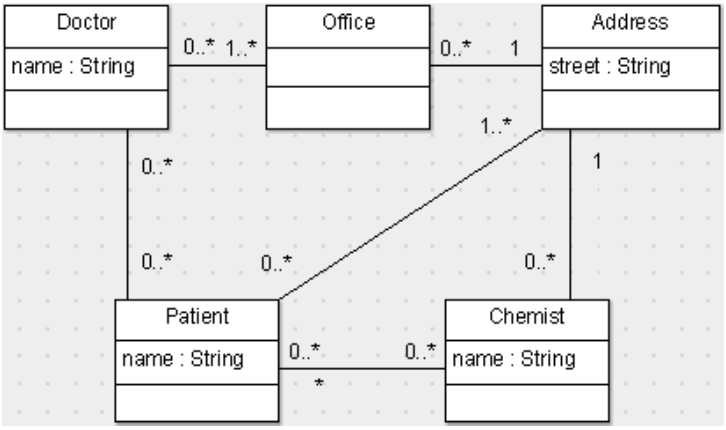


FIGURE 3.5 – *Extrait d'un modèle de concepts.*

ou plusieurs tâches » (Normand 1992). Clairement, l'identification des espaces de travail répond à une préoccupation de groupement (spatial dans le cas d'IHM graphiques). Cette préoccupation est motivée par le critère de « Guidage-Groupement/Distinction entre items » tel que formulé dans le référentiel de Scapin et Bastien (1997). La figure 3.6 présente les espaces de travail de la figure 3.4, plaqués sur le modèle de tâches.

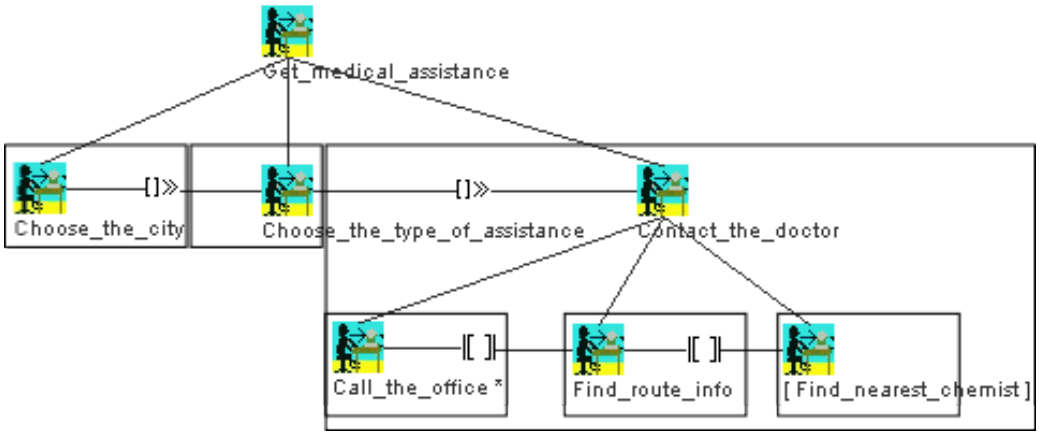


FIGURE 3.6 – *Tâches regroupées en espaces de travail.*

L'**Interface Utilisateur Concrète** (IUC) fait le choix d'interacteurs pour :

- les espaces de travail alors réifiés en fenêtres ou canevas dans le cas d'IHM graphiques. La figure 3.7 présente les réifications (flèches) des espaces de dialogue en fenêtres ;
- le contenu des espaces de travail : pour les espaces de travail, les tâches élémentaires et concepts du domaine sont classiquement des boutons radio, cases à cocher, textes, etc.
- la navigation entre espaces est classiquement matérialisée par des

séparateurs (espaces ou traits) ou des objets de navigation (boutons, liens hypertexte, etc.).

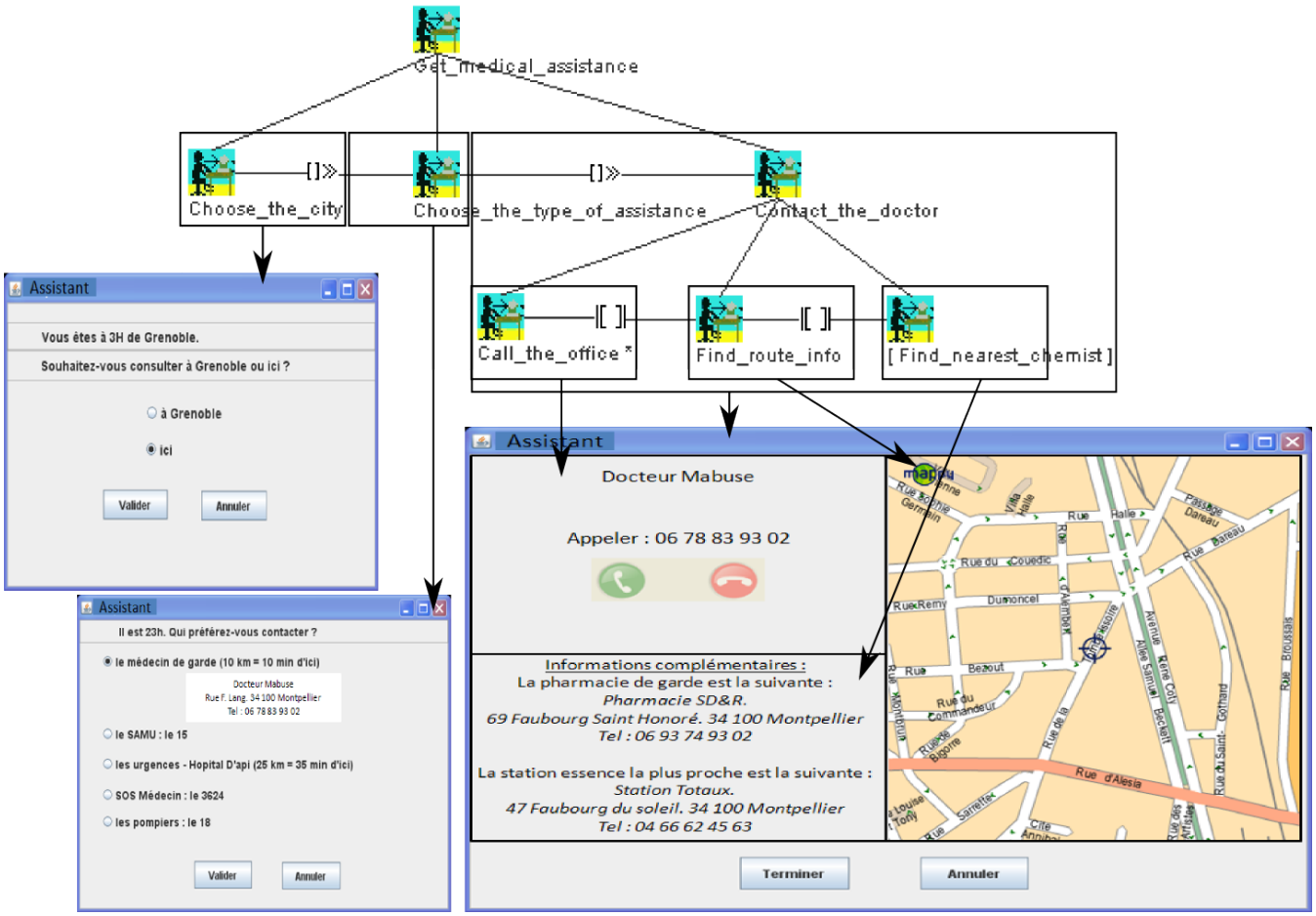


FIGURE 3.7 – *Tâches réifiées en espaces de travail, eux-mêmes réifiés en fenêtres et canevas.*

L'**Interface Utilisateur Finale** (IUF) fait le choix d'environnements de programmation et d'exécution.

Ces niveaux d'abstraction ont été repris pour raisonner sur l'adaptation des IHM à leur contexte d'usage : la plasticité des IHM.

3.3 PLASTICITÉ

La propriété de *Plasticité* a été introduite en 1998 comme étant la « capacité d'une IHM à s'adapter à son contexte d'usage dans le respect de son utilisabilité » (Thevenin et Coutaz 1999).

Les niveaux d'abstraction ont été repris pour raisonner sur la conception d'IHM plastiques. En effet, l'impact d'une modification, suite à un changement de contexte d'usage, peut être longitudinal et toucher tous les niveaux d'abstraction. Typiquement, la mise à disposition subite d'un service à valeur ajoutée pour la tâche de l'utilisateur modifie le niveau tâches et peut, en conséquence, ajouter un espace de travail dans l'IUA et consécutivement des interacteurs dans l'IUC et l'IUF.

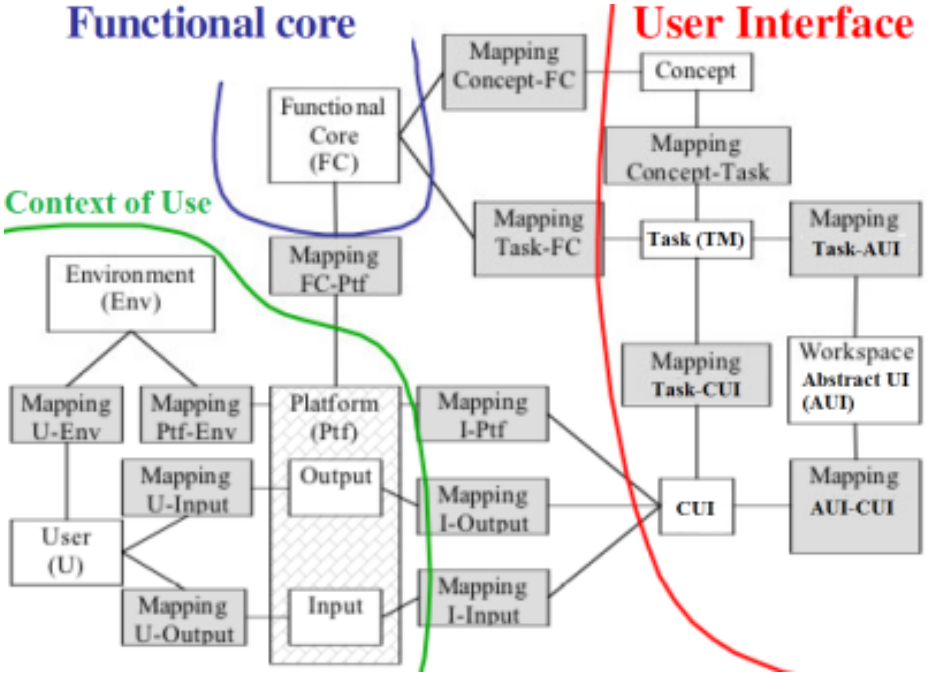


FIGURE 3.8 – *Graphe de modèle d'un système interactif (Sottet et al. 2007).*

Sottet et al. (2007) se fondent sur les niveaux d'abstraction pour décrire une IHM en un graphe de modèles et de relations (*mappings*). Les relations (mappings) entre modèles correspondent aux choix de conception. Ils sont motivés par les propriétés ergonomiques à satisfaire. Ils couvrent :

- les mappings intra-IHM entre les niveaux d'abstraction du processus de conception : Concept-Task, Task-AUI, AUI-CUI et Task-CUI. Concept-Task énumère les concepts manipulés dans chaque tâche. Par exemple, la tâche « Call the office » manipule le concept de médecin et son bureau. Task-AUI assigne des espaces de travail aux différentes tâches. Par exemple, la tâche « Call the office » est liée à l'espace de travail la réifiant (flèche) dans la figure 3.6. AUI-CUI et Task-CUI associent des interacteurs aux espaces de travail et à leur contenu. Par exemple, l'espace de travail réifiant la tâche « Call the office » contient un bouton pour décrocher le téléphone ;
- les mappings inter NF-IHM : Concept-FC et Task-FC établissent les liens entre les données et fonctions du NF d'une part et les concepts

et tâches de l'IHM d'autre part. Par exemple, la tâche « Call the office » est liée à une fonctionnalité pour appeler une personne à partir de son numéro de téléphone ;
- les mappings inter Contexte-IHM modélisent le déploiement de l'IHM sur les ressources d'interaction (I-Ptf) en entrée et en sortie (I-Input et I-Output). Par exemple, l'IHM composée pour Victor dans la version 1 est exécutée sur l'écran du mur numérique

(Demeure et al. 2008) proposent une boîte à outils d'interacteurs plastiques. Un interacteur plastique est un interacteur défini au niveau tâches (par exemple, l'interacteur « choisir ») et comporte plusieurs présentations (par exemple, des boutons radio, un menu linéaire, circulaire, etc.) lui offrant, en conséquence, un potentiel d'adaptation. Ces interacteurs, appelés COMETs, embarquent en fait un micro graphe de modèles tel que défini dans la figure 3.8. Les auteurs militent pour embarquer ce graphe à l'exécution pour que l'IHM soit pleinement plastique, i.e., que tous les choix de conception puissent être revisités à l'exécution suite à un changement du contexte d'usage. Le graphe exprime alors et maintient tous les niveaux d'abstraction durant l'exécution de l'IHM.

3.4 Multimodalité : propriétés CARE

Une *modalité* d'interaction est définie par Nigay et Coutaz (1995) comme un couple $< d, L >$ où d désigne un *dispositif physique* et L, un *langage d'interaction*. Un dispositif physique est un élément du système qui acquiert des informations (dispositif d'entrée) ou en fournit à l'utilisateur (dispositif de sortie). Des exemples de dispositifs sont un clavier, une souris, etc. Un langage d'interaction définit un ensemble d'expressions bien formées (par exemple, un assemblage conventionnel de symboles). La composition de modalités a été étudiée et caractérisée. Ces études fournissent une base théorique et des propriétés intéressantes pour la composition d'IHM.

Les propriétés CARE (Complémentarité, Assignation, Redondance et Equivalence) (Coutaz et al. 1995) permettent de raisonner sur la multimodalité en IHM. La description des propriétés CARE utilise les notions d'état, but, modalité, relation temporelle :
- un *état* est un vecteur de variables observables, un ensemble de propriétés mesurables à un instant donné qui caractérise une situation.
- un *but* est un état à atteindre par un *agent*, i.e., l'utilisateur ou le système ou un composant du système. Cet agent est une entité capable d'initialiser le déroulement d'une action.

- une *modalité* est une méthode d'interaction utilisée par un agent pour atteindre un but. Pour estimer le pouvoir d'expression d'une modalité, les auteurs définissent une fonction appelée « Reach » qui prend la valeur vraie, pour deux états, si une modalité permet de passer directement de l'un à l'autre.
- une *relation temporelle* caractérise l'utilisation dans le temps d'un ensemble de modalités. Cette relation est une séquence ou un parallélisme. Une séquence signifie que les modalités sont actives l'une à la suite de l'autre. Le parallélisme signifie que les modalités sont actives en même temps.

Les propriétés CARE sont utiles pour classer et évaluer les systèmes multimodaux. Ces propriétés définissent trois types de composition de modalités : la complémentarité, la redondance et l'équivalence. L'assignation désigne au contraire l'absence d'alternatives.

- L'*équivalence* entre modalités $M_1, \ldots M_n$ signifie que des modalités permettent toutes individuellement d'atteindre le but. Cette propriété exprime l'existence d'un choix entre un ensemble de modalités qui permettent d'atteindre l'objectif utilisateur. Par exemple, pour entrer son objectif (but), Victor peut saisir « Je veux voir un médecin » avec un clavier ou à la voix.
- La *redondance* exprime que des modalités sont équivalentes et utilisées séquentiellement ou parallèlement.
- La *complémentarité* entre les modalités d'un ensemble exprime le fait que pour passer d'un état à un autre, il faut utiliser toutes les modalités de cet ensemble. Cela signifie qu'une seule modalité de cet ensemble ne suffit pas à elle seule pour passer de l'état de départ à l'état d'arrivée.

3.5 Relations topologiques

La composition est étudiée en mathématiques sous la forme d'« espaces ». Ces « espaces » mathématiques mettent en avant différentes notions qui peuvent permettre de caractériser différemment les positions des objets (Bessière et al. 1997) :

- les espaces topologiques procurent les notions de connexité et d'inclusion ;
- les espaces vectoriels considèrent l'alignement ;
- les espaces métriques permettent de mesurer des distances entre les objets.

Les deux approches quantitative et qualitative s'opposent dans la supposition de l'existence d'une mesure. On peut aussi opposer métrique à topologique et numérique à symbolique. Il serait concevable de travailler sur des approches numériques, et lorsqu'on décrit de manière précise une situation entièrement définissable par des paramètres physiques, c'est évidemment la meilleure solution. Cependant, nous disposons beaucoup plus fréquemment de données qualitatives, car les conclusions qui nous intéressent sont également le plus souvent qualitatives, et les modèles symboliques sont généralement mieux à même de travailler sur le qualitatif.

Nous retenons ici les espaces topologiques pour raisonner de manière symbolique. Nous en présentons les relations temporelles et spatiales.

Relations temporelles

Une approche qualitative pour la caractérisation des connaissances temporelles est celle de Allen (1983). Il caractérise des intervalles temporels liés par des relations qualitatives (par exemple, un intervalle peut en chevaucher un autre, sans que l'on ait d'indications sur la durée de chacun d'entre eux). Plus généralement, on peut être amené, par exemple lorsqu'on s'intéresse à la composition de processus qui comportent des phases de repos et des phases d'activité, à caractériser des relations entre intervalles d'exécution.

Allen (1983) définit treize relations possibles entre deux intervalles de temps (figure 4.6) :

- *Meet* signifie que le deuxième intervalle succède immédiatement au premier dans le temps. Sa relation inverse est *met-by*.
- *Before* signifie que le deuxième succède au premier dans le temps avec un laps de temps entre les deux. Sa relation inverse est *after*.
- *Equal* signifie que les deux intervalles sont dans la même temporalité. Sa relation inverse est elle-même.
- *Start* signifie que les deux intervalles commencent en même temps, mais que le premier se termine avant le deuxième. Sa relation inverse est *started-by*
- *Finish* signifie que le premier intervalle commence après le deuxième, mais qu'il se termine en même temps. Sa relation inverse est *finished-by*.
- *Overlap* signifie que le deuxième intervalle chevauche le premier. Sa relation inverse est *over-lapped-by*.
- *During* signifie que le premier intervalle a lieu pendant le deuxième. Sa relation inverse est *during inverse*.

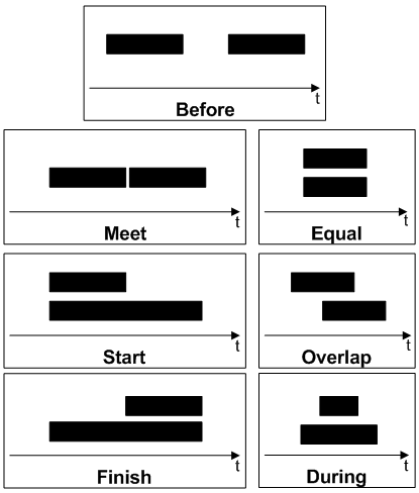


FIGURE 3.9 – *Relations temporelles entre intervalles de temps.*

Relations spatiales

Une approche largement adoptée pour caractériser des connaissances spatiales est celle de Egenhofer et Franzosa (1991). Les auteurs caractérisent les relations qualitatives entre régions spatiales en deux dimensions (par exemple, une région peut en recouvrir une autre, sans que l'on ait d'indication sur la taille de la surface recouverte). Plus généralement, on peut être amené, par exemple lorsqu'on s'intéresse à la composition d'IHM, à caractériser des IHM graphiques par leur région d'exécution sur l'écran ou des IHM vocales par leur région de propagation du son. Les sens perceptifs humains (vision et ouïe) permettent à l'utilisateur de percevoir plusieurs éléments au même endroit.

Egenhofer et Franzosa (1991) définissent huit relations possibles entre régions suivantes (figure 3.10) :
- *Disjoint* signifie que les régions sont séparées.
- *Meet* signifie que les régions sont côte à côte.
- *Overlap* signifie que les régions sont superposées.
- *Equal* signifie que les régions partagent exactement le même espace.
- *Covers* signifie qu'une région couvre l'autre. Sa relation inverse est *covered-by*.
- *Contains* signifie qu'une région contient l'autre. Sa relation inverse est *inside*.

3.6 Conclusion

A l'heure actuelle, plusieurs référentiels permettent d'évaluer la qualité de l'IHM. Ils servent d'aide au concepteur d'IHM mais leur prise en

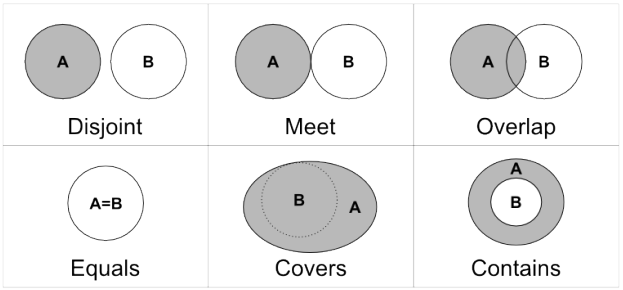


FIGURE 3.10 – *Relations spatiales entre régions.*

compte automatique nécessite leur formalisation. Une IHM est modélisée à différents niveaux d'abstraction. Ces niveaux permettent de séparer les préoccupations lors de la conception, chaque niveau identifiant des aspects à prendre en compte. Ces niveaux ont ainsi été repris pour modéliser des IHM plastiques. Les propriétés CARE permettent de raisonner sur la multimodalité en IHM. Ces propriétés sont ainsi une base pour raisonner formellement sur la composition d'IHM. De la même manière, les relations topologiques temporelles et spatiales sont des relations formelles permettant la compréhension des relations temporelles et spatiales dans la composition d'IHM.

Espace problème de la composition d'IHM

4

Résumé

Ce chapitre dresse un espace problème de la composition dynamique d'IHM. Dans un premier temps, il en définit les concepts fondamentaux notamment la notion de composeur, une fonction qui compose des IHM pour produire une IHM composite. Dans un deuxième temps, nous proposons une taxonomie des composeurs. Cette taxonomie permet de décrire un composeur selon différents niveaux d'abstraction et ainsi de séparer les préoccupations dans le but de raisonner sur la qualité de l'IHM composite. Dans un troisième temps, nous analysons le processus de composition d'IHM en proposant une décomposition fonctionnelle et un espace problème. L'espace problème de la composition d'IHM permet de caractériser l'état de l'art sur la composition d'IHM dans le chapitre suivant.

Publication associée : TSI'11 (Gabillon et al. 2011a)

Sommaire

Introduction

Ce chapitre présente un espace problème de la composition d'IHM. Dans un premier temps, il présente les concepts fondamentaux de la composition d'IHM. Dans un deuxième temps, nous proposons une taxonomie des composeurs. Dans un troisième temps, le chapitre décompose le fonctionnement du processus de composition d'IHM et en caractérise les problèmes.

4.1 Définitions

La composition dynamique d'IHM s'appuie sur les définitions suivantes :

4.1.1 Modèle

Il n'existe pas de définition unifiée de la notion de *modèle* (Sottet 2008). Cependant on peut trouver un consensus autour de la relation entre un modèle et le système étudié et de leur complémentarité (Bézivin 2005). La relation entre un système et le modèle étudié est notée μ (Favre 2004) : on dit qu'un modèle est une représentation simplifiée d'un système. Cette représentation (modèle) est utilisée pour répondre à des questions à la place du système. Par exemple un diagramme de séquence donne l'ensemble des objets impliqués dans une cascade d'appels de méthodes. Le modèle permet par conséquent au développeur de comprendre le système, sans avoir à décrypter le code. Dans l'exemple, le modèle (de tâches) de la figure 3.4 sert d'abstraction à l'IHM sur le mur numérique de la figure 1.4.

La composition nécessite de raisonner sur les systèmes à composer. Il est donc en conséquence important de disposer de modèles. Aussi, nous nous intéressons à la composition de modèles de systèmes, en particulier de modèles d'IHM.

Modèle d'une IHM

Une IHM est modélisée par un graphe de modèles. Les nœuds sont les modèles fondamentaux en IHM (c.f. chapitre 3) : tâches, concepts, espaces de travail, interacteurs, etc. Les arcs sont les transformations ou mises en correspondance entre ces modèles. Par exemple, l'IHM pour aller consulter un médecin est modélisée par la tâche « Find route

info ». Cette tâche est associée à l'espace de travail w_{find_route}. Cet espace est concrétisé en un canevas. Le canevas contient la carte utile à la tâche « Find route info », la date du jour et les boutons « Cancel » et « Validate ».

IHM élémentaire

Lorsque la tâche est élémentaire (c.f. chapitre 3), c'est-à-dire décomposable qu'en actions physiques, alors l'IHM est dite *élémentaire*.

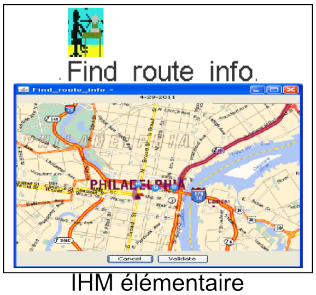


FIGURE 4.1 – *IHM élémentaire pour se déplacer.*

IHM composée

Une *IHM non élémentaire* est dite *composée* : la tâche utilisateur se décompose en sous-tâches. Par exemple, l'IHM composée pour le mur numérique car sa tâche utilisateur « get medical assistance » se décompose en sous-tâches.

4.1.2 Composeur

Un *composeur* est un opérateur de composition. Il compose une séquence d'IHM. L'IHM résultante, dite *composite*, est soit élémentaire, soit composée.

Si elle est élémentaire, le composeur avait pour vocation de renforcer certains critères d'ergonomie pour par exemple, accentuer un concept important. Il est alors représenté deux fois.

Si elle est composée, le composeur avait pour vocation de créer une nouvelle tâche utilisateur. Le composeur se définit fonctionnellement au niveau tâches et peut avoir une empreinte perceptible à l'utilisateur (par exemple onglets, boutons de navigations, etc.) C'est donc une IHM que l'on appellera *Composeur*, ou *IHM composeur*. Dans la figure 4.2, le composeur compose l'IHM pour appeler un médecin et s'y rendre. Son IUC

est un onglet. Ce composeur produit une IHM composite permettant de réaliser la tâche « Contact the doctor ».

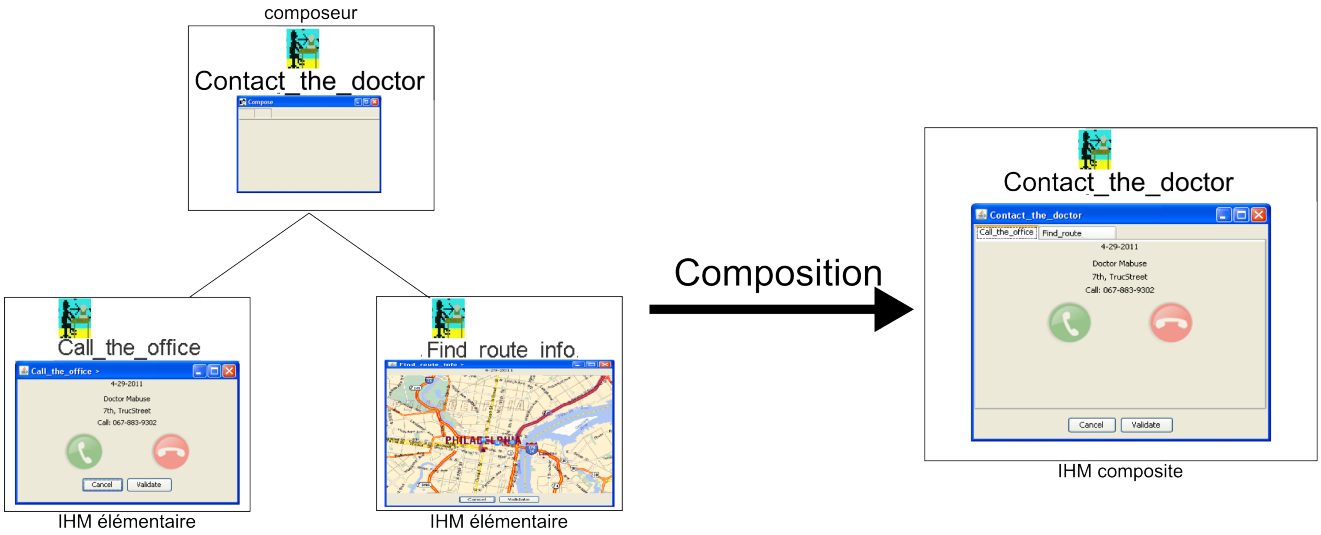


FIGURE 4.2 – *Composition des IHM pour appeler le médecin et s'y rendre.*

Schéma de composition

Le schéma de composition est le modèle niveau tâches d'un composeur.

État

Un *état* se définit par un ensemble de propriétés vérifiables à un instant d'observation.

But

Un *but* est un état que l'utilisateur souhaite atteindre. Par exemple, l'IHM composite pour Victor doit lui permettre d'atteindre un état de bonne santé.

4.2 TAXONOMIE DES COMPOSEURS

4.2.1 Composeurs de base

En mathématiques, les opérateurs sont définis pour composer des ensembles. Les opérateurs binaires de base sont l'*union*, l'*intersection* et

la *différence* (Halmos 1960). Ces trois opérateurs sont dits de base, car les autres opérateurs peuvent être exprimés par leur composition (au sens mathématique). Par exemple, le complémentaire peut être exprimé par l'intersection et la différence.

Les composeurs sont la composition (au sens mathématique) des opérateurs de base : union, intersection et différence.

– L'*union* produit une IHM constituée de l'union des éléments des deux IHM source et de l'IHM composeur. Par exemple, l'union des IHM pour appeler le médecin et s'y rendre est une IHM assemblant dans un onglet les IHM sources.

– L'*intersection* produit une IHM constituée des éléments communs aux IHM sources. Par exemple, l'intersection entre les IHM pour appeler un médecin et s'y rendre (figure 4.3) produit une IHM qui réalise la tâche « display current date ». A tous niveaux d'abstraction, seuls les canevas, la date du jour et les boutons « Validate » et « Cancel » sont communs aux deux IHM. Il vont être composés dans un canevas. Le composeur n'a ici pas de représentation.

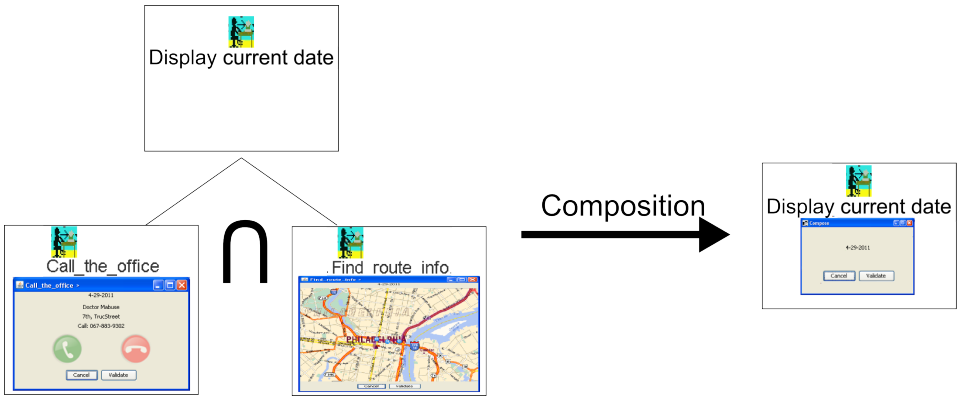


FIGURE 4.3 – *Un exemple d'intersection.*

– *différence* produit une IHM constituée des éléments modélisant la première IHM privée des éléments communs de la deuxième. Par exemple, la différence entre les IHM pour appeler un médecin et s'y rendre produit une IHM privée de la date du jour et des boutons « Validate » et « Cancel ».

La différence et l'intersection sont rendues observables (on peut imaginer que ce soit souhaitable), alors le problème peut être réécrit en $Union(IHMintersecter/diffrencier, IHMcomposeur)$.

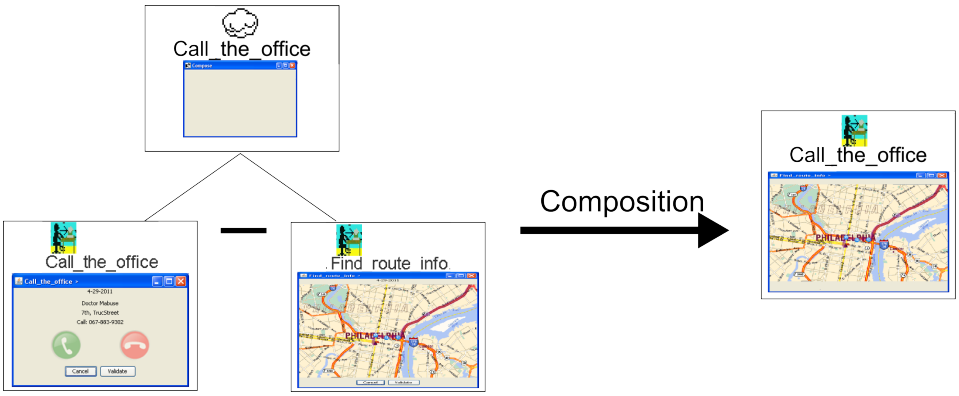


FIGURE 4.4 – *Un exemple de différence.*

On constate que l'union est un opérateur qui demande une étude approfondie. Il peut être affiné en différentes variantes. Par exemple, les IHM du cas d'étude sont toutes composées avec l'union, pourtant on constate des différences visibles de composition. En particulier, les IHM pour appeler un médecin et le consulter ne sont pas composées de la même manière selon la variante.

Pour affiner les composeurs, il est nécessaire de s'intéresser à la nature d'éléments à composer et à leurs relations entre eux (figure 4.5). Nous étudions ces relations à différents niveaux d'abstraction. Ainsi, nous identifions deux aspects de composition :

– l'aspect temporel au niveau tâches ;
– l'aspect spatial au niveau interface utilisateur concrète (IUC).

La suite se structure selon ces deux aspects.

4.2.2 Aspect temporel

L'aspect *temporel* caractérise les relations possibles entre tâches dans une fenêtre temporelle, c'est-à-dire un intervalle de temps. Ces relations sont mises en évidence au niveau tâches par les opérateurs entre tâches (Paternò et al. 1997). A ce niveau, une tâche est active/exécutée pendant un intervalle de temps. Une composition de tâches peut s'exprimer par une relation temporelle entre ces intervalles. Les relations temporelles possibles entre intervalles de temps sont définies par Allen (Allen 1983) et présentées dans le chapitre « Concepts Fondamentaux » 3.5. Allen définit 13 relations binaires possibles entre intervalles. Nous réutilisons ces relations pour caractériser l'aspect temporel de la composition d'IHM.

Selon l'aspect temporel, deux tâches t_1 et t_2 sont composées par une

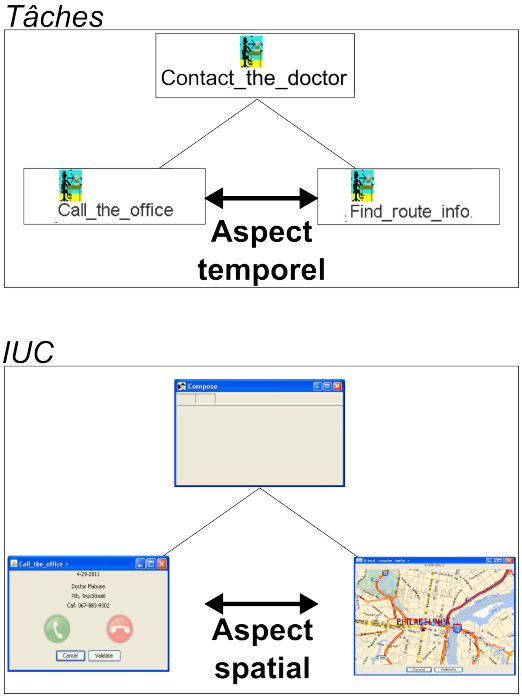


FIGURE 4.5 – *Relations entre IHM à composer par l'union.*

relation temporelle entre leurs intervalles de temps respectifs $tw1$ et $tw2$:

- $Before(t_1, t_2)$ (resp. $After(t_2, t_1)$) signifie que tw_1 précède tw_2. Cette relation exprime que la tâche t_2 doit être active un certain laps de temps une fois t_1 réalisée. Ce laps de temps est défini par le concepteur ou l'utilisateur. Par exemple, un utilisateur peut spécifier dans un agenda un laps de temps avant que son rendez-vous n'apparaisse.

- $Meet(t_1, t_2)$ (resp. $Met\text{-}by(t_2, t_1)$) signifie que tw_1 est juste avant (resp. juste après) tw_2. Cette relation exprime que t_2 (resp. t_1) doit être active juste après la fin de t_1 (resp. t_2). Par exemple, dans le cas d'étude, l'utilisateur choisit une ville puis un type d'assistance pour se soigner.

- $Equal(t_1, t_2)$ signifie que tw_1 et tw_2 sont en parallèle. Cette relation exprime que t_1 et t_2 sont actives en même temps. Par exemple, cette relation est nécessaire pour exprimer la relation entre les tâches « Call the office » et « Find route info » dans la version 2 (figure 1.4).

- $Start(t_1, t_2)$ (resp. $Started\text{-}by(t_2, t_1)$) signifie que tw_1 et tw_2 commencent en même temps mais tw_1 (resp. $tw2$) se termine en premier. Cette relation exprime que t_1 et t_2 doivent être actives

en même temps, mais que la première doit se terminer avant la deuxième. Cette relation permet de caractériser les situations où une tâche doit obligatoirement se terminer avant une autre. Par exemple, une publicité peut être composée en même temps qu'un site internet quelconque et peut empêcher sa fermeture, tant que que la publicité n'est pas consultée.

– $Overlap(t_1, t_2)$ (resp. $Over\text{-}lapped\text{-}by(t_2, t_1)$) signifie que tw_2 (resp. tw_1) doit commencer et se terminer après tw_1 (resp. tw_2). Cette composition permet de caractériser les cas où une tâche doit obligatoirement se terminer pour que la suivante soit activée. Par exemple, l'IHM composée dans le cas d'étude pourrait attendre que Victor arrive à joindre le médecin pour lui proposer l'itinéraire.

– $Finish(t_1, t_2)$ (resp. $Finished\text{-}by(t_2, t_1)$) signifie que tw_1 (resp. tw_2) doit commencer après tw_2 (resp. tw_1) et que tw_1 et tw_2 doivent se terminer en même temps. Cette relation décrit qu'une IHM doit obligatoirement être active après une autre. Par exemple, la tâche « consulter le carnet d'adresses » peut être active après la tâche « téléphoner », et se terminer en même temps.

– $During(t_1, t_2)$ ($During\_inverse(t_2, t_1)$) signifie que tw_1 (resp. tw_2) doit commencer après et se terminer avant tw_2 (resp. tw_1). Cette composition permet de caractériser les cas où une tâche s'active dans l'intervalle d'une autre. Par exemple, la confirmation d'installation des mises à jour automatiques a lieu pendant que le bureau du système d'exploitation est actif.

La figure 4.6 rappelle les relations temporelles entre deux intervalles de temps représentant t_1 et t_2 et présentées dans le chapitre « Concepts Fondamentaux » 3.5.

Pour résumer, au niveau tâches, faire l'union de deux IHM, c'est exprimer entre leur tâche, une relation temporelle ou logique : $\{Before, After, Meet, Met\text{-}by, Equal, Start, Started\text{-}by, Overlap, Over\text{-}lapped\text{-}by, Finish, Finished\text{-}by, Durring, During\_inverse\}$.

4.2.3 Niveau IUC : aspect spatial

Un interacteur est déployé dans une région d'exécution. Par exemple, la région d'exécution des interacteurs graphiques est un écran. La région d'exécution des interacteurs vocaux est la région de propagation du son. Les interacteurs olfactifs ont une région de diffusion de l'odeur. Ces

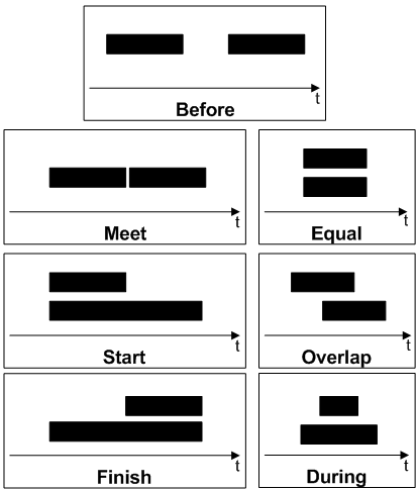


FIGURE 4.6 – *Relations temporelles entre deux tâches t_1 et t_2.*

régions reflètent la volonté du concepteur ou du système de composer les IHM à des endroits spécifiques de l'environnement de perception de l'utilisateur.

En mathématiques, les relations spatiales entre régions sont étudiées par Egenhofer et Franzosa (1991) et présentées dans le chapitre « Concepts Fondamentaux » 3.5. Ils définissent huit relations spatiales possibles entre régions.

Nous utilisons ces relations spatiales pour décrire la composition de deux interacteurs. Selon l'aspect spatial, les interacteurs i_1 et i_2 sont déployés respectivement sur les régions r_1 et r_2, composées par une relation spatiale :

- $DisjointI(i_1, i_2)$ signifie que r_1 et r_2 sont éloignées. Une composition disjointe caractérise les cas où l'utilisateur perçoit une distance entre les IHM composées. Par exemple, l'IHM pour appeler un médecin et celle pour s'y rendre sont composées de manière disjointe sur le mur numérique et le Smartphone dans la version 3, car les canevas contenant les autres interacteurs sont disjoints.

- $MeetI(i_1, i_2)$ signifie que r_1 et r_2 sont à côté. Cette composition caractérise les cas où les interacteurs sont côte à côte.

- $OverlapI(i_1, i_2)$ signifie que r_1 recouvre r_2. Cette composition caractérise le chevauchement de deux interacteurs.

- $EqualI(i_1, i_2)$ signifie que r_1 et r_2 sont égales. Cette composition caractérise deux interacteurs composés au même endroit. Par

exemple, les canevas des IHM composées dans un onglet partagent la même région.

– $CoversI(i_1, i_2)$ (resp. $Covered\text{-}byI(i_2, i_1)$) signifie que r_1 couvre (resp. est couverte par) r_2. Cette composition caractérise qu'un interacteur en recouvre un autre. Par exemple, un canevas peut en recouvrir un autre.

– $ContainsI(i_1, i_2)$ (resp. $InsideI(i_2, i_1)$) signifie que r_1 contient (resp. est contenue dans) r_2. Cette composition caractérise qu'un interacteur en contient un autre. Par exemple, un canevas peut en contenir un autre.

La figure 4.7 présente des variantes d'union d'IHM pour appeler un médecin et s'y rendre selon différentes relations spatiales entre leur canevas.

Au niveau IUC, faire l'union de deux IHM, c'est exprimer une relation spatiale entre leurs interacteurs : $\{DisjointI|MeetI|OverlapI|EqualI|CoversI|Covered\text{-}byI|ContainsI|InsideI\}$.

4.2.4 Discussion

Nous disposons d'une taxonomie (figure 4.8) des composeurs à différents niveaux d'abstraction. Cette taxonomie permet la séparation des préoccupations et la caractérisation des relations entre les éléments à composer. Elle cerne l'espace des possibilités en composition d'IHM.

Les relations de navigation/groupement entre espaces de travail peuvent être déduites de ces aspects. Par exemple, deux tâches composées en séquence (Meet) sont réifiées par des espaces de travail composés avec des relations de navigation.

Les composeurs peuvent être caractérisés par cette taxonomie. Par exemple, pour la version 2 (figure 1.5), le composeur permettant de réaliser la tâche « contact the doctor » compose, dans un onglet, l'IHM pour appeler un médecin et s'y déplacer. On peut décrire cette union comme une relation de composition de type (figure 4.9) : $\{Equal, EqualI\}$.

Les aspects permettent de raisonner sur la composition d'IHM et plus particulièrement sur l'ergonomie de l'IHM composite. Par exemple, s'il existe une redondance de la date du jour entre deux IHM à composer, et si ces dates sont de même temporalité sur des régions proches, alors

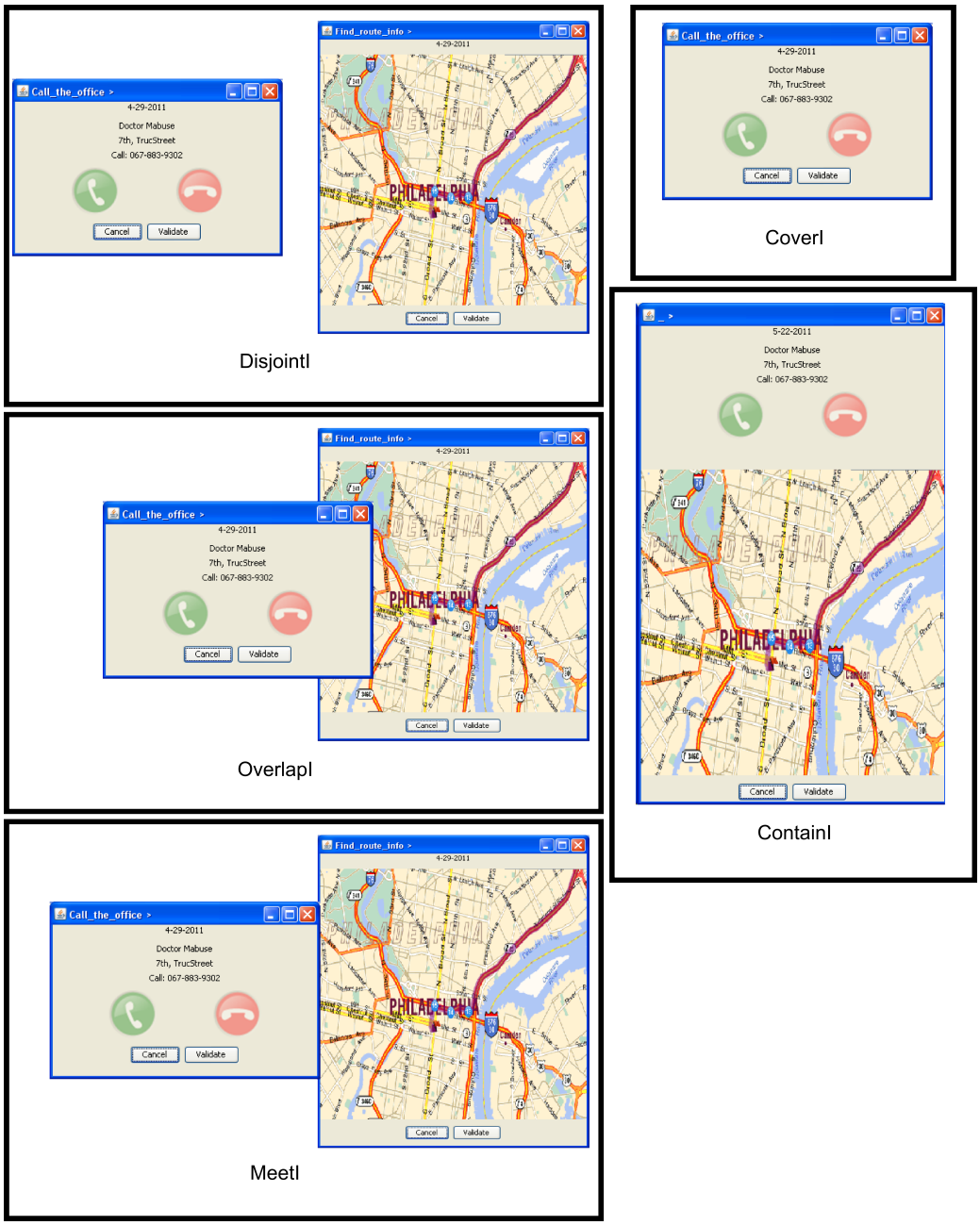


FIGURE 4.7 – *Exemples de relations spatiales entre canevas.*

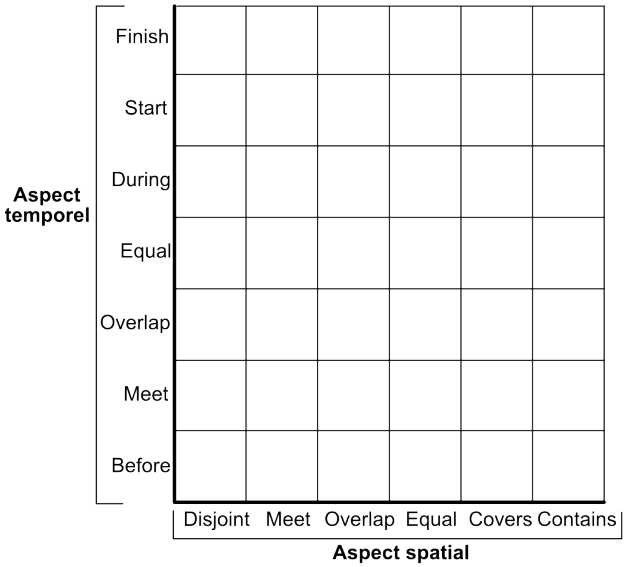


FIGURE 4.8 – *Taxonomie des composeurs.*

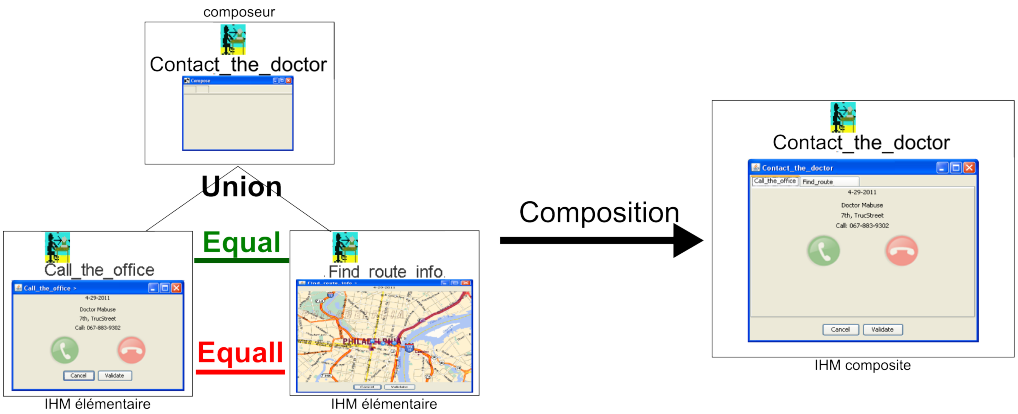


FIGURE 4.9 – *Caractérisation du composeur « contact the doctor » dans la version 2.*

elles sont redondantes et ne doivent être conservées que par volonté du concepteur de faire de l'insistance pour que l'utilisateur perçoive absolument cette information. Au contraire, si ces dates sont éloignées, elles ne sont pas redondantes et peuvent permettre à Victor d'avoir toujours la date du jour affichée sur ses différentes plates-formes.

4.3 Processus de composition d'IHM

Nous décrivons le processus de composition en deux parties. Dans la première, nous présentons la décomposition fonctionnelle de ce processus. Dans la deuxième, nous en dressons un espace problème.

4.3.1 Décomposition fonctionnelle

D'un point de vue fonctionnel (figure 4.10), le processus de composition d'IHM prend en entrée l'objectif de l'utilisateur et le contexte d'usage courant. Il produit en sortie une IHM composite dont la capitalisation est souhaitée par les utilisateurs (chapitre 3). Le calcul s'appuie sur les modèles d'IHM existantes.

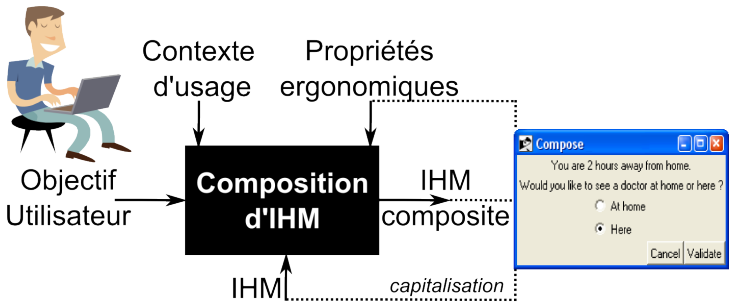


Figure 4.10 – *Entrées / sorties d'un système de composition d'IHM.*

Il s'effectue en deux temps :

1. **Calcul du schéma de composition d'IHM** par sélection d'un ensemble d'IHM à composer et de composeurs pour atteindre un état s' à partir de l'état courant s.
2. **Calcul de l'IHM composée** par application des composeurs sur les IHM à composer pour produire l'IHM composée.

Il est régulé par les propriétés ergonomiques à privilégier et sous-tendu par des fonctions d'évaluation. Les priorités peuvent être révisées dynamiquement selon l'écho favorable ou non des utilisateurs.

4.3.2 Caractérisation

Nous proposons une caractérisation du problème de la composition dynamique d'IHM fondée sur (Paternò et al. 2009) où l'on retrouve les niveaux d'abstraction en IHM et la prise en compte du moment de la composition. Cependant, nous l'étendons pour intégrer la dynamicité de la composition et en particulier l'objectif utilisateur, l'ergonomie et le schéma de composition. Notre espace problème se divise en trois classes de problèmes selon qu'ils relèvent du but, des IHM à composer ou des contraintes sur le calcul de composition.

But

Pour le but, il est nécessaire de caractériser :
- l'**Acteur** en charge de spécifier le but. L'acteur peut être l'*utilisateur* et/ou le *système*. Dans le cas d'étude, l'acteur est Victor mais on aurait pu imaginer que des capteurs de température détectent le mal être de Victor.
- Le **Schéma de composition** peut être exprimé explicitement par l'utilisateur (souvent en conséquence expert) ou calculé par le système à partir de l'objectif utilisateur pour atteindre le but. Par exemple, dans le cas d'étude, le schéma de composition des IHM à composer est calculé par le système. A l'inverse, dans les Mashups, l'utilisateur spécifie lui-même les IHM à composer. Si la composition est dynamique, le schéma de composition peut être calculé à partir des *IHM* et/ou des *NF*.

Éléments à composer

Pour les éléments à composer, il faut s'intéresser à leur nature :
- **Niveaux d'abstraction :** caractérisent les niveaux d'abstraction auxquels sont modélisés les éléments à composer : *Tâches*, *IUA*, *IUC* et *IUF*.
- **Couverture fonctionnelle :** identifie si les éléments à composer sont des *IHM* et/ou des *noyaux fonctionnels*.

Contraintes sur le calcul

Pour caractériser les contraintes sur le calcul de composition, il faut se concentrer sur :
- **Hétérogénéité :** informe si les éléments peuvent être décrits à des niveaux d'abstraction différents. La charge est alors laissée au sys-

tème d'aligner les éléments entre eux. Par exemple, un élément peut être décrit au niveau tâches et l'autre au niveau IUC, la question est de savoir à quel niveau les composer.
- **Niveaux d'abstraction :** caractérisent les niveaux d'abstraction auxquels sont effectués le calcul de composition de l'IHM composée par les composeurs.
- **Contexte :** informe si le contexte est *explicite* ou *implicite*. S'il est explicite, le calcul de composition peut s'appuyer sur ce contexte.
- **Ergonomie :** identifie si les critères d'ergonomie sont *explicites* ou *implicites*. S'ils sont explicites, le calcul de composition peut s'*appuyer* sur ces critères. Ces critères peuvent être évalués par l'utilisateur une fois que l'IHM composite est proposée ou par le système durant le calcul de l'IHM.
- **Moment :** caractérise le moment où a lieu la composition. Cette composition peut être effectuée à la *conception* et/ou à l'*exécution*. Par exemple, dans notre cas d'étude, l'IHM est composée spécialement pour Victor à l'exécution. Notons que le moment peut être hybride lorsque les IHM sont définies comme des variables instanciées à l'exécution selon le contexte d'usage et les propriétés ergonomiques.

Synthèse

La figure 4.11 synthétise cet espace problème de la composition d'IHM en trois classes de problèmes. Ces classes s'affinent en items représentant un problème à part entière.

4.4 Conclusion

La taxonomie des composeurs proposée permet de les caractériser et illustre la combinatoire importante du nombre de composeurs. Cette taxonomie a pour but d'aider le concepteur à composer des IHM grâce à une séparation des préoccupations en aspects.

La caractérisation des problèmes liés au processus de composition met en avant des classes de problèmes permettant de caractériser l'état de l'art en composition de systèmes interactifs, et en particulier d'IHM.

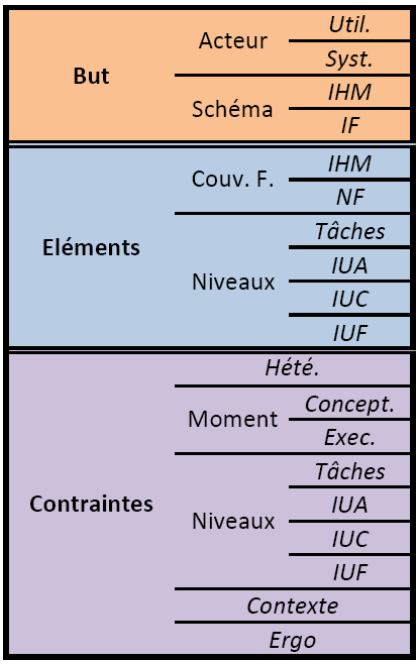


FIGURE 4.11 – *Espace problème de la composition d'IHM.*

État de l'art

5

Résumé

Ce chapitre dresse un état de l'art en composition de systèmes interactifs. La composition du noyau fonctionnel (NF) est étudiée en génie logiciel, et plus précisément dans les approches à composants et à services. La notion de service a permis de réduire le couplage des composants et ainsi d'identifier un besoin pour la composition dynamique : l'utilisation des techniques de l'Intelligence Artificielle. En particulier, la planification automatique est la voie la plus explorée. L'état de l'art en composition d'IHM est étudié selon les dimensions de l'espace problème. La composition d'IHM peut être effectuée dynamiquement. On constate cependant que les tâches utilisateur ne sont pas composées à partir de l'objectif utilisateur, mais déduites des tâches système. Ceci nous amène à orienter notre travail sur l'utilisation des techniques de l'Intelligence Artificielle pour la composition dynamique d'IHM, et plus précisément du modèle de tâches de l'IHM composite.

Publication associée : TSI'11 (Gabillon et al. 2011a)

Sommaire

Introduction

Ce chapitre se structure en deux temps : l'état de l'art en composition du noyau fonctionnel (NF) puis en composition d'IHM.

5.1 Composition du noyau fonctionnel

La composition du noyau fonctionnel est étudiée en génie logiciel. De nouvelles approches sont proposées, chaque nouveau paradigme essayant de pallier les limitations des précédents. Bien souvent, un nouveau paradigme ne remplace pas complètement les autres. Il essaie de réutiliser certains principes et d'en ajouter de nouveaux (Marin 2008). Ainsi, on a vu l'apparition des approches à objets (Taylor 1998), à composants (Szyperski et al. 2002) et dernièrement des approches à services (Papazoglou 2003) qui possèdent des points communs évidents.

L'*approche orientée objets* est fondée sur l'idée qu'un système interactif est composé d'un ensemble d'objets travaillant en collaboration. L'objet repose sur les notions de classe, d'héritage et de polymorphisme. La pratique a démontré que les systèmes construits par l'approche orientée objets sont difficilement composables. La principale cause, qui est aussi le principal inconvénient de cette approche, est le fait que la composition d'un système doit être réalisée à un niveau d'abstraction trop bas, celui des classes.

La notion de composition se développe grâce au paradigme de composants puis de services.

5.1.1 Composition de composants

Il n'y a pas actuellement de consensus sur la notion de composant. La définition généralement retenue est celle de Szyperski (Szyperski et al. 2002) :

> « A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties. »

L'*approche à composants* est fondée sur l'idée qu'un système est composé de briques logicielles préfabriquées appelées des composants. Cette approche se place à un niveau d'abstraction supérieur par rapport aux

objets et promeut en conséquence l'emploi de la composition comme mécanisme d'assemblage des composants. Un composant représente une unité de déploiement qui encapsule son implémentation et sa description. Pour permettre son utilisation, la description d'un composant précise quelle est la fonctionnalité qu'il réalise, mais aussi ses dépendances fonctionnelles.

La composition de composants se fonde sur la notion de *langage de description d'architecture* (ADL) qui permet de décrire les dépendances entre des composants. Avec cette technique, la logique de composition est disséminée dans les constituants du système. Ces derniers connaissent les relations qui les unissent aux autres. Des modèles à composants traditionnels, comme EJB (De Michiel et al.), CCM (components 1999), Fractal (Bruneton et al.), définissent la structure des composants et la manière de les composer.

Comme exemple de référence, Fractal définit un langage de description d'architecture permettant de décrire l'architecture d'une application à base de composants. Il sépare les aspects fonctionnels et non fonctionnels en introduisant la notion de conteneur. De plus, Fractal supporte la composition hiérarchique de composants. Le système conforme à ce modèle fournit des mécanismes permettant de composer des composants. Il est ainsi possible de modifier les propriétés et les liaisons des composants ainsi que de changer le contenu des composites. Enfin, Fractal propose des mécanismes d'introspection, ce qui permet d'observer et modifier les éléments administrés.

Les modèles à composants comme Fractal sont peu modifiables à l'exécution, c'est-à-dire qu'une fois la composition de composants effectuée, les changements dans l'architecture sont difficiles. La principale cause est le fort couplage des composants liés par des dépendances. Les approches à services essaient d'éliminer les dépendances entre les différents éléments d'une application et de permettre, au contraire, un faible couplage.

5.1.2 Composition de services

Il n'y a pas actuellement de consensus sur la notion de service. Nous utiliserons la définition issue de (Papazoglou 2003) :

> « Services are self-describing, platform agnostic computational elements. »

L'approche à services (Papazoglou 2003) est basée sur l'idée qu'un système peut être réalisé par composition de services. Les services sont configurés à l'exécution pour répondre à des besoins spécifiques et sont ensuite invoqués. Un service est une entité logicielle qui est décrite, publiée, découverte et utilisée.

Plusieurs implémentations des services existent comme CORBA (cor 1995), OSGI (The OSGi Alliance 2005) ou bien encore les Web Services.

Bucchiarone et Gnesi (2006) proposent une taxonomie de composition de services Web présentée par la figure 5.1. Un processus de composition de services web peut être statique ou dynamique. Dans la composition statique, un processus de composition de services est défini à partir d'un plan d'exécution de services tandis que dans la composition dynamique, il est engendré à partir d'un objectif utilisateur et un contexte, i.e., le schéma de composition en termes de services doit être calculé.

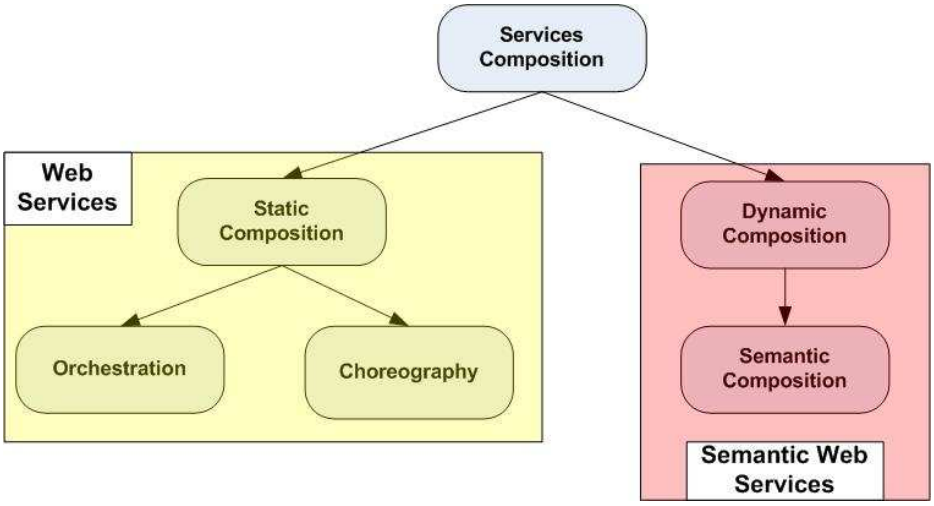


FIGURE 5.1 – *Taxonomie de la composition de services Web (Bucchiarone et Gnesi 2006).*

Composition statique

La composition statique de services est classifiée en se fondant sur les interactions entre les services. Les deux approches pour la composition statique de services sont :

– *Orchestration* : compose des services existants de manière centralisée par un coordinateur central (un orchestrateur) qui est responsable d'appeler et de faire interagir des services. L'exécution d'une orchestration de services est réalisée et contrôlée par le coordinateur

selon son plan d'exécution.

Comme exemple de référence, BPEL4WS (Business Process Execution Language for Web Services) (Curbera et al. 2003) définit un modèle et une grammaire qui décrivent les interactions entre le processus et leurs partenaires en utilisant les interfaces des services web. Elle définit aussi les états et la logique de composition. Un document BPEL4WS utilise XML pour décrire des aspects d'un processus métier comme des partenaires, des actions atomiques, des variables. Le résultat de l'utilisation de BPEL4WS pour modéliser un processus métier exécutable est un nouveau service Web composé par des services existants.

– *Chorégraphie* : compose des services existants de manière décentralisée sous la forme de tâches complexes par l'intermédiaire de la définition de conversations qui devraient être effectuées par chaque service participant. Une conversation décrit l'échange de messages ordonnés entre deux services selon un protocole.

  Comme exemple de référence, Web Service Choreography Interface (WSCI) (Kavantzas 2004) est un langage de chorégraphie qui décrit les messages échangés entre les services web participant. Il définit le flot de messages échangés par un service web décrivant son comportement. En spécifiant les dépendances temporelles et logiques parmi les échanges de messages, on peut décrire un service de telle façon que d'autres services Web peuvent l'appeler.

Les systèmes d'orchestration et de chorégraphie permettent de définir correctement une composition de services. Un des principaux bénéfices de la composition de services est le couplage faible entre les différents services réunis dans un même système. Les dépendances fonctionnelles exprimées dans les approches à composants sont éliminées dans l'approche à services. Cependant, ces travaux ne considèrent pas les environnements dynamiques dans lesquels les buts de l'utilisateur et la disponibilité des services peuvent varier.

Composition dynamique

Le défi de la composition dynamique est de composer automatiquement des services pour atteindre un but de l'utilisateur en contexte. L'automatisation dynamique de la composition nécessite l'ajout de la description sémantique à la description des services. Appelé DAML-S dans ses premières versions (Ankolekar et al. 2002), le langage OWL-S

(Martin et al. 2004) a pour objectif d'ajouter des descriptions sémantiques aux services (en plus de leur description syntaxique WSDL (Chinnici et al. 2004)) pour permettre de découvrir, invoquer, composer et gérer les services automatiquement.

L'hypothèse sur laquelle se fonde la composition dynamique est que chaque service est considéré comme une action décrite par sa signature de méthodes, ses préconditions, ses postconditions et ses effets. Cette hypothèse tire sa légitimité des raisons suivantes (Hanh 2009) :

- un service est un logiciel autonome qui prend des données d'entrée et produit des données de sortie. Ainsi les préconditions et les postconditions sont évaluées pour déclencher son exécution et valider sa terminaison.
- un service change l'état de la composition après son exécution. Ainsi l'état demandé pour l'exécution d'un service est la précondition, et les nouveaux états produits après l'exécution sont les effets de son exécution.

Nous présentons la composition dynamique par protocole d'interaction et la composition par les techniques d'Intelligence Artificielle. Cette partie est inspirée de (Hanh 2009).

Protocole d'interaction :

Hassen et al. (2008) utilisent une machine à états finis pour représenter les échanges chorégraphiques de messages, c'est-à-dire le protocole d'interactions de services web. Les états représentent les différentes phases par lesquelles un service peut passer durant son interaction avec un demandeur (service). Les transitions sont déclenchées par des messages envoyés par un demandeur au fournisseur (service). Chaque transition est étiquetée par un nom de message.

Hassen et al. (2008) considèrent la composition dynamique de services comme un ensemble de services atomiques. Il s'intéresse à la façon dont ils communiquent entre eux. Ils adaptent les méthodes d'orchestration et de chorégraphie afin de les rendre dynamiques. Cependant, il est en conséquence très difficile d'avoir une vision détaillée du système composé.

Techniques d'Intelligence Artificielle :

Une composition dynamique de services en se fondant sur les techniques d'Intelligence Artificielle (IA) consiste à planifier (semi)automatiquement un *plan d'exécution* (schéma de composition)

de services. Rao et Su (2005) identifient différentes techniques d'IA actuellement appliquées à la composition de services :

- **Calcul situationnel (situation calculus) :** McIlraith et Son (2002) adaptent et étendent le langage Golog pour la construction automatique de services. Golog est un langage de programmation logique qui permet de faire du calcul situationnel, i.e., langage logique qui sert à représenter des changements ou évolutions en terme de situations, d'actions et d'objets. Les auteurs abordent le problème de la composition de services à travers des procédures génériques de haut niveau et la personnalisation des contraintes. Golog est un formalisme naturel pour représenter et raisonner sur le problème de la composition. Le problème est abordé de la façon suivante : le but de l'utilisateur et les contraintes des services sont représentés en terme de prédicats du premier ordre dans le langage de calcul situationnel. Les services sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et sont instanciés à l'exécution.

- **Planification PDDL :** Klusch et al. (2005) ont présenté un système de composition de services OWL-S, appelé OWLS-Xplan qui permet de composer des services. OWLS-Xplan convertit des descriptions de service en OWL-S en des descriptions de problème et de domaine équivalentes qui sont décrites par le langage de description du domaine de planification PDDL (McDermott et al. 1998). OWLS-Xplan utilise un planificateur Xplan pour produire un plan de composition de services qui satisfait un but donné.

- **Planification HTN :** un réseau hiérarchique de tâches se compose des tâches primitives et non primitives. Une tâche primitive est une action tandis qu'une tâche non primitive est décomposée par des tâches primitives et non primitives. La planification HTN consiste à réaliser une liste de tâches ordonnées (réseau de tâches). Wu et al. (2003) ont utilisé le planificateur SHOP2 (Ilghami et Nau 2003) pour composer dynamiquement des services Web dont les signatures, les préconditions et les effets sont décrits en OWL-S.

- **Rules based planning :** Medjahed (2004) présente une technique pour générer des services composites à partir d'une description déclarative de haut niveau. La méthode utilise des règles de composabilité pour déterminer si deux services sont composables. L'approche de composition se compose de quatre phases. Tout d'abord, la phase de spécification de haut niveau permet la des-

cription des compositions désirées en utilisant un langage appelé Composite Specification Language Service (CSSL). Deuxièmement, la phase de jumelage utilise les règles de composabilité pour générer des plans de composition qui sont conformes aux spécifications du demandeur de service. La troisième phase est la phase de sélection. Si plus d'un plan est généré dans la phase de sélection, alors le demandeur de service sélectionne un plan basé sur la qualité de la composition (rang, coût, ...). La phase finale est la phase de génération. Une description détaillée du service composite est automatiquement générée et présentée au demandeur de service.

– **Theorem proving :** Waldinger (2001) propose une approche pour la composition dynamique de services par preuve de théorèmes. Cette approche est fondée sur la déduction automatique et la synthèse de programmes. Les services disponibles et le but de l'utilisateur sont traduits dans un langage du premier ordre. Puis des preuves sont produites à partir d'un *prouveur de théorèmes*. La composition est obtenue à partir de preuves particulières. Dans le même genre, Rao et al. (2003) ont introduit une méthode de composition automatique de services sémantiques en utilisant des preuves de théorèmes logiques linéaires (Linear Logic Theorem Proving).

5.1.3 Discussion

Le principal avantage de la composition dynamique est de composer automatiquement un schéma de composition, réduisant en conséquence au maximum l'intervention humaine.

Cependant, Hanh (2009) identifie des freins à la réalisation de la composition dynamique de services :

– la construction d'une base de connaissance des fonctionnalités des services est coûteuse ;

– les planificateurs comme XPlan, Shop2 se fondent sur l'hypothèse d'un ensemble fermé d'actions ;

– une longue latence du calcul d'un plan d'exécution approprié peut ne pas être acceptable pour les utilisateurs ;

– la génération d'un nouveau plan peut ne pas être totalement conforme aux besoins exprimés. Par exemple, Xplan ou Shop2

n'expriment pas le choix ou la séquence non ordonnée de services ;

– La recomposition dynamique d'une composition remet en cause totalement le précédent plan. Trois solutions ont été proposées par Van der Aalst et al. (1999) pour résoudre cette difficulté :
  • Annulation de toutes les instances en cours de la composition modifiée : cette solution n'est pas acceptée par certaines applications comme, par exemple, lorsque l'activité de paiement est déjà réalisée.
  • Hypothèse de cohérence du changement : cette solution suppose que les changements ne vont pas engendrer d'incohérence dans le plan déjà calculé.
  • Modification du plan d'exécution des instances en cours d'exécution : cette solution est impossible pour la plupart des moteurs de composition de services. Quelques systèmes comme eFlow (Casati et al. 2000) permettent de modifier manuellement le plan d'exécution au moment d'exécution des instances par l'intervention du concepteur.

L'automatisation complète de la composition dynamique est toujours un sujet ouvert qui nécessite l'intervention de techniques de l'IA. Parmi ces techniques, Rao et Su (2005) affirment que la planification est la voie la plus prometteuse. Elle est en tous cas la voie la plus empruntée. La raison principale est sa capacité à résoudre un des problèmes principaux : la différence entre les concepts que les utilisateurs utilisent et les données que les ordinateurs interprètent. En effet, cet obstacle peut être surmonté en employant des technologies de Web sémantique comme OWL-S.

La composition dynamique de services se concentre sur les interactions machine-machine. Pourtant, l'interaction homme-machine dans ces services joue un rôle important : une bonne IHM pourrait rendre un service composite acceptable voir désirable.

5.2 Composition d'IHM

Actuellement, différentes approches développées en parallèle s'intéressent à la composition d'IHM plus ou moins directement. Les travaux en génération automatique d'IHM (Mori et al. 2002, Myers 2009, Clerckx et al. 2005), en adaptation d'IHM (Tan et al. 2004, Stürzlinger et al. 2006) ou les Mashups (Lin et al. 2009, Ennals et Gay 2007) en sont des exemples. On peut organiser ces approches en deux catégories selon le

moment où a lieu la composition : à la conception ou à l'exécution.

5.2.1 Composition à la conception

Dans cette partie, nous présentons les travaux étudiant la composition d'IHM à la conception. Nous les organisons selon le niveau d'abstraction auquel ils opèrent.

Composition au niveau IUA et IUC : ComposiXML et Amusing

ComposiXML (Lepreux et Vanderdonckt 2006, Lepreux et al. 2007) permet la composition d'IHM graphiques par le concepteur. La composition d'arbres représentant les IHM à composer est effectuée au niveau IUC (Lepreux et Vanderdonckt 2006) puis étendue au niveau IUA (Lepreux et al. 2007). Les opérateurs disponibles dans ComposiXML sont des opérateurs unaires ou binaires (figure 5.2). Les opérateurs unaires sont les suivants :

- *Set* ou *Selection* sélectionne un ensemble d'éléments d'une IHM. *Set* effectue la sélection à partir d'un identifiant représentant un sous-arbre d'une IHM. Par exemple, $Set(T1)$ signifie que l'arbre $T1$ est sélectionné. *Selection* effectue cette sélection à partir d'un ensemble d'éléments. Par exemple, $Selection(\{I1, I2\}, T1)$ signifie que les interacteurs $I1$ et $I2$ sont sélectionnés dans l'IHM $T1$.
- *Complementary* sélectionne l'IHM en retirant les éléments sélectionnés. Par exemple, $Complementary(\{I1, I2\}, T1)$ signifie que T_1 est sélectionné en supprimant les interacteurs $I1$ et $I2$.
- *Cut* supprime un sous-arbre d'une IHM en spécifiant un nœud. Ce nœud est le sommet du sous-arbre à supprimer. Par exemple, $Cut(t1, T2)$ supprime le sous-arbre de racine $t1$ dans $T2$.
- à l'inverse, *Projection* extrait un sous-arbre de l'IHM à partir d'un nœud. Ce nœud est le sommet du sous-arbre à sélectionner. Par exemple, $Projection(t1, T2)$ extrait le sous-arbre de racine $t1$ de $T2$.

Les opérateurs binaires de composition sont les suivants :

- *Left/Right difference* supprime une IHM d'une autre. Par exemple, $Left\ difference(T1, T2) = Right\ difference(T2, T1)$ sélectionne $T1$ en supprimant les éléments qui appartiennent aussi à $T2$.
- *Normal union* réalise l'union de deux IHM en fusionnant les éléments communs. Par exemple, si la date du jour est présente dans les deux IHM à composer, cette date est supprimée dans une des deux pour que l'IHM composite ne comporte pas deux fois cette date.

- *Unique union* réalise l'union de deux IHM en supprimant l'ensemble des éléments communs. Par exemple, si la date du jour est présente dans les deux IHM à composer, l'IHM composée ne comportera pas cette date.
- *Fusion* réalise *Normal union* sans supprimer les doublons.
- *Intersection* réalise l'intersection de deux IHM.
- *Join* concatène un ensemble de nœuds de deux arbres en fonction des nœuds communs.

ComposiXML dispose aussi d'opérateurs qui permettent de comparer deux IHM :

- *Similarity* compare la structure des deux IHM sans se préoccuper des données dans les nœuds.
- *Equivalence* compare la structure et les données de deux IHM.
- *Subsumption* vérifie si une IHM est incluse dans une autre. Par exemple, $Subsumption(T1, T2)$ est vraie si et seulement si il existe un sous-arbre $T1$ dans $T2$.

Amusing (Pinna-Dery et al. 2003) compose des IHM décrites au niveau IUA. Les IHM sont décrites en SunML (Pinna-Dery et al. 2003). Amusing propose un moteur de rendu qui permet de réifier la description de l'IHM pour obtenir une IUF décrite en Swing. Le contexte d'usage est implicite. La composition est réalisée par les opérateurs de composition suivants :

- *add* inclut une IHM dans une autre.
- *select* permet d'extraire un sous arbre de l'IHM à partir d'une racine.
- *union* permet une fusion de deux IHM en supprimant les informations redondantes. L'union fusionne des arbres en unifiant les éléments sémantiquement identiques (figure 5.3).
- *intersection*, à l'inverse, permet de récupérer les parties communes de deux IHM.
- *difference* permet de supprimer d'une IHM une partie commune avec une autre.
- *substitution* permet de remplacer un sous-arbre (soit un nœud, soit une feuille) par un autre dans une IHM.

ComposiXML et Amusing sont des travaux de référence en composition d'IHM. Ils ont été les premiers à étudier la composition et plus particulièrement les opérateurs de composition d'IHM au niveau IUA. Ils portent sur une composition redondante et équivalente. Cependant, on constate que ces opérateurs ne traitent pas le niveau tâches. En conséquence, l'aspect temporel n'est pas considéré. La sémantique est exprimée par des éléments supposés équivalents par le concepteur. Ce dernier est ainsi l'acteur du processus et spécifie lui-même le schéma

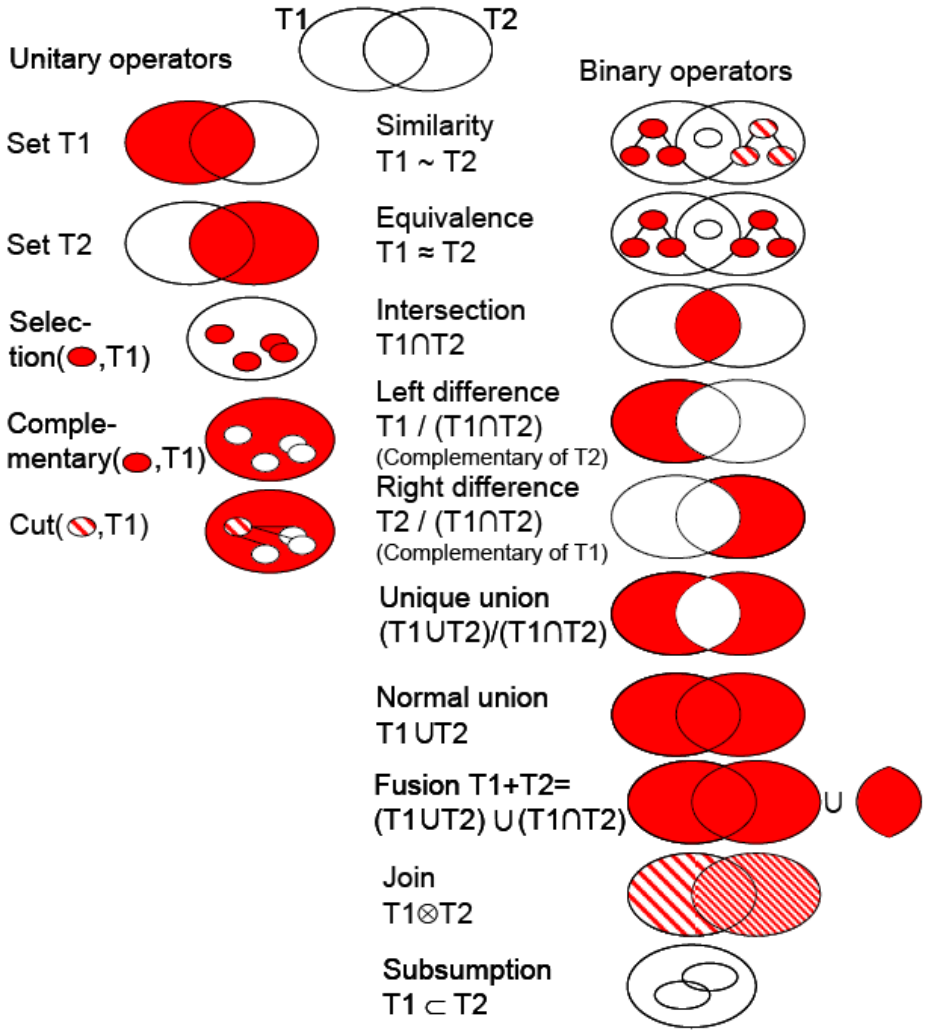


FIGURE 5.2 – *Opérateurs dans ComposiXML (Lepreux et Vanderdonckt 2006).*

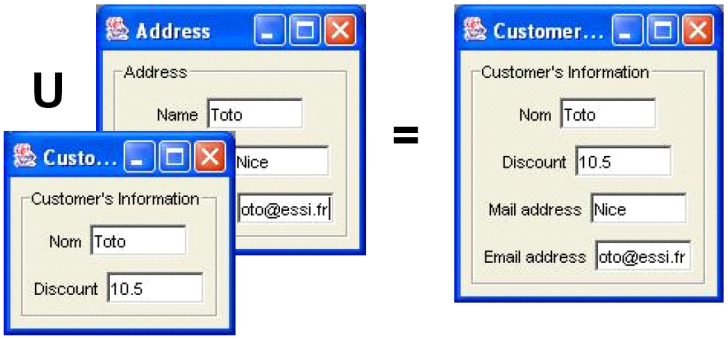


FIGURE 5.3 – *Opérateur d'union dans Amusing (Pinna-Dery et al. 2003).*

de composition, i.e, les IHM et les opérateurs utilisés sans expliciter le contexte. Il est en conséquence garant de la validité et de la qualité ergonomique de l'IHM composée.

Composition du modèle de tâches (Lewandowski et al. 2007)

Lewandowski et al. (2007) se concentrent sur la composition complémentaire et redondante au niveau tâches. Le concepteur est garant de la qualité et de la validité de la conception. Les auteurs proposent une approche pour réaliser l'union d'arbres de tâches préexistants exprimés en XML. Par exemple, ils réalisent une union entre un arbre de tâches d'un chat et celui d'un tableau blanc partagé entre plusieurs utilisateurs. Ces modèles de tâches comportent des sous-tâches identifiées comme similaires par le concepteur (figure 5.4). Pour le chat, l'utilisateur doit se connecter pour pouvoir discuter. Pour le tableau blanc, l'utilisateur doit s'identifier pour écrire sur le tableau et partager ces informations. Ces tâches similaires expriment des relations de redondance et de complémentarité. Par exemple, les tâches « Connect » et « Identification » sont redondantes, les tâches « set login » et « set name » sont complémentaires. Les auteurs proposent un algorithme pour implémenter cet opérateur d'union pour produire un modèle de tâches composé (figure 5.5). L'algorithme crée d'abord un nouveau sous arbre contenant la composition redondante des deux tâches, ici « Connect » et « Identification ». Cette composition est possible car les deux tâches redondantes ont des structures identiques. L'algorithme importe l'une d'elle, dans le nouvel arbre, et ajoute de manière ordonnée les sous-tâches non redondantes apparaissant dans le second. Par exemple, les tâches « set password » sont redondantes, une seule est conservée.

L'acteur du processus de composition est le concepteur. Le calcul ne prend pas en compte le contexte d'usage et les propriétés ergonomiques. Il fait, par ailleurs, une hypothèse forte en supposant les modèles de tâches à composer de même structure.

Composition dirigée par le NF (Joffroy 2011)

Joffroy (2011) propose une approche pour composer des IHM en étant dirigé par le NF. Il facilite le travail du développeur en lui permettant de réutiliser l'intégralité des systèmes à composer, c'est-à-dire, le NF, l'IHM, et les interactions entre eux. Il assure l'indépendance technologique par l'utilisation d'un niveau d'abstraction. Ce niveau permet de représenter l'ensemble des IHM et du NF tout en ne conservant que ce

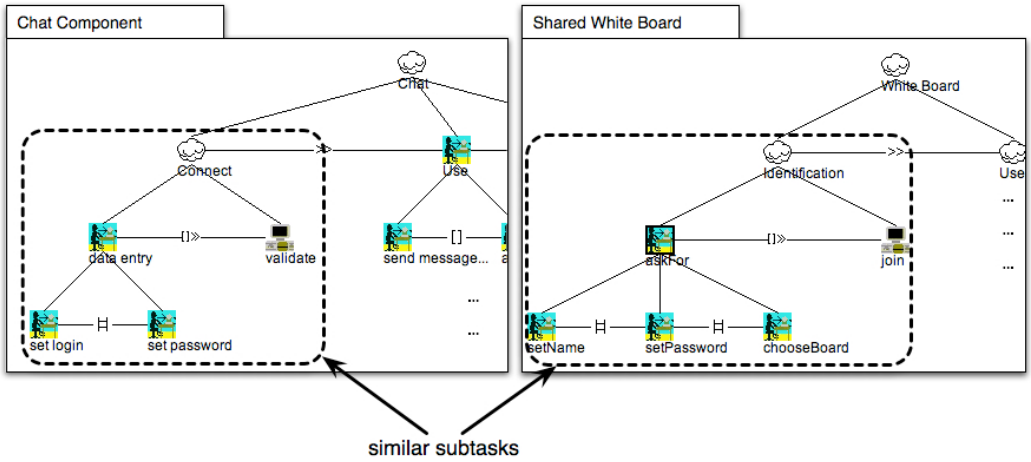


FIGURE 5.4 – *Tâches similaires entre le chat et le tableau blanc (Lewandowski et al. 2007).*

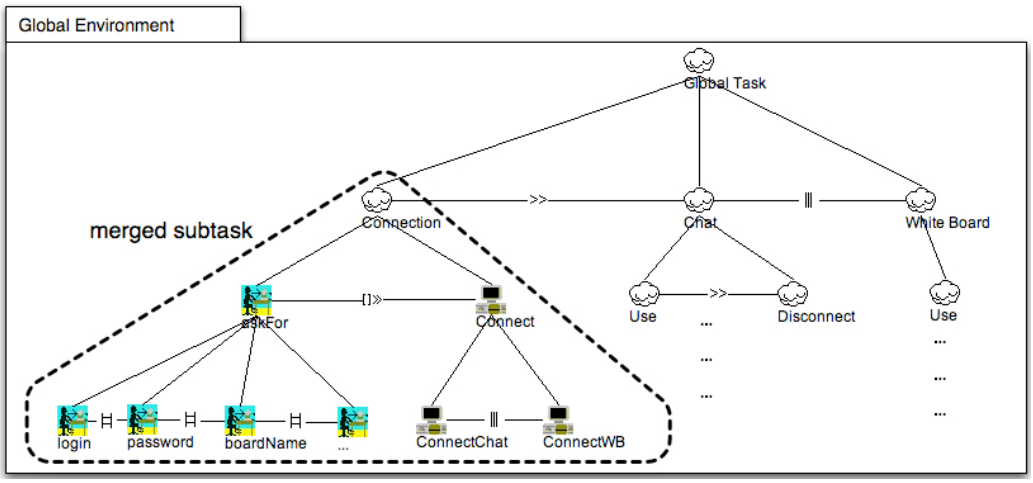


FIGURE 5.5 – *Extrait du modèle de tâches composé par l'algorithme (Lewandowski et al. 2007).*

qui est nécessaires à la réalisation de la composition. L'ensemble des informations nécessaire à la composition est décrit au sein du méta-modèle AliasComponent. Ce méta-modèle s'inspire des approches à composants en représentant le noyau fonctionnel et l'IHM comme des composants. La composition du NF est décrite par un composant composite. La composition d'IHM est effectuée au travers de deux types de liens : événements et données. Les liens d'événements décrivent le déclenchement d'opérations. Les liens de données décrivent les échanges de données entre l'IHM et le NF. Joffroy (2011) montre qu'il était techniquement possible de réaliser la composition dirigée par la composition fonctionnelle. Il guide le concepteur dans le développement d'IHM composée sans prendre en compte le contexte et l'ergonomie.

5.2.2 Composition à l'exécution

Nous présentons en quatre temps, les approches qui se concentrent sur :

- la génération automatique d'IHM ;
- les Mashups ;
- l'adaptation des IHM ;
- et la composition d'IHM couplée à la composition dynamique de services.

Génération d'IHM

Les travaux s'intéressant à la génération d'IHM (Mori et al. 2002, Myers 2009) sont une forme de composition même si ce terme et les opérateurs de composition ne sont pas explicites. Depuis les années 90, ces travaux ont porté sur la génération d'IHM à partir d'un modèle de tâches donné. Les transformations de modèles étaient « câblées en dur » dans le code des générateurs. Elles n'étaient ni modifiables, ni extensibles (Johnson et al. 1993), aboutissant en conséquence à des IHM de faible qualité, généralement graphiques, mono-écran, bien loin des IHM modernes (dites « post-WIMP »), ne laissant aucune place à la créativité.

Un saut est franchi dans les années 2004 avec le développement de l'Ingénierie Dirigée par les Modèles (IDM) : les transformations de modèles deviennent des modèles. Dès lors, les transformations sont capitalisables et traçables au fil des évolutions, à la conception comme à l'exécution. De nouvelles possibilités s'ouvrent alors aux modèles du processus de conception (Myers 2009) en supposant toujours les tâches utilisateur fixées par le concepteur.

SUPPLE (Gajos et Weld 2004) sert ici d'exemple de référence. Il génère des IHM sur différentes plates-formes. SUPPLE utilise un modèle fonctionnel de l'application représenté sous la forme d'un arbre. Cet arbre exprime des relations sémantiques entre des objets typés pour lesquels il convient de produire une représentation concrète. Par exemple, le niveau de lumière est représenté par un entier compris entre 0 et 10. La génération d'une IHM est ainsi vue comme un problème d'optimisation de ce modèle fonctionnel. Un algorithme de rendu génère une IHM en transformant chaque élément de l'arbre en un interacteur. Le type d'interacteur est sélectionné en fonction du type de données manipulées dans le modèle de décomposition fonctionnelle. Par exemple un entier de 1 à 10 donnera lieu à une liste déroulante. Le booléen quand à lui sera représenté par une case à cocher.

Dans SUPPLE, la procédure pour répondre à l'objectif utilisateur est connue et exprimée sous la forme d'un arbre. Cet arbre est le pendant fonctionnel du modèle de tâches. Gajos et Weld (2004) font donc l'hypothèse que le schéma de composition d'IHM est fixé par le concepteur. Cette approche convient pour des applications simples comme le contrôle des lumières et des projecteurs vidéo d'une pièce : les types d'objets sont simples et l'arbre guide le placement des widgets dans l'IHM concrète. SUPPLE++ fait suite à ces travaux pour la génération d'IHM pour mal-voyants (Gajos et al. 2007).

Comme le remarque Nichols, la génération automatique d'IHM pour des plates-formes différentes peut donner lieu à des IHM incohérentes. UNIFORM (Using Novel Interfaces For Operating Remotes that Match) vise à réduire ces incohérences uniquement pour le cas des télécommandes (Nichols et al. 2006).

Les travaux en génération d'IHM étudient la composition indirectement par les relations exprimées entre les modèles d'une IHM à tous niveaux d'abstraction. Il génère une IHM à partir de la composition de composants de faible granularité ce qui entraîne un fort couplage des éléments composés. La composition est ainsi longitudinale, ancrée à tous niveaux d'abstraction.

Mashups

Les Mashups (Fujima et al. 2004, Lin et al. 2009) permettent une composition d'applications dirigée par les données. Les IHM sont gérées au niveau IUF. La composition est placée sous le contrôle de l'utilisateur

final pour la sélection des données et des services à la volée.

Comme exemple de référence des Mashups, iGoogle[1] propose à l'utilisateur de créer son propre environnement de travail accessible par navigateur web. Il permet à l'utilisateur de modifier à l'exécution cet environnement en ajoutant des IHM proposant des services. Par exemple, la figure 5.6 représente un environnement iGoogle qui compose l'affichage de l'heure, du temps et permet de lancer une recherche sur Wikipedia. La composition se limite à une juxtaposition ($Union = \{Equals, MeetI\}$) des IHM sélectionnées par l'utilisateur. Cette composition est équivalente à une composition d'IHM au niveau IUC où le schéma de composition est exprimé par l'utilisateur. Ce dernier a la possibilité de (re/de)composer ses services et donc son IHM.

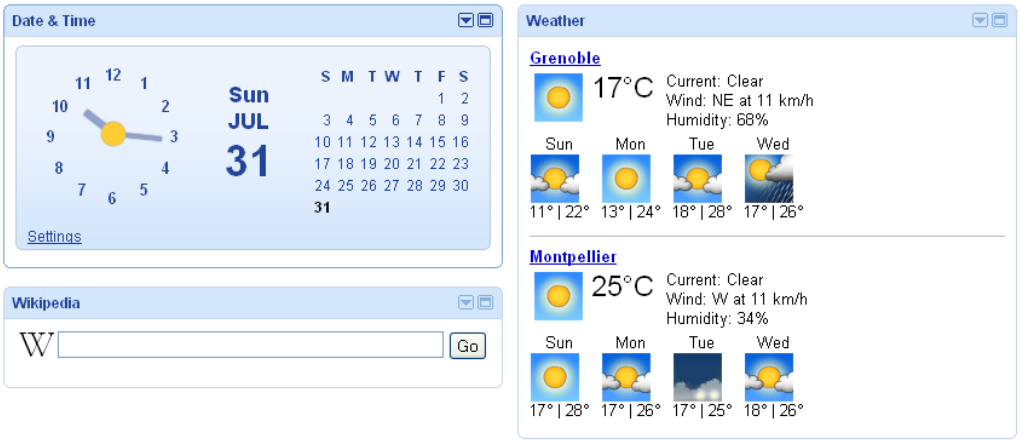


FIGURE 5.6 – *IHM composées par juxtaposition par iGoogle.*

Les Mashup sont centrés utilisateur. La composition est réalisée à l'exécution par l'utilisateur qui exprime ainsi lui-même le schéma de composition de l'IHM qu'il souhaite obtenir. Les composeurs sont donc « explicites » pour l'utilisateur dans le sens où ce dernier doit comprendre intuitivement comment composer et ce qu'il compose. Par exemple, dans iGoogle, l'utilisateur apprend rapidement que les services qu'il sélectionne seront juxtaposés sur sa page. Dès lors, les dépendances entre services sont anticipées à bas niveau d'abstraction par le concepteur qui garantit une composition cohérente pour l'utilisateur.

1. `http://www.google.com/ig`

Adaptation des IHM

Les travaux sur l'adaptation s'intéressent au contexte d'usage et à la distribution des IHM sur un ensemble de plates-formes (Stürzlinger et al. 2006, Nichols et al. 2008, Tan et al. 2004). Une IHM adaptable a, par exemple, la capacité de se remodeler pour s'accommoder des caractéristiques de cette nouvelle plate-forme de façon à ce que l'utilisateur puisse poursuive sa tâche avec confort, efficacité et sécurité. Lors de ce processus d'adaptation, l'IHM doit se re/décomposer. Par exemple, l'IHM pour appeler un médecin doit être détachée de l'IHM composée pour migrer sur le Smartphone allumé par Victor. L'utilisateur peut sélectionner manuellement les parties de l'IHM à déplacer (tailoring, (Wulf et al. 2008)). Au contraire, la sélection peut être faite par le système (Paternò et al. 2008) pour s'adapter aux modifications du contexte d'usage.

Comme exemple de référence, Grolaux et al. (2005) s'intéressent à la décomposition d'une IHM et la recomposition des sous-parties ensemble. Leur objectif est de proposer une IHM détachable. Une IHM détachable est une IHM dont les morceaux peuvent être détachés ($Difference$) d'une plate-forme à l'exécution, pour migrer et s'adapter à une autre. Par exemple, la barre de menu et la palette graphique du logiciel paint peuvent être détachées de la TabletPC pour migrer et être attachées ($Union = \{Equals, MeetI\}$) ensemble sur un PC (figure 5.7). L'utilisateur est l'acteur de ce processus. Les opérateurs de composition sont implicites mais cette composition se porte sur la juxtaposition et décomposition d'IHM.

Du point de vue de la composition, les travaux sur l'adaptation se concentrent sur la (de/re)composition d'une IHM supposée connue. La plupart du temps, la composition est laissée sous le contrôle de l'utilisateur ce qui lui permet de spécifier lui-même les compositions qu'il souhaite. Le contrôle peut être assuré via une méta-IHM (Balme et al. 2005).

Composition de services Web couplée à la composition d'IHM

La composition d'IHM couplée à la composition dynamique de services a pour objectif de pallier le manque de prise en compte de l'interaction homme-machine en composition de services en utilisant le savoir faire en IHM. L'idée est de profiter du plan d'exécution calculé dynamiquement à partir des services disponibles, pour composer ensuite les IHM de ces services.

En particulier, Feldmann et al. (2010) calculent l'IHM composée à par-

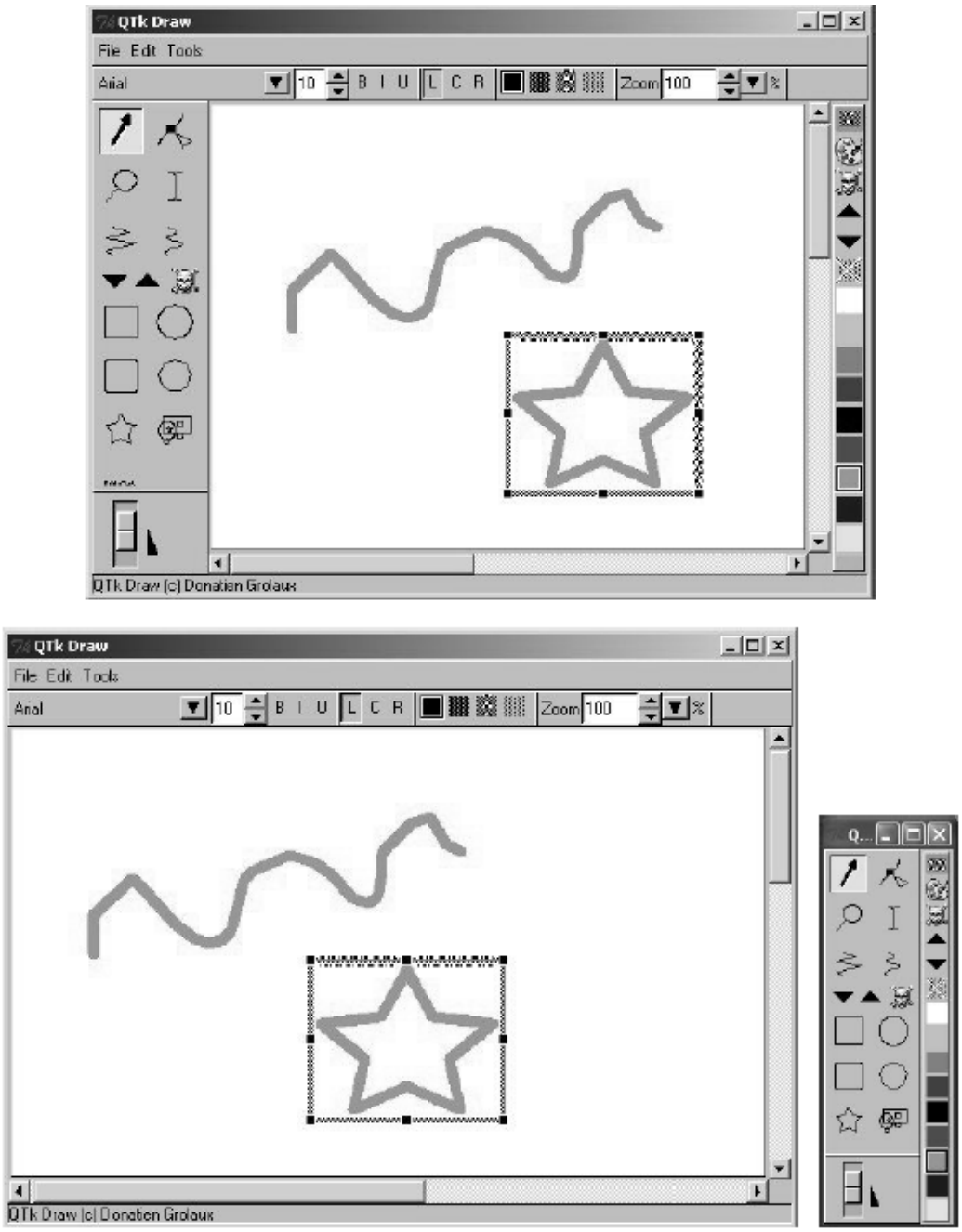


FIGURE 5.7 – *Barre de menu et palette graphique détachées d'une TabletPC et attachées ensemble sur un PC (Grolaux et al. 2005).*

tir des tâches système assemblées dynamiquement. Les tâches système sont le support d'une *opération d'un service web annoté*. Une opération d'un service web annoté permet d'associer les tâches système à des tâches d'interaction avec l'utilisateur. Par exemple, la tâche système qui vérifie qu'un login est annoté (check login) nécessite que ce login soit saisi à l'aide d'une tâche d'interaction (login) (figure 5.8). Les tâches utilisateur sont donc déduites à partir des tâches système annotées. Une fois les tâches utilisateur ajoutées aux tâches système, elles peuvent être groupées par un seul opérateur de composition *normal union* de CompoSiXML. Cet opérateur compose les IHM en les réunissant. Par exemple, pour réserver un vol, l'union est effectuée sur les tâches « login succes » et « enter search criteria » (figure 5.9).

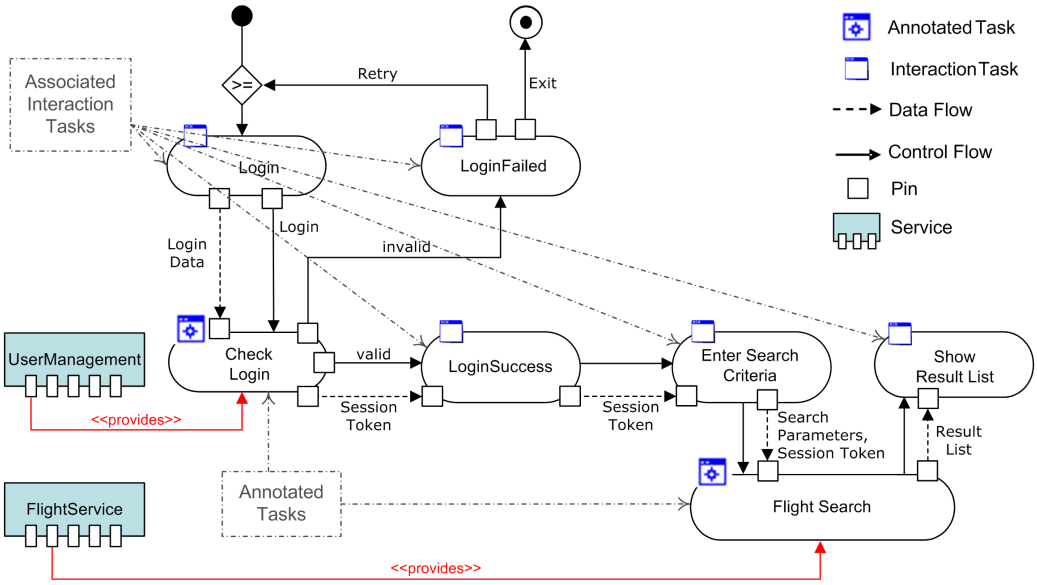


FIGURE 5.8 – *Tâches utilisateur déduites des tâches système pour composer une IHM pour réserver un vol (Feldmann et al. 2010).*

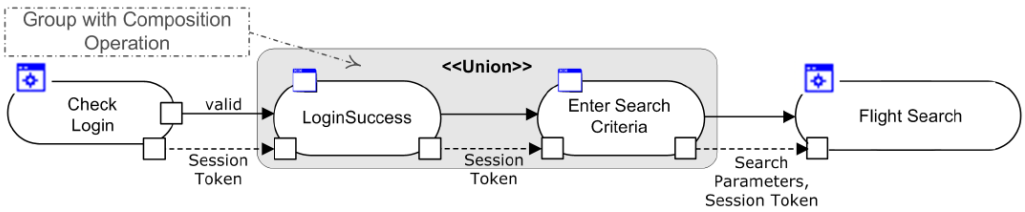


FIGURE 5.9 – *Union des tâches « login succes » et « enter search criteria » (Feldmann et al. 2010).*

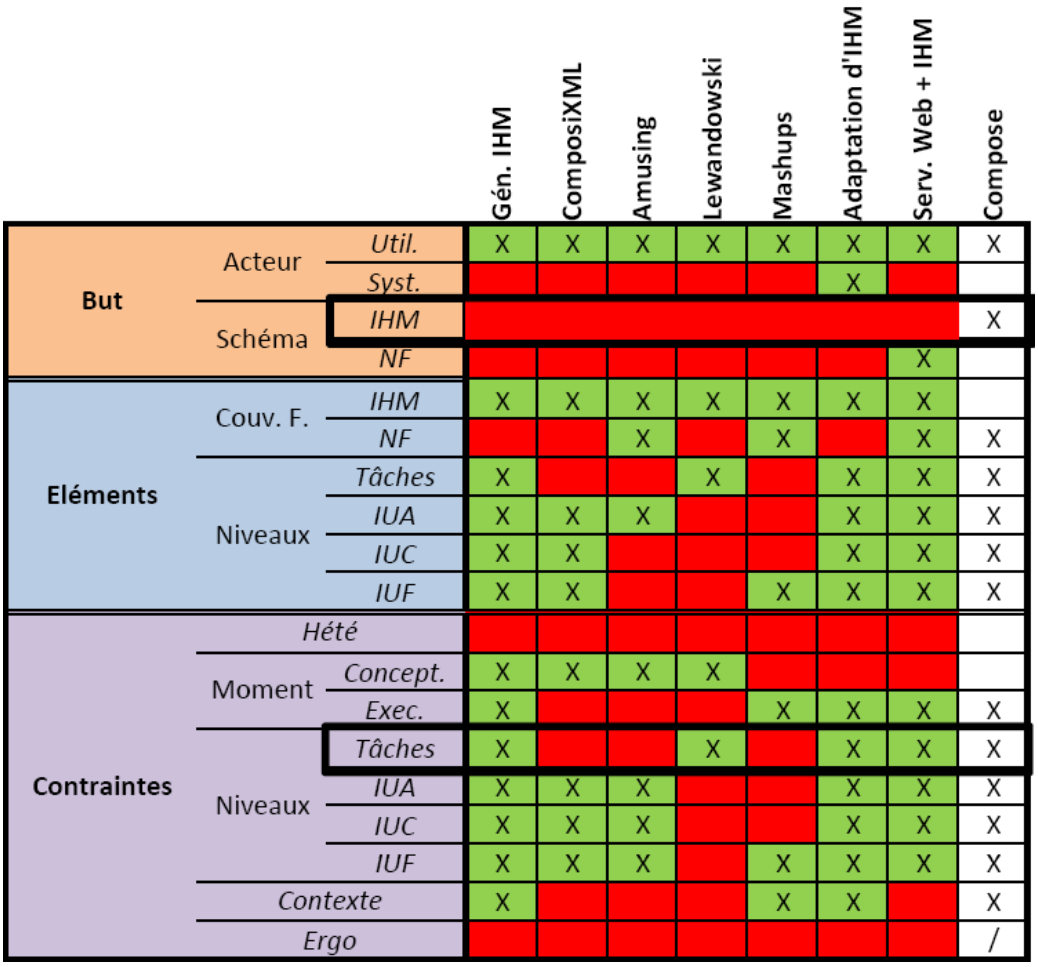
La composition d'IHM est traitée après le processus de composition dynamique et n'est pas prise en compte lors du calcul du plan d'exécution.

5.3 Synthèse et positionnement

La composition dynamique du NF se concentre sur les interactions machine-machine en éludant les interactions homme-machine. Pourtant, l'interaction homme-machine dans ces services joue un rôle important et promeut un système composé centré utilisateur. L'IHM doit être composée conjointement avec le NF pour prendre en compte ses spécificités en amont dans le processus de composition dynamique. Les travaux en composition de services identifient cependant des solutions pour composer dynamiquement des IHM : les techniques de l'Intelligence Artificielle. La planification constitue ainsi une voie à explorer pour la composition dynamique d'IHM.

La figure 5.10 propose une synthèse des principaux travaux de l'état de l'art en composition d'IHM et positionne l'objectif de cette thèse par rapport à l'existant. Les travaux en composition d'IHM se concentrent sur la composition à différents niveaux d'abstraction. Elle a lieu soit à la conception, soit à l'exécution. A la conception, ComposiXML et Amusing proposent des opérateurs de composition mais font l'impasse du modèle de tâches. D'autres le supposent connu comme les travaux en génération d'IHM. Lewandowski et al. (2007) composent des modèles de tâches et annoncent la difficulté de la sémantique. A l'exécution, les Mashups et les travaux sur l'adaptation permettent à l'utilisateur de choisir lui-même les composants et les composeurs utilisés. Ces travaux peuvent être couplés à la composition dynamique de services mais le schéma de composition n'est pas calculé à partir de l'objectif utilisateur.

Notre étude s'intéresse à la composition dynamique d'IHM pour permettre l'opportunisme promis en informatique ambiante. Ainsi, de façon complémentaire à cet état de l'art, notre travail se concentre sur la composition du modèle de tâches à partir de l'objectif utilisateur. Notre approche consiste à utiliser des algorithmes de planification pour composer ce modèle de tâches.



			Gén. IHM	ComposiXML	Amusing	Lewandowski	Mashups	Adaptation d'IHM	Serv. Web + IHM	Compose
But	Acteur	*Util.*	X	X	X	X	X	X	X	X
		Syst.						X		
	Schéma	*IHM*								X
		NF							X	
Eléments	Couv. F.	*IHM*	X	X	X	X	X	X	X	
		NF			X		X		X	X
	Niveaux	*Tâches*	X			X		X	X	X
		IUA	X	X	X			X	X	X
		IUC	X	X				X	X	X
		IUF	X	X			X	X	X	X
Contraintes	*Hété*									
	Moment	*Concept.*	X	X	X	X				
		Exec.	X				X	X	X	X
	Niveaux	*Tâches*	X			X		X	X	X
		IUA	X	X	X			X	X	X
		IUC	X	X	X			X	X	X
		IUF	X	X	X		X	X	X	X
	Contexte		X				X	X		X
	Ergo									/

FIGURE 5.10 – *Synthèse de l'état de l'art en composition d'IHM et positionnement de Compose.*

Deuxième partie

UNE SOLUTION PAR PLANIFICATION

Composition d'IHM par planification

6

Résumé

Ce chapitre se structure en trois parties. Dans la première partie, nous présentons les concepts fondamentaux en planification automatique. Dans la deuxième, nous étudions la composition dynamique d'IHM comme un problème de planification HTN. Cette étude permet de pointer les manques d'HTN pour composer un modèle de tâches. La troisième partie formalise la composition d'IHM comme un problème de planification et propose une procédure implémentée pour composer un modèle de tâches.

Publications associées : TSI'11 (Gabillon et al. 2011a), SEMAIS'11 (Gabillon et al. 2011b), JFPDA'08 (Gabillon et al. 2008)

Sommaire

Introduction

Ce chapitre a pour objectif de composer un modèle de tâches pour trouver une solution au problème de composition d'IHM par des algorithmes de planification. Nous nommons *tâche but*, la tâche racine du modèle de tâches de l'IHM composite. Par exemple, la *tâche but* pour que Victor obtienne une assistance médicale est « Get medical assistance ». Une tâche est dite *exécutable* si elle permet de passer d'un état courant à un état dans lequel la tâche est atteinte.

Ce chapitre se structure en trois temps. Dans un premier temps nous présentons les concepts fondamentaux de la planification automatique. Dans un deuxième temps, nous étudions les algorithmes HTN appliqués à la composition d'IHM. Ceci nous permet d'exprimer un problème de composition d'IHM en un problème de planification HTN. Cette étude permet de réutiliser les planificateurs HTN existants pour composer des IHM. Cependant, elle montre que les algorithmes de planification HTN ne répondent que partiellement au problème de composition d'IHM. En conséquence, nous formalisons dans un troisième temps le problème de composition d'IHM comme un problème de planification, et proposons une procédure pour résoudre ce problème.

6.1 Concepts fondamentaux

La planification automatique (Ghallab et al. 2004) relève de l'Intelligence Artificielle. Elle a pour objectif de réaliser un processus de délibération explicite qui sélectionne et organise des actions. La planification vise à calculer automatiquement un plan, c'est-à-dire un ensemble d'actions pour atteindre un but à partir d'une situation initiale donnée. Un plan doit permettre, par son exécution réelle (en général par un système robotique) ou simulée, de faire évoluer l'univers modélisé de manière à satisfaire un objectif à atteindre. Un plan est élaboré par un planificateur. Par exemple pour se rendre d'une pièce à une autre, le plan est de se lever de sa chaise, de s'orienter vers la porte, puis de marcher en réalisant un certain nombre de pas jusqu'à ce que l'on soit dans la pièce à atteindre.

Selon Devy et al. (2001), la planification prend plusieurs formes, parmi lesquelles :

– La *planification du mouvement* a pour objectif de synthétiser un chemin et une trajectoire dans l'espace d'un robot, en prenant en

compte principalement la géométrie et la cinématique.

– La *planification sensorielle* veut répondre à des questions du type : quelle information est nécessaire pour la tâche en cours et à quel moment ? Comment et où peut-on l'acquérir, avec quels capteurs ? Selon quel point de vue, avec quelles modalités sensorielles ?

– La *planification pour la navigation* choisit et organise un ensemble de primitives de localisation et de primitives de mouvement en vue d'atteindre un but ou d'explorer un environnement. Il s'agit par exemple de primitives de suivi d'une route, de contournement d'un obstacle, de poursuite d'une cible par asservissement. Cette forme de planification combine d'une certaine façon les deux précédentes.

– La *planification pour la manipulation* veut élaborer une stratégie de manipulation d'objets pour obtenir des assemblages.

– la *planification pour la communication* vise la coopération ou la coordination multi-agents. Il s'agit de déterminer et d'organiser les requêtes et les retours d'interaction pour organiser l'activité en cours.

– la *planification de tâches* est la forme la plus générale et la plus abstraite. Elle cherche à déterminer et à organiser l'ensemble des activités d'un agent (la plupart du temps un robot) dans le temps et à lui attribuer des ressources, compte tenu des évolutions prévisibles dans l'environnement.

L'approche dite « classique » de la planification de tâches est la planification dans un espace d'états. Cette approche connaît des progrès significatifs essentiellement au niveau de *représentations de plus en plus expressives* et au niveau *algorithmique* (Devy et al. 2001). L'*expressivité de la représentation* en planification dans un espace d'états est étendue en particulier pour gérer le temps, les ressources, un environnement dynamique, etc. La *résolution* par un algorithme d'un problème de planification est ambitieuse car elle équivaut à trouver un chemin solution dans un graphe d'états. La recherche de ce chemin peut demander d'explorer un nombre exponentiel d'états étant donné le nombre et la taille des variables d'état du problème. Pour faire face à cette explosion combinatoire, des algorithmes efficaces utilisent des heuristiques qui guident la recherche dans des chemins potentiellement plus courts, des solutions approximées ou même des méthodes hiérarchiques (planification hiérarchique).

Dans un premier temps, nous présentons la planification dans un espace d'états. Dans un deuxième temps, nous exposons l'approche de planification que nous avons étudiée pour la composition d'IHM : la planification hiérarchique.

6.1.1 Planification dans un espace d'états

La planification dans un espace d'états tire son nom du fait que les *états du monde* possibles sont représentés par les nœuds d'un graphe d'*états*. Les *actions* permettent de passer d'un *état* à l'autre dans ce graphe de changement d'états.

État

Le langage $\mathscr{L}$ pour représenter les connaissances est un langage de logique du premier ordre dans lequel il y a des symboles de prédicats, de variables et de constantes. Un terme de $\mathscr{L}$ est un symbole de variable ou de constante. Pour les symboles de prédicats et de constantes, $\mathscr{L}$ utilise une chaîne de caractères alphanumériques de longueur supérieure ou égale à 1 (par exemple : *isConnected* ou $pc1$). Pour les symboles de variables, $\mathscr{L}$ utilise une chaîne de caractères alphanumériques de longueur supérieure ou égale à 1, précédée d'un point d'interrogation (par exemple : $?x$ ou $?x2$). Un atome est constitué d'un symbole de prédicat et de symboles de termes (par exemple : $isConnected(pc1, ?x)$). Un littéral est un atome ou la négation d'un atome (auquel cas, il est appelé littéral positif ou négatif).

Un *état du monde* s_0 est un ensemble d'atomes positifs de $\mathscr{L}$. Par exemple, l'état du monde s est représenté par l'ensemble des atomes suivants (figure 6.1) :

$SurTable(A)$: le bloc A est sur la table,
$SurTable(B)$: le bloc B est sur la table,
$SurTable(C)$: le bloc C est sur la table,
$Sur(D, C)$: le bloc D est sur le bloc C,
$PinceVide$: la pince est vide, c'est-à-dire qu'elle ne tient pas de bloc,
$Libre(A)$: il n'existe pas de bloc sur A,
$Libre(B)$: il n'existe pas de bloc sur B,
$Libre(D)$: il n'existe pas de bloc sur D.

La notion de *satisfaction* permet de savoir si un ensemble de littéraux positifs est présent et si un ensemble de littéraux négatifs est absent dans

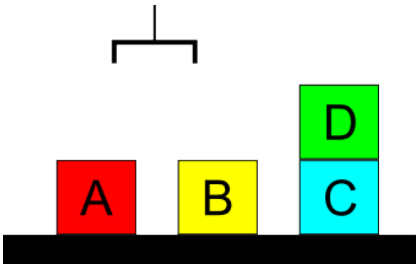


FIGURE 6.1 – *Etat s*

un état du monde.

Si g est un ensemble de littéraux (atomes positifs ou négatifs), on dit que l'état s *satisfait* g quand tous les littéraux positifs de g sont dans s et tous les littéraux négatifs de g ne sont pas dans s. Autrement dit, les littéraux positifs (respectivement négatifs) de g sont *vrais* (respectivement *faux*) dans l'état du monde s.

Par exemple, supposons s l'état de l'exemple précédent et g l'ensemble des littéraux suivants :

$SurTable(A)$: le bloc A est sur la table
$\neg Libre(C)$: il existe un bloc sur C

Alors s *satisfait* g car $SurTable(A)$ appartient à s et $Libre(C)$ n'appartient pas à s. On dit que $SurTable(A)$ est vrai dans s et que $Libre(C)$ est faux dans s.

L'espace de recherche est un graphe d'états dont les nœuds sont des états et les arcs sont des actions (Figure 6.2).

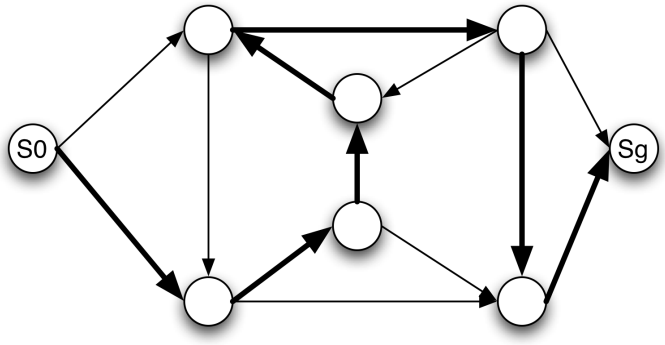


FIGURE 6.2 – *Graphe d'états.*

Action

Une *action* permet de passer d'un état à un autre dans le graphe d'états. Plus formellement, une *action* est un triplet $a = (name(a),\ precond(a),\ effects(a))$ tel que :

- $name(a)$, le nom de l'action est une expression syntaxique de la forme $n(x_1, \ldots, x_k)$, où n est un symbole appelé *symbole de l'action* et $x_1, \ldots, x_k$ sont tous les symboles de constantes qui apparaissent dans a.
- $precond(a)$, la précondition de a est un ensemble de littéraux. L'ensemble des littéraux positifs est noté $precond^+(a)$ et l'ensemble des littéraux négatifs est noté $precond^-(a)$. L'état s_i doit satisfaire $precond(a)$ pour que l'action puisse s'appliquer.
- $effects(a)$, les effets de a sont un ensemble de littéraux. L'ensemble des littéraux positifs est noté $effects^+(a)$ et l'ensemble des littéraux négatifs est noté $effects^-(a)$.

Par exemple, supposons l'état s défini dans la figure 6.1 et l'action a qui réalise l'action de prendre le bloc B :

- $name(a) : Prendre(B)$
- $precond(a) : \{PinceVide, Libre(B), SurTable(B)\}$: la pince doit être vide, le bloc B est libre et sur la table.
- $effects(a) : \{\neg PinceVide, \neg Libre(B), \neg SurTable(B), Tenu(B)\}$: La pince ne sera plus vide, le bloc B ne sera plus libre, le bloc B ne sera plus sur la table et le bloc B sera dans la pince.

Une action a est *applicable* sur l'état s_i si et seulement si s_i satisfait $precond(a)$. Si une action est *applicable*, on calcule l'état suivant s_{i+1} par la *fonction de transition* γ entre états qui effectue une transformation instantanée de l'état s_i en l'état s_{i+1} telle que $\gamma(s_i, a) = (s_i - effect^-(a)) \cup effects^+(a) = s_{i+1}$.

Par exemple, l'état $s' = \gamma(s, a)$ est l'ensemble des atomes suivants (figure 6.3) :

$SurTable(A)$: le bloc A est sur la table,
$SurTable(C)$: le bloc C est sur la table,
$Sur(D, C)$: le bloc D est sur le bloc C,
$Libre(A)$: il n'existe pas de bloc sur A,
$Libre(D)$: il n'existe pas de bloc sur C,
$Tenu(B)$: le bloc B est tenu. Ce prédicat est ajouté par a.

La fonction de transition est généralisable pour une séquence d'actions. Soit π une séquence d'actions de longueur k et s un état. L'état obtenu par l'application successive des actions de π est :

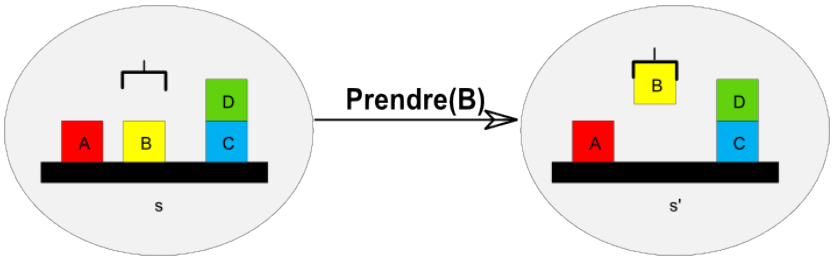


FIGURE 6.3 – *État* $s' = \gamma(s, Prendre(B))$.

$$\gamma(s,\pi) = \begin{cases} s & \text{si } k = 0 \\ \gamma(\gamma(s,a_1),\langle a_2,\ldots,a_k\rangle) & \text{si } k > 0 \text{ et } a_1 \text{ applicable à } s \end{cases}$$

Pour éviter de décrire toutes les actions explicitement, la notion d'*opérateur* est introduite. Un *opérateur* représente un ensemble d'actions.

Un *opérateur* est un triplet $o = (name(o), precond(o), effects(o))$ tel que :

- $name(o)$, le nom de l'opérateur est une expression syntaxique de la forme $n(x_1,\ldots,x_k)$, où n est un symbole appelé *symbole de l'opérateur* et $x_1,\ldots,x_k$ sont tous les symboles de constantes mais aussi de variables qui apparaissent dans o.
- $precond(o)$, la precondition de o est un ensemble de littéraux. L'ensemble des littéraux positifs est noté $precond^+(o)$ et l'ensemble des littéraux négatifs est noté $precond^-(o)$.
- $effects(o)$, les effets de o sont un ensemble de littéraux. L'ensemble des littéraux positifs est noté $effects^+(o)$ et l'ensemble des littéraux négatifs est noté $effects^-(o)$.

Par exemple, soient quatre opérateurs :

- $Prendre(?X)$: cet opérateur décrit que la pince prend un bloc $?X$.
  - $precond$: $\{PinceVide, Libre(?X), SurTable(?X)\}$: la pince doit être vide, le bloc X libre et posé sur la table
  - $effects$: $\{\neg PinceVide, \neg Libre(?X), \neg SurTable(?X), Tenu(?X)\}$: La pince ne sera plus vide, le bloc X ne sera plus libre et ne sera plus posé sur la table, il sera tenu par la pince

- $Deposer(?X)$: cet opérateur décrit que la pince dépose un bloc $?X$
  - $precond$: $\{Tenu(?X)\}$
  - $effects$: $\{\neg Tenu(?X), PinceVide, Libre(?X), SurTable(?X)\}$

- $Depiler(?X, ?Y)$: cet opérateur décrit que la pince prend le bloc $?X$ qui était sur le bloc $?Y$.
  - $precond$: $\{PinceVide, Libre(?X), Sur(?X, ?Y)\}$

- $effects : \{\neg PinceVide, \neg Libre(?X), \neg Sur(?X, ?Y), Tenu(?X), Libre(?Y)\}$

– $Empiler(?X, ?Y)$: cet opérateur décrit que la pince dépose le bloc $?X$ sur le bloc $?Y$
  - $precond : \{Tenu(?X), Libre(?Y)\}$
  - $effects : \{\neg Tenu(?X), \neg Libre(?Y), PinceVide, Libre(?X), Sur(?X, ?Y)\}$

Problème de planification

Résoudre un problème de planification signifie trouver un ensemble d'actions permettant d'atteindre un état qui satisfait le but g à partir d'un état initial s_0.

(Problème) : Un *problème* est un triplet $\mathscr{P} = (s_0, g, A)$ tel que :
– s_0 est un ensemble de littéraux représentant l'*état initial*,
– g est un ensemble de littéraux représentant le but.
– A est un ensemble d'actions.

Pour décrire les problèmes de planification, on utilise le plus souvent le langage PDDL (McDermott et al. 1998) et ses diverses extensions (Fox et Long 2003, Edelkamp et Hoffmann 2004, Gerevini et Long 2006). PDDL est fondé sur le formalisme Strips (Fikes et Nilsson 1971). Il décompose le problème en deux parties : le *domaine* qui contient l'ensemble des opérateurs O, et le *problème* constitué de l'état initial s_0 et de la définition du but g.

Plan solution

L'objectif d'un planificateur est de trouver une séquence d'actions permettant de passer de l'état initial du problème à un état final où le but g est atteint (figure 6.4).

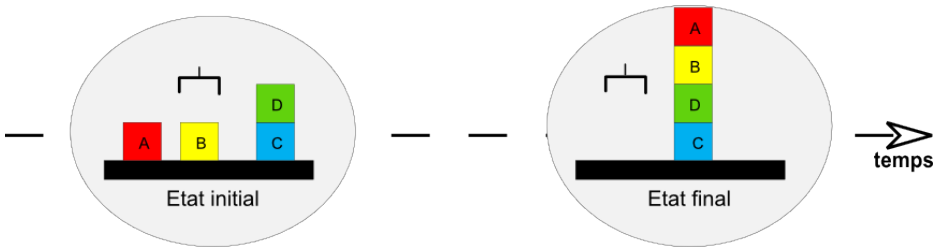


FIGURE 6.4 – *Etat initial s_0 et état final s_n.*

Un *plan* π est une séquence d'actions de la forme $\pi = \langle a_1, a_2, ..., a_n \rangle$ telle que $a_1, a_2, \ldots, a_n$ sont n actions. Un exemple de plan est :

$\langle Prendre(B), Empiler(B,D)\rangle$.

Un plan $\pi = \langle a_1, ..., a_n\rangle$ est un *plan solution* si $s_{n+1} = \gamma(s_0, \pi)$ et s_{n+1} satisfait g. s_{n+1} est appelé *état final*. Par exemple, soit l'état s défini dans la figure 6.1 et les quatre opérateurs $\{Prendre(X), Deposer(X), Empliler(X,Y), Depiler(X,Y)\}$, et le but $g = \{Sur(A,B), Sur(B,D), Sur(D,C)\}$. Le plan solution $\pi = \langle Prendre(B), Empiler(B,D), Prendre(A), Empiler(A,B)\rangle$ permet d'atteindre un état satisfaisant g (Figure6.5).

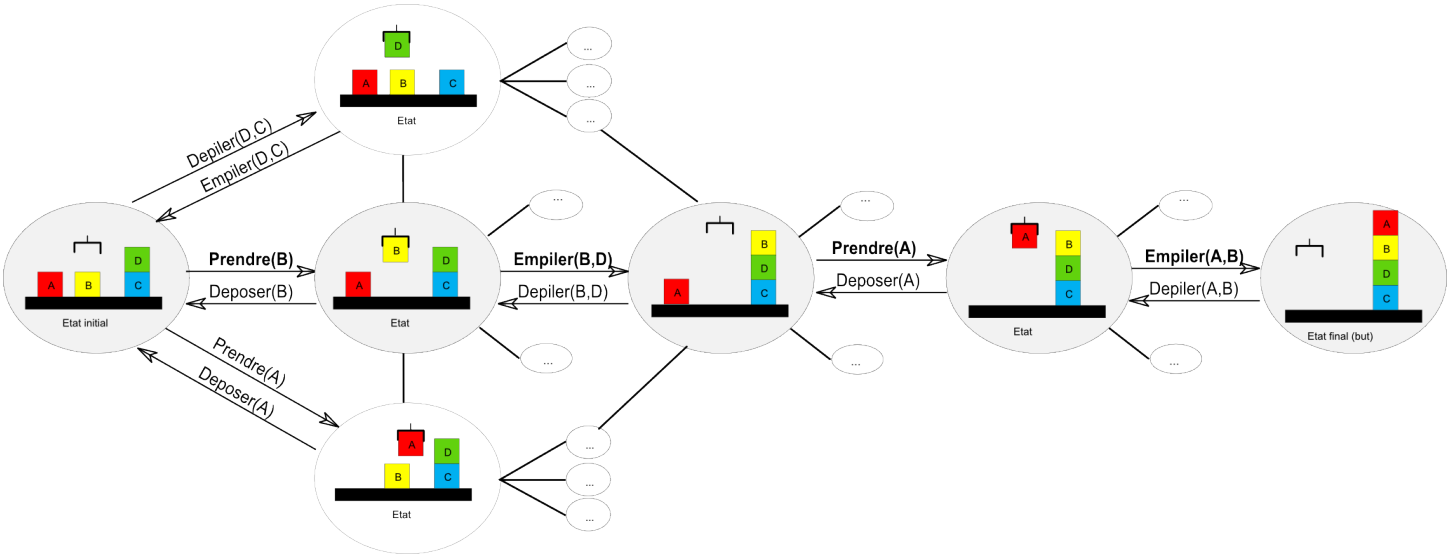


FIGURE 6.5 – *Extrait du graphe d'états G et plan solution π (en gras).*

Procédure de planification

La procédure de planification va construire et parcourir le graphe d'états G pour atteindre un état qui satisfasse le but. On peut donc mettre en œuvre les algorithmes classiques de recherche dans les graphes, tels que celui ci-dessous (algorithme 1). Cet algorithme effectue une recherche à partir de l'état initial s_0. L'appel $Forward(s_0, g, \emptyset)$ retourne un chemin dans G de s_0 à un état satisfaisant g, s'il existe. L'étape « Choix-NonDeterministe » signifie qu'une action est choisie dans l'ensemble des *actions_applicables*. Dans le pire des cas, toutes les actions sont testées. Si aucune action ne convient, le choix non déterministe renvoie un échec.

L'algorithme est correct et complet.

Algorithme 1 : **Forward (**s**,** g**,** *path***)** - Recherche avant

Data : s l'état courant, g le but et *path* le chemin parcouru.
Result : *path* un chemin solution s'il existe, *échec* sinon.
if $g \in s$ **then**
  return *path*;
else
  $actions\_applicables \leftarrow \{a \in A | s$ satisfait $precond(a)\}$;
  if $actions\_applicables = \emptyset$ **then**
    return *échec*;
  else
    ChoixNonDeterministe $a \in actions\_applicables$;
    return $(Forward(\gamma(s,a), g, path \cup a)$;
  end
end

L'algorithme présenté n'est qu'un exemple. D'autres algorithmes sont développés pour effectuer la recherche d'une solution comme (Nau et al. 1999, Ilghami et Nau 2003, Blum et Furst 1995).

6.1.2 Planification hiérarchique

La planification hiérarchique (Hierarchical Task Network (Erol 1995), HTN) permet de décrire le domaine de planification en utilisant des *tâches*. Un planificateur HTN diffère du planificateur dans un espace d'états par le fait que l'objectif n'est plus d'atteindre un état satisfaisant le but, mais d'accomplir un ensemble de *tâches*. Par exemple, l'objectif peut être de voyager de Montpellier à Grenoble ($travel(Montpellier, Grenoble)$). Le planificateur HTN décompose les *tâches non primitives* récursivement en tâches de plus en plus petites, jusqu'à ce que des *tâches primitives* puissent être exécutées. Par exemple, la tâche « taxi-travel » se décompose en « get-taxi », « ride », « pay-driver ». Les tâches primitives sont exécutées en utilisant un opérateur de la planification dans un espace d'états.

Les définitions de terme, littéral, opérateur, action, fonction de transition et plan sont les mêmes qu'en planification dans un espace d'états. Cependant le langage HTN comporte les notions de tâche, de méthode et de réseau de tâches utilisées dans la définition d'une solution.

Tâches et méthodes

Le langage HTN possède des *symboles de tâches*. Les symboles d'opérateurs sont des symboles de tâches primitives, auxquels on ajoute des symboles de tâches nommés *symboles de tâches non primitives* qui sont les symboles des méthodes.

Une *tâche* est une expression de la forme $t(r_1, \ldots, r_k)$ telle que t est un symbole de tâche, et $r_1, \ldots, r_k$ sont des termes. Si t est un symbole d'opérateur, la tâche est *primitive*, sinon la tâche est *non primitive*. Une tâche est *instanciée* si $r_1, \ldots, r_k$ sont des constantes. Par exemple, la tâche $travel(?x, ?y)$ est une tâche non instanciée contrairement à $travel(Grenoble, NewYork)$.

On dit qu'une action a est *pertinente* pour une tâche primitive t dans l'état s si $name(a) = t$ et a est applicable sur s. Par exemple, une action fly est pertinente pour la tâche $fly(?x, ?y)$.

Un *réseau de tâches* est un graphe acyclique $w = (U, E)$ où U est un ensemble de nœuds et E un ensemble d'arcs. Chaque nœud $u \in U$ est une tâche t_u. w est *instancié* si toutes les tâches $\{t_u | u \in U\}$ sont instanciées, sinon w est non instancié. w est primitif si toutes les tâches $\{t_u | u \in U\}$ sont primitives, sinon w est *non primitif*.

Les arcs de w définissent un ordonnancement partiel de U. Il existe un arc (u, v) entre les nœuds $u, v \in U$ si et seulement si il existe un chemin de u à v. Dans ce cas, on dit que u précède v.

Une *méthode* est un quadruplet $m = (name(m), precond(m), network(m))$ tel que :

- $name(m)$ est le nom de la méthode. Ce nom est une expression syntaxique de la forme $n(x_1, \ldots, x_k)$, où n est un symbole appelé *symbole de la méthode* et $x_1, \ldots, x_k$ sont tous les symboles de variables ou de constantes qui apparaissent dans m. Une méthode est *instanciée* si tous les symboles sont des symboles de constantes. Une méthode m est $applicable$ dans un état du monde s si s satisfait $precond(m)$,
- $precond(m)$ est un ensemble de littéraux appelé précondition de m.
- $network(m)$ est le réseau de tâches décomposant m. Les tâches de ce réseau sont les sous-tâches de $task(m)$ notées $subtasks(m)$.

Par exemple, pour voyager (tâche $travel(?x, ?y)$), supposons qu'il existe deux méthodes : une pour voyager en avion et l'autre en taxi. Supposons les tâches $u_1 = get\_ticket$, $u_2 = travel(?x, ?a)$, $u_3 = fly(?a, ?b)$ et $u_4 = travel(?b, ?y)$. La méthode pour voyager par avion « $travel(?x, ?y)$ » se définit comme suit (figure 6.6) :

- $name(m) = travel(?x, ?y)$
- $precond(m) = \{long\text{-}distance(?x, ?y)\}$
- $network(m) = (U, E)$ tel que $U = \{u_1, u_2, u_3, u_4\}$ et $E = \{(u_1, u_3), (u_2, u_3), (u_3, u_4)\}$.

Supposons les tâches $u_5 = get\_taxi(?x, ?y)$, $u_6 = ride(?x, ?y)$ et $u_7 = pay\_driver$. La méthode pour voyager en taxi « $travel(?x, ?y)$ » se définit comme suit (figure 6.6) :

- $name(m) = travel(?x, ?y)$
- $precond(m) = \{short\text{-}distance(?x, ?y)\}$
- $network(m) = (U, E)$ tel que $U = \{u_1, u_2, u_3\}$ et $E = \{(u_1, u_2), (u_2, u_3)\}$.

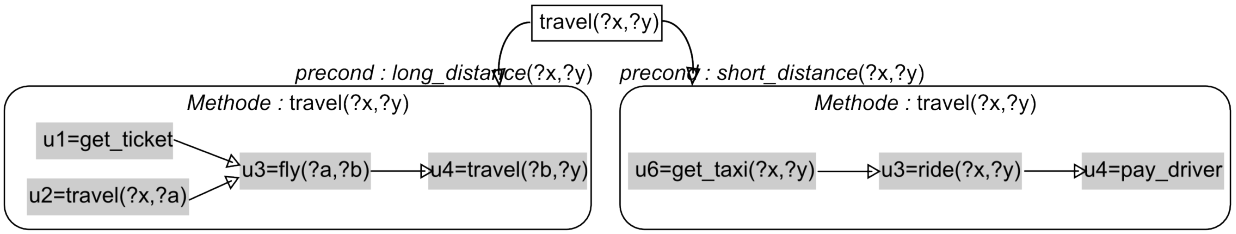


FIGURE 6.6 – *Deux méthodes pour $travel(?x, ?y)$: en avion ou en taxi.*

Une méthode doit être applicable dans l'état du monde courant et pertinente pour une tâche à accomplir. Nous formalisons ce que nous entendons par *pertinent*.

On dit qu'une méthode m est *pertinente* pour une tâche primitive t dans l'état s si $name(m) = t$ et m est applicable sur s.

Quand une méthode m est utilisée pour décomposer une tâche t, en pratique, c'est que cette tâche fait partie d'un nœud de U. Par exemple, la tâche $travel(?x, ?y)$ est décomposée par deux méthodes pour voyager. Dans ce cas, la tâche non primitive est remplacée par ses sous-tâches pour obtenir le réseau de tâches suivant $\delta(w, u, m, \sigma)$ défini comme suit.

Soit $w = (U, E)$ un réseau de tâches, u un nœud qui n'a pas de prédécesseurs dans w, et m une méthode pertinente pour t_u selon la substitution σ. Soit $succ(u)$ l'ensemble des successeurs immédiats de u, i.e., $succ(u) = \{u' \in U | (u, u') \in E\}$. Soit $succ_1(u)$ l'ensemble de tous les successeurs immédiats de u pour qui u est le seul prédécesseur. Soit (u', E') le réseau résultant de la suppression de u et de tous les arcs contenant u. Soit (U_m, E_m) une copie du réseau de tâches $network(m)$. Si (U_m, E_m) est non vide, alors le résultat de la décomposition de u dans w par m selon σ est le réseau de tâches suivant :

$$\delta(w, u, m, \sigma) = \{(\sigma(U' \cup U_m), \sigma(E_v))|v \in subtasks(m)\}$$

où :

$$E_v = E_m \cup (U_m \times succ(u)) \cup \{(v, u')|u' \in succ_1(u)\}$$

Sinon, $\delta(w, u, m, \sigma) = \{(\sigma(U'), \sigma(E'))\}$.

Par exemple, soit le réseau de tâches w constitué d'une seule tâche $u = travel(Grenoble, NewYork)$. Soit une méthode pour se déplacer en avion $m = travel(?x, ?y)$ pertinente pour u (avec $\sigma = \{(?x \rightarrow Grenoble), (?y \rightarrow NewYork)\}$. Soir m la méthode pour se déplacer en avion (figure 6.6). Alors $\delta(w, u, m, \sigma)$ calculé est constitué d'un seul réseau suivant obtenu en remplaçant la tâche *travel* par $network(m)$ (figure 6.7). Le réseau suivant est représenté dans la figure 6.7.

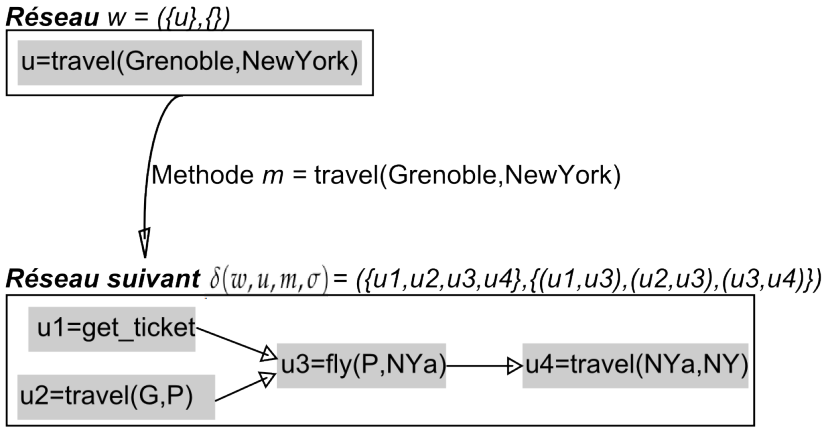


FIGURE 6.7 – *Réseau suivant* $\delta(w, u, m, \sigma)$.

L'interprétation est intuitive : u est supprimé de w et une copie de $subtasks(m)$ est insérée à la place. Par exemple, le nœud tâche $u = travel(Grenoble, NewYork)$ est remplacé par les nœuds $u_1 = get - ticket$, $u_2 = travel(Grenoble, NewYork\_airport)$, $u_3 = fly(Paris, NewYork\_airport)$ et $u_4 = travel(NYa, NY)$. Chaque contrainte d'ordonnancement (arc dans E) appliquée à u, est appliquée aux sous-tâches de m ($subtasks(m)$). Dans l'exemple précédent, $(u, \emptyset)$ (où v est un nœud quelconque) est remplacé par $([u_1, u_2, u_3, u_4], \emptyset)$.

Comme les $subtask(m)$ sont partiellement ordonnées, il peut y avoir plusieurs nœuds qui n'ont pas de prédécesseurs dans w. Il peut donc y avoir différents réseaux suivants selon le nœud u sélectionné. En conséquence, $\delta(w, u, m, \sigma)$ est un ensemble d'alternatives de réseaux suivants. Par exemple, le réseau suivant de la figure 6.7 peut engendrer deux réseaux suivants selon si la tâche *get_ticket* ou *travel* est sélectionnée en premier.

Domaine, problème et plan solution HTN

Un *domaine* HTN est une paire $\mathscr{D} = (O, M)$ où O est l'ensemble des opérateurs et M l'ensemble des méthodes.

Un *problème HTN* utilise un domaine HTN. Le but est exprimé sous la forme d'un réseau de tâches w à réaliser.

Plus formellement, un *problème* HTN est un quadruplet $\mathscr{P} = (s_0, w, O, M)$ où s_0 est l'état initial, w est le réseau de tâches initial, O est l'ensemble des opérateurs et M l'ensemble des méthodes représentant le domaine du problème.

Intuitivement, trouver une solution à un problème HTN $\mathscr{P}$, c'est trouver une séquence d'actions $\pi = \langle a_1, \ldots, a_n \rangle$ permettant d'accomplir le réseau de tâches initial w, i.e, une manière de décomposer w en π tel que π est exécutable à partir de l'état du monde courant.

Soit $\mathscr{P} = (s_0, w, O, M)$ un problème HTN. Il y a trois cas dans lesquels un plan π est solution de $\mathscr{P}$:

- **Cas 1 :** w est vide. Alors, π est solution de $\mathscr{P}$ si π est vide (i.e $n = 0$).

- **Cas 2 :** Il existe un nœud tâche primitif $u \in w$ qui n'a pas de prédécesseurs dans w. Alors π est solution de $\mathscr{P}$ si a_1 est applicable à t_u dans s_0 et le plan $\pi = \langle a_2, \ldots, a_n \rangle$ est solution au problème :

  $$\mathscr{P}' = (\gamma(s_0, a_1), w - \{u\}, O, M)$$

  Intuitivement, $\mathscr{P}'$ est le problème de planification produit par l'exécution de la première action de π et la suppression du nœud tâche correspondant à a_1. Par exemple, une fois l'action $a_1 = get\_ticket$ exécutée, cette action est solution si les actions $get\_taxi(G, P), \ldots, pay\_driver$ sont exécutées successivement à partir de l'état $\gamma(s_0, a_1)$.

- **Cas 3 :** Il existe un nœud tâche non primitif $u \in w$ qui n'a pas de prédécesseurs dans w. Supposons une méthode instanciée $m \in M$ tel que m est pertinente pour t_u et m est applicable dans s_0. Alors, π est solution de $\mathscr{P}$ s'il existe un réseau de tâches $w' \in \delta(w, u, m, \sigma)$ tel que π est une solution de (s_0, w', O, M).
  Intuitivement, il faut calculer tous les réseaux suivants jusqu'à obtenir des tâches primitives. Si ces tâches sont solution, alors la tâche non primitive est solution.

Si π est solution de (s_0, w, O, M), alors pour chaque nœud tâche $u \in U$, il existe un *arbre de décomposition* dont les feuilles sont les actions de π.

Pour notre exemple, le plan solution π pour le réseau $w = travel(Grenoble, Paris)$ est constitué des actions $\langle$ *get-ticket*, $get\text{-}taxi(Grenoble, Paris)$, $ride(Grenoble, Paris)$, *pay-driver*, $fly(Paris, New\ York\ airport)$, $get\text{-}taxi(New\ York\ airport, New\ York)$, $ride(New\ York\ airport, New\ York)$, *pay-driver* $\rangle$. L'arbre de décomposition est représenté dans la figure 6.8

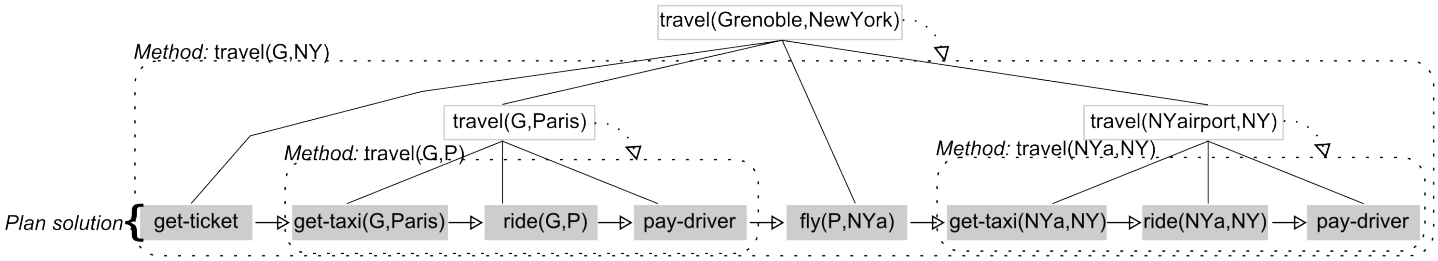


FIGURE 6.8 – *Plan solution et arbre de décomposition obtenu.*

Procédure de planification HTN

La procédure **Basic-HTN** de l'algorithme 2 présente une procédure de base pour résoudre un problème HTN (Erol 1995, Ghallab et al. 2004) selon cette formalisation. Cette procédure produit un plan solution π à un problème de planification $\mathscr{P}$.

Dans le cas où $w = \emptyset$, l'algorithme renvoie le plan solution vide. Sinon, un nœud tâche t_u qui n'a pas de prédécesseur est choisi de manière non déterministe. Dans le cas où la tâche t_u est primitive (cas 2), la procédure choisit une action a pertinente pour la tâche t_u et l'applique pour calculer l'état initial du problème suivant $\mathscr{P}'$. Si $\mathscr{P}'$ a une solution, l'action a est ajoutée au plan solution. Dans le cas où la tâche t_u est non primitive (cas 3), la procédure choisit une méthode m décomposant t_u et calcule le réseau suivant. Cette procédure est une implémentation directe

de la définition d'une solution.

Algorithme 2 : **Basic-HTN** (s_0, w, O, M) - Procédure de planification HTN

Data : (s_0, w, O, M) un problème de planification HTN.
Result : π un plan solution s'il existe, *échec* sinon.
if $w = \emptyset$ **then**
  return *échec*;
end
ChoixNonDeterministe $u \in w$ qui n'a pas de prédécesseur dans w;
if t_u *est primitive* **then**
  $active \leftarrow \{(a, \sigma) | a$ est une instance d'un opérateur dans O,
    σ est une substitution tel que $name(a) = \sigma(t_u)$,
    et a est applicable sur s_0 $\}$;
  if $active = \emptyset$ **then**
    return *échec*;
  end
  ChoixNonDeterministe $(a, \sigma) \in active$;
  $\pi \leftarrow$Basic-HTN$(\gamma(s_0, a), \sigma(w - \{u\}), O, M)$;
  if $\pi = échec$ **then**
    return *échec*;
  else
    return $a.\pi$;
  end
else
  $active \leftarrow \{(m, \sigma) | m$ est une instance d'une méthode dans M,
    σ est une substitution tel que $name(m) = \sigma(t_u)$,
    et m est applicable sur s_0 $\}$;
  if $active = \emptyset$ **then**
    return *échec*;
  end
  ChoixNonDeterministe $(m, \sigma) \in active$;
  ChoixNonDeterministe d'un réseau de tâches $w' \in \delta(w, u, m, \sigma)$;
  return Basic-HTN (s_0, w', O, M);
end

Discussion

On constate des correspondances directes entre la planification HTN et la composition dynamique d'IHM :

- La notion réseau de tâches primitif permet d'exprimer l'objectif de l'utilisateur.
- La notion d'état permet de représenter le contexte d'usage et les propriétés ergonomiques à privilégier.

– Les actions et les méthodes correspondent aux tâches du modèle de tâches.
– L'arbre de décomposition calculé correspond au schéma de composition d'IHM.

Les similitudes entre les domaines de la planification et de l'IHM d'une part, la capacité de la planification à proposer des algorithmes spécifiques permettant d'atteindre un but d'autre part, ont motivé notre exploration de la planification HTN pour la composition dynamique d'IHM.

6.2 Composition d'IHM par planification HTN

Nous étudions la capacité des algorithmes de planification HTN à calculer le modèle de tâches de l'IHM composite.

Nous exprimons dans un premier temps le problème de composition d'IHM en un problème de planification HTN. Dans un deuxième temps, nous étudions comment transformer le plan solution HTN obtenu en un modèle de tâches calculé.

Par la suite, nous nommons *tâche HTN* une tâche du *réseau de tâches*, par opposition à *tâche IHM*, qui est une tâche du *modèle de tâches*.

6.2.1 Composition d'IHM exprimée en problème de planification HTN

Nous exprimons en planification HTN les entrées d'un système de composition dynamique d'IHM : l'objectif utilisateur, le contexte d'usage, les propriétés ergonomiques à satisfaire ainsi que les tâches et opérateurs disponibles.

L'**objectif utilisateur** est décrit par une tâche HTN à réaliser. Cette tâche constitue le **réseau de tâches initial** w du problème HTN. Par exemple, la tâche « Get medical assistance » du cas d'étude est le réseau initial.

Le **contexte d'usage courant** et les propriétés ergonomiques sont décrits par l'**état initial** s_0 du problème HTN. Les propriétés ergonomiques à privilégier sont décrites par des prédicats. Si un tel prédicat est présent dans un état, la propriété est privilégiée. Par exemple, l'état initial peut

être est représenté par l'ensemble des propositions logiques suivantes :

$$s_0 = \begin{cases} has(Victor, Smartphone) \\ internet(Smartphone) \\ at(Victor, Montpellier) \\ Guidance\_Prompting() \\ \dots \end{cases}$$

Dans cet état, le critère « Guidage - Incitation » est privilégié.

Les **tâches IHM** sont décrites par les **tâches HTN**. Par exemple, la tâche IHM « Call the office » est décrite par la tâche HTN de même nom. Ainsi, les effets temporaires d'une action peuvent être exprimés par la présence d'un prédicat à la fois dans les effets positifs et négatifs.

Les *opérateurs entre tâches* IHM expriment les informations sur la manière de décomposer une tâche IHM non élémentaire. Nous avons vu qu'une tâche HTN se décompose par un réseau de tâches. Nous décrivons donc les décompositions des tâches IHM en réseau de tâches HTN. Les **opérateurs entre tâches** sont décrits par des **contraintes** (arcs) sur les nœuds tâches HTN. Soient les tâches IHM T, T_1, T_2, ..., T_n telles que T se décompose en T_1, T_2, ..., T_n. Le réseau de tâches varie selon l'opérateur entre tâches décomposant T.

Puis

L'opérateur entre tâches « **puis** » signifie que T est exécutable si T_1, T_2, ... puis T_n sont exécutables. Cette séquence se décrit en HTN par un arc entre les nœuds tâches du réseau de tâches (figure 6.9). Ainsi la tâche HTN T est décomposée par une méthode m tel que :

- $name(m) = T$ est le nom de la méthode réalisant T.
- $network(m) = (U, E)$ tel que $U = \{u_{T_1}, u_{T_2}, \dots, u_{T_n}\}$ et $E = \{(u_{T_1}, u_{T_2}), (u_{T_2}, u_{T_3}), \dots, (u_{T_{n-1}}), u_{T_n})\}$ où u_{T_1}, u_{T_2}, ..., u_{T_n} sont les nœuds tâches contenant respectivement T_1, T_2, ..., T_n.

Ordre Indépendant et Concurrence

Les opérateurs entre tâches « **ordre indépendant** » et « **concurrence** » signifient que T est exécutable si T_1, T_2, ..., et T_n le sont. Ainsi, ces opérateurs se décrivent en HTN par le fait que T_1, T_2, ..., et T_n sont exécutables sans contrainte d'ordre entre elles. Ainsi, T est décomposée par une méthode m tel que (figure 6.10) :

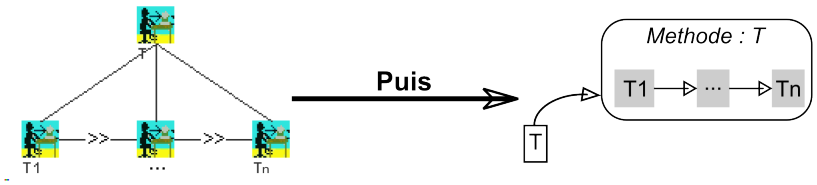


FIGURE 6.9 – *Décomposition de T avec l'opérateur puis exprimée en langage HTN.*

- $name(m) = T$
- $network(m) = (U, E)$ tel que $U = \{u_{T_1}, u_{T_2}, \ldots, u_{T_n}\}$ et $E = \emptyset$ où u_{T_1}, u_{T_2}, ..., u_{T_n} sont les nœuds tâches contenant respectivement T_1, T_2, ..., T_n.

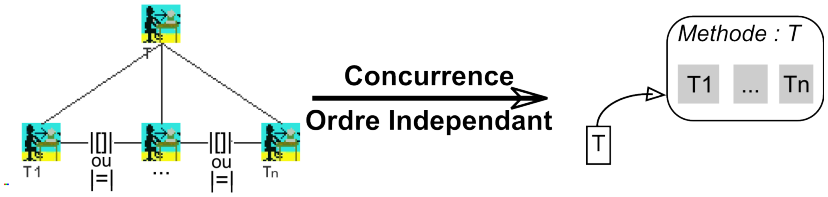


FIGURE 6.10 – *Décomposition de T avec l'ordre indépendant (ou concurrence) exprimée en langage HTN.*

Choix

L'opérateur entre tâches « **choix** » signifie que l'utilisateur choisit s'il souhaite interagir avec T_1, T_2, ... ou T_n. Une seule sous-tâche de T est exécutable. En conséquence, nous considérons que T est exécutable si T_1, T_2, ... ou T_n est exécutable. Ainsi la tâche HTN T décrivant la tâche IHM est décomposée par n méthodes m pour les n sous-tâches possibles de la tâche T définie ainsi (figure 6.11) :

- $name(m) = T$
- $network(m) = (U, E)$ tel que $U = \{u_{T_i}\}$ et $E = \emptyset$ où u_{T_i} est le nœud tâche contenant la i-ième tâche T_i.

Problème de composition d'IHM en problème HTN

Nous sommes maintenant capables d'exprimer un problème de composition d'IHM en un problème HTN :

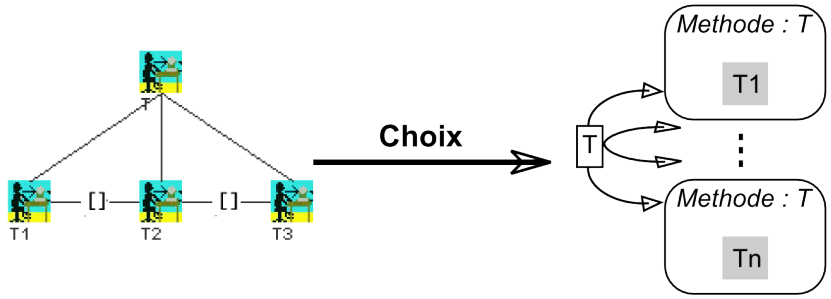


FIGURE 6.11 – *Décomposition de T avec choix exprimée en langage HTN.*

On définit le problème HTN de la composition d'IHM comme un quadruplet $\mathscr{P} = (s_0, w, O, M)$ où s_0 est l'état initial décrivant le contexte d'usage et les propriétés ergonomiques à privilégier, w est le réseau de tâches initial décrivant la tâche but (l'objectif de l'utilisateur), O est l'ensemble des opérateurs et M l'ensemble des méthodes décrivant les tâches IHM.

6.2.2 Plan solution décrivant une IHM composite

Dans cette phase, l'objectif est d'étudier l'expressivité du plan solution HTN pour l'interpréter comme un modèle de tâches.

Un planificateur HTN produit une séquence d'actions $\pi = \langle a_1, ...a_n \rangle$. Il existe donc un *arbre de décomposition* dont les feuilles sont les actions de π.

L'**arbre de décomposition** fournit un **modèle de tâches** de l'IHM composite. Les **tâches HTN** de l'arbre de décomposition sont les **tâches IHM** de l'IHM composite exécutable sur une **plate-forme disponible**. La plate-forme disponible est référencée par un paramètre des actions. Par exemple, l'action « $Call\_the\_assistance(Wall)$ » est la tâche appeler un médecin (figure 3.7) exécutable sur le mur numérique de l'hôtel. Les tâches IHM élémentaires de ce modèle de tâches sont les **actions** du plan solution (i.e. les feuilles de l'arbre de décomposition). Les tâches IHM non élémentaires décrivent les **méthodes** de l'arbre de décomposition.

On constate que la procédure de planification HTN peut être utilisée pour composer un modèle de tâches. Cependant, le plan solution étant une séquence d'actions, seul l'**opérateur entre tâches « puis »** est exprimé dans la solution. La procédure ne tient pas non plus compte des décorations des tâches.

Pour la version 1 du cas d'étude, l'objectif utilisateur est décomposé

en un réseau de tâches pour obtenir le plan solution et l'arbre de décomposition de la figure 6.12. Ce plan solution exprime un modèle de tâches. Chaque tâche de l'arbre de décomposition peut éventuellement être réifiée en une IHM (figure 6.13). Par contre, les opérateurs entre tâches exprimés dans le plan solution sont des « puis » puisque le plan solution trouvé est une séquence.

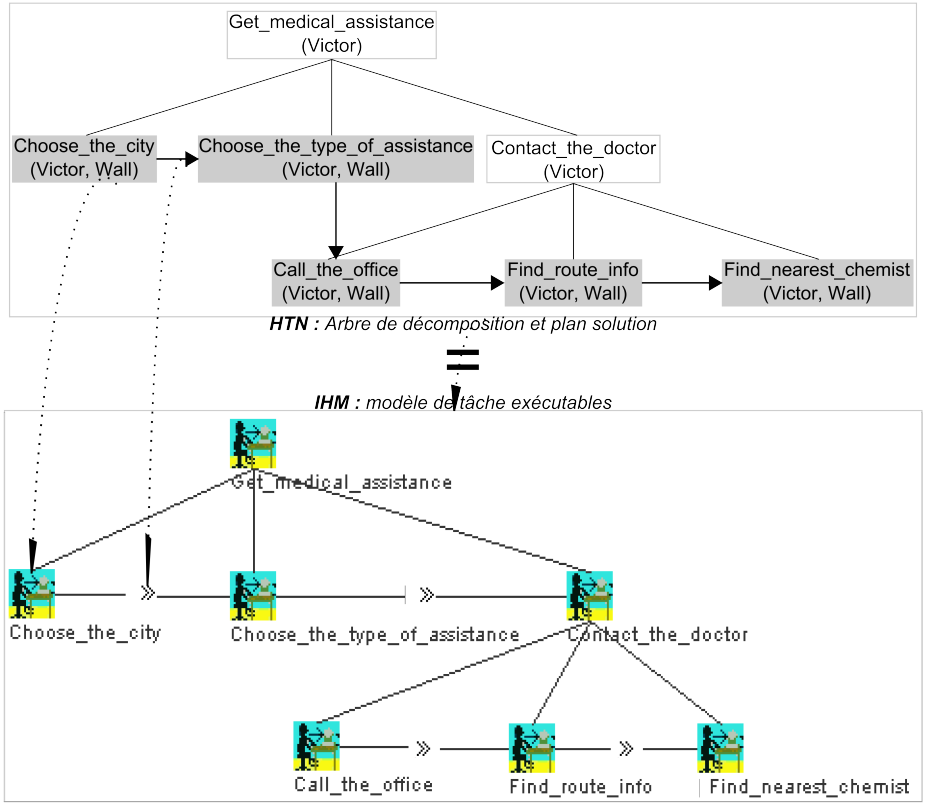


FIGURE 6.12 – *Interprétation du plan solution en modèle de tâches.*

Discussion

Dans une problématique d'informatique ambiante, la planification HTN est appropriée pour la composition d'IHM grâce aux propriétés suivantes :

- la correspondance entre les entrées (objectif utilisateur, contexte d'usage, tâches) ; idem pour les sorties (IHM exécutable permettant à l'utilisateur de réaliser son objectif) ;
- l'existence de planificateurs pour le calcul de procédures permettant de réaliser un but dans un état du monde donné.

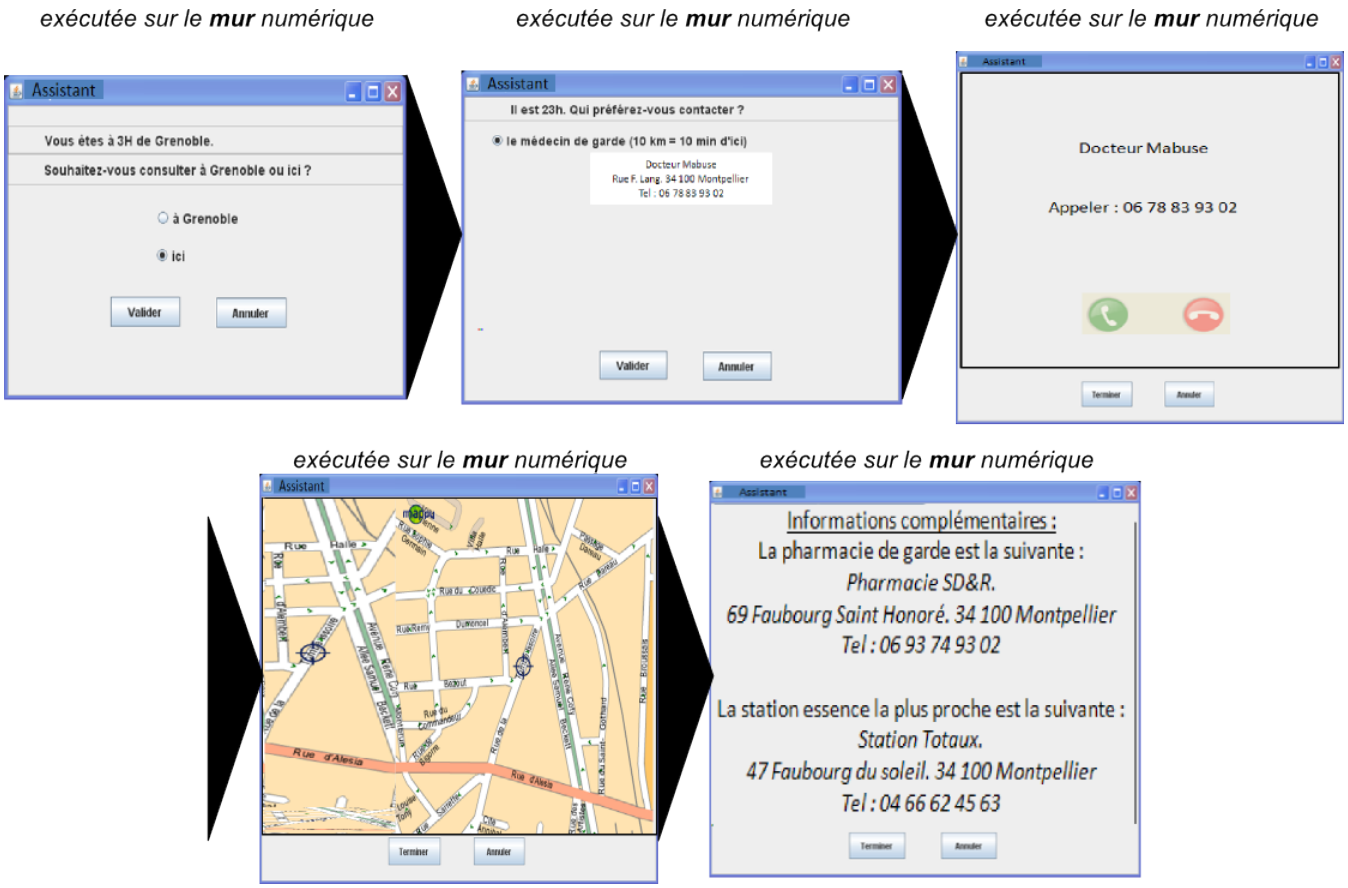


FIGURE 6.13 – *Exemple de composition en séquence des IHM du cas d'étude.*

On constate que la planification HTN produit une solution au problème de composition d'IHM.

Cependant, l'expressivité du plan solution HTN obtenu est limitée pour la composition d'IHM. En effet, la solution calculée n'exprime pas les opérateurs entre tâches « choix », « concurrence » et « ordre indépendant ». L'IHM composite ne peut être qu'une séquence d'IHM (figure 6.13). Les décorations de tâches (itération et optionalité) ne sont pas non plus couvertes.

L'opérateur « choix » pose une difficulté particulière. En effet, une fois un plan solution calculé, l'ajout d'un choix pour l'utilisateur entraîne la remise en cause du plan. Par exemple, si une IHM pour se déconnecter d'internet est ajoutée comme choix, elle peut remettre en cause l'exécutabilité des autres IHM nécessitant cette connexion.

Cette étude permet d'orienter notre travail et d'envisager une formalisation du problème de composition d'IHM et une procédure pour le résoudre par planification. Cette procédure a pour objectif de calculer un arbre de tâches possédant des tâches en parallèle, plutôt qu'une simple suite d'actions.

6.3 Formalisation d'une solution par planification

Notre objectif est de formaliser le problème de la composition d'IHM et sa solution par planification pour calculer dynamiquement le modèle de tâches de l'IHM composite. Par rapport à HTN, nous souhaitons produire un plan solution exprimant le parallélisme de deux IHM. Cette formalisation reprend les concepts existants d'état, d'action, d'opérateur déjà définis, et adapte le concept de méthode pour exprimer les opérateurs entre tâches « concurrence » et « ordre indépendant ». Cette formalisation permet de définir une solution au problème de planification. Ainsi, nous proposons ensuite une procédure de planification pour calculer un modèle de tâches solution. Cette procédure est implémentée dans le planificateur *Compose Planner* intégré à *Compose*.

6.3.1 État et action

La notion d'état ne change pas par rapport à HTN. L'état courant est l'état initial s_0 d'un graphe d'états. Cet état est représenté par un

ensemble de propositions logiques comme précédemment :

$$s_0 = \begin{cases} has(Victor, Smartphone) \\ internet(Smartphone) \\ at(Victor, Montpellier) \\ Guidance\_Prompting() \\ \dots \end{cases}$$

Dans cet état, le prédicat $Guidance\_Prompting()$ signifie que ce critère est privilégié.

De la même façon, les besoins de l'utilisateur, c'est-à-dire son but, sont représentés par un ensemble de propositions g. On appelle états but s_g les états satisfaisant g :

$$s_g = \begin{cases} found(Victor, medical\_assistance) \\ ... \end{cases}$$

Contrairement à HTN, l'espace de recherche est également un graphe d'états dont les nœuds sont des états et les arcs des actions (figure 6.2). Les actions a sont définies de manière identique à HTN. Elle seront réalisées via les IHM.

6.3.2 Opérateur et méthode

L'algorithme de planification connaît les actions que l'utilisateur peut réaliser via les IHM préexistantes. Ces actions sont exprimées « en intention » sous la forme d'*opérateurs* et de *méthodes*. La notion d'opérateur est identique à HTN. Par contre, les méthodes tiennent compte de l'opérateur entre tâches pour exprimer le parallélisme.

Une méthode m est de la forme $(name(m), type(m), precond(m), subAct(m))$. Elle est définie, à l'instar d'HTN, par son nom, ses préconditions et sa décomposition $subAct(m)$ constituée d'opérateurs et de méthodes à accomplir pour exécuter m. Nous ajoutons le type de la décomposition exprimé dans $type(m)$ qui est soit une *séquence* soit un *parallélisme*. Dans une séquence, les méthodes ou opérateurs de $subAct(m)$ sont totalement ordonnés. Dans un parallélisme, les éléments de $subAct(m)$ peuvent être exécutés sans contrainte d'ordre entre eux. Par exemple, la méthode « *Get_medical_assistance* » correspond à la séquence de « *Choose_the_city* », « *Choose_the_doctor* » puis « *Contact_the_doctor* ». Au contraire, la méthode « *Contact_the_doctor* » correspond au parallélisme de « $Call\_the\_office$ » et « $Find\_route\_information$ ».

6.3.3 Solution

Nous allons maintenant formaliser la solution du problème de composition par planification. Cette solution ne se limite plus à une suite d'actions mais inclue le parallélisme nécessaire en IHM.

A partir de l'état initial s_0, du but g et d'un ensemble de méthodes et opérateurs, l'algorithme de planification calcule un modèle de tâches (arbre de décomposition) qui permet à l'utilisateur d'accomplir son objectif. Ce modèle de tâches contient deux types de nœuds (Figure 6.14) : les feuilles de l'arbre sont des actions c'est-à-dire des opérateurs totalement instanciés ; les nœuds internes sont des méthodes totalement instanciées (toutes les variables sont liées à des constantes) de type *séquence* (« $>$ » équivalent à « puis ») ou *parallèle* (« $||$ », exprimant « concurrence » et « ordre indépendant »).

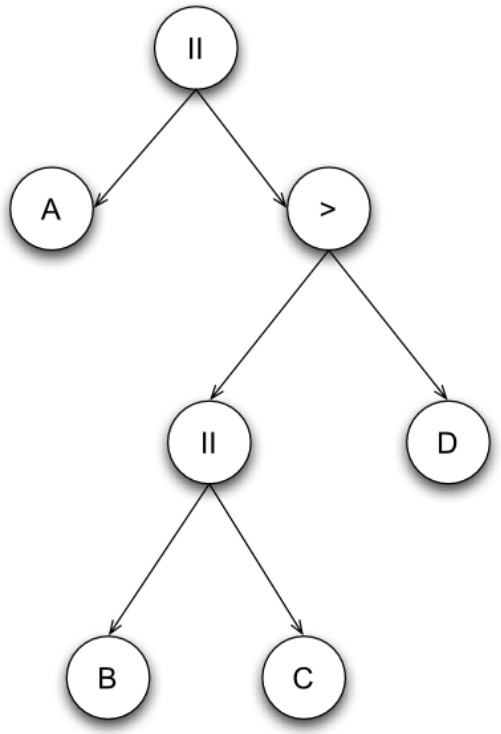


FIGURE 6.14 – *Modèle de tâches obtenu par le planificateur Compose.*

On appelle projection ρ du modèle de tâches, l'ensemble partiellement ordonné d'actions qui en découle (Figure 6.15). La relation d'ordre « $C < D$ » exprime que nécessairement l'action C *précède* l'action D. Dans la figure 6.15, 0 et ∞ sont des actions formelles représentant le début et la fin de l'exécution de l'IHM composite.

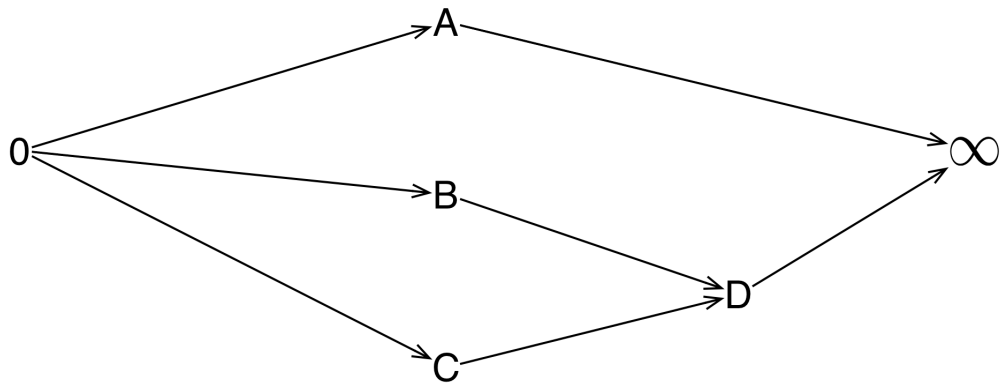


FIGURE 6.15 – *Projection ρ de l'arbre de tâches.*

On appelle *linéarisation* de ρ un ensemble composé des mêmes actions totalement ordonnées par la relation de précédence. Par exemple :

$$0 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow \infty$$

Soit $L(\rho)$ l'ensemble des linéarisations de ρ. Par exemple :

$$\begin{aligned}
&\{0 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow \infty,\\
&0 \rightarrow A \rightarrow C \rightarrow B \rightarrow D \rightarrow \infty,\\
&0 \rightarrow B \rightarrow C \rightarrow A \rightarrow D \rightarrow \infty,\\
&0 \rightarrow B \rightarrow C \rightarrow D \rightarrow A \rightarrow \infty,\\
&\ldots\}
\end{aligned}$$

Nous sommes à présent en mesure d'exprimer formellement notre problématique.

Soit $\mathscr{P} = (s_0, g, A)$ un problème de planification où s_0 est l'état initial, g est le but et A est l'ensemble des actions possibles (en pratique exprimées sous la forme de méthodes M et d'opérateurs O). Résoudre ce problème signifie trouver un ensemble ρ (en pratique, un arbre de tâches) tel que :

$$\forall \pi \in L(\rho), g \subseteq \gamma(s_o, \pi)$$

Intuitivement, la résolution du problème de planification consiste à calculer l'ensemble des séquences d'actions (linéarisations) réalisables par l'utilisateur pour atteindre son but.

6.3.4 Procédure de composition d'IHM par planification

Notre formalisation du problème de composition a permis de définir une solution par planification au problème de composition dynamique. Nous allons maintenant proposer une procédure permettant de calculer cette solution. Cette procédure est implémentée dans le planificateur *Compose Planner*. Elle se fonde sur les concepts suivants.

La racine de l'arbre de tâches calculé est la *tâche but*. La projection ρ solution est en pratique un ensemble d'actions représentant un modèle de tâches. Nous notons cet ensemble $\rho(t_u)$ représentant le modèle de tâches de racine t_u. Notre approche pour calculer $\rho(t_u)$ est de trouver une solution à partir des solutions $\rho(t_1), \rho(t_2), \ldots, \rho(t_n)$ telles que $t_1, t_2, \ldots, t_n$ sont les sous-tâches de t_u.

Effets et préconditions d'une tâche exécutable

Une fois $\rho(t_u)$ calculé pour exécuter t_u, on peut connaître les effets et les préconditions de t_u. Soit t_u une tâche et $\rho(t_u)$ sa solution. On appelle les *effets d'une tâche exécutable* t_u l'ensemble des effets engendrés par l'exécution de cette tâche, et les *préconditions d'une tâche exécutable* t_u l'ensemble des préconditions qui ont été satisfaites pendant l'exécution de t_u. Plus formellement, pour définir les effets et les préconditions de t_u deux cas sont à considérer selon si t_u est primitive ou non :

- Si t_u est primitive, alors t_u est une action a. Les effets et les préconditions de cette tâche exécutable sont les effets et les préconditions de l'action a. Formellement, si $\rho(t_u)$ est solution et t_u primitive, alors $effect(t_u) = effect(a)$ et $precond(t_u) = precond(a)$.

- Si t_u est non primitive, alors t_u est une méthode m totalement instanciée. Les effets et les préconditions de cette tâche exécutable sont l'union des effets de ses sous-tâches. Formellement, soient $t_1, \ldots, t_n$ les sous-tâches de t_u dans $\rho(t_u)$, si $\rho(t_u)$ est solution et t_u non primitive, alors $effect(t_u) = effect(t_1) \cup \cdots \cup effect(t_n)$ et $precond(t_u) = precond(t_1) \cup \cdots \cup precond(t_n)$.

Actions et tâches exécutables indépendantes

Deux actions sont *indépendantes* si et seulement si :
- $effect^-(a_1) \cap (precond^+(a_2) \cup effect^+(a_2)) = \emptyset$ et $effect^+(a_1) \cap (precond^-(a_2)) \cup effect^-(a_2)) = \emptyset$ et,

– $effect^{-}(a_2) \cap (precond^{+}(a_1) \cup effect^{+}(a_1)) = \emptyset$ et $effect^{+}(a_2) \cap (precond^{-}(a_1)) \cup effect^{-}(a_1)) = \emptyset$.

Intuitivement, deux actions sont *indépendantes* si elles n'ont pas d'effets pouvant empêcher l'application de l'autre. Dans ce cas, on dit que ces actions ne sont pas en *conflit*.

A partir de la définition d'actions indépendantes, la connaissance des effets et des consommations d'une tâche exécutable permet de savoir si deux tâches exécutables en parallèle peuvent être en conflit, c'est-à-dire si ces tâches ne sont pas exécutables en parallèle de manière indépendante. Formellement, deux tâches exécutables t_1 de $\rho(t_1)$ et t_2 de $\rho(t_2)$ sont *indépendantes* si et seulement si :

– $effect^{-}(t_1) \cap (precond^{+}(t_2) \cup effect^{+}(t_2)) = \emptyset$ et $effect^{+}(t_1) \cap (precond^{-}(t_2)) \cup effect^{-}(t_2)) = \emptyset$ et,
– $effect^{-}(t_2) \cap (precond^{+}(t_1) \cup effect^{+}(t_1)) = \emptyset$ et $effect^{+}(t_2) \cap (precond^{-}(t_1)) \cup effect^{-}(t_1)) = \emptyset$.

Intuitivement, deux tâches exécutables sont indépendantes si un effet engendré par l'exécution d'une action d'un modèle de tâches n'empêche pas l'exécution d'une action de l'autre. Par exemple, si deux modèles de tâches en parallèle ont chacun une action correspondant à une IHM vocale mobilisant les enceintes, alors ces deux modèles de tâches sont en conflit.

Notre problème est donc de construire une IHM telle que, quelles que soient les tâches du modèle de tâches calculé, cette IHM mène de l'état initial à l'état but. Si t_u est la tâche but exécutable du modèle de tâches $\rho(t_u)$, alors il existe un plan solution $\pi = \langle a_1, \ldots, a_k \rangle$ permettant d'exécuter t_u. Soit s l'état courant. L'état obtenu à partir de s à la fin de l'exécution de t_u est défini par :

$$\gamma(s, t_u) = \begin{cases} s, & \text{si } k = 0 \\ \gamma(\gamma(s, a_1), \langle a_2, \ldots, a_k \rangle), & \text{si } k > 0 \end{cases}$$

Solution

Nous sommes à présent en mesure d'exprimer la solution implémentée.

Soit $\mathscr{P} = (s_0, t_u, 0, M)$ un problème de planification où s_0 est l'état initial, t_u est la tâche but du modèle de tâches à calculer et O et M sont l'ensemble des opérateurs et des méthodes.

La tâche but t_u est décomposable récursivement en sous-tâches $t_1, \ldots t_n$ telles que $t_1, \ldots t_n$ sont les sous-tâches de t_u dans $\rho(t_u)$. Pour calculer l'ensemble des linéarisations, la procédure de composition d'IHM est maintenant en mesure de calculer une seule linéarisation et de vérifier que les tâches en parallèle ne sont pas en conflit pour décomposer le problème en sous-problèmes. On peut ainsi définir récursivement si $\rho(t_u)$ est solution à partir des solutions de ses sous-tâches. Pour cela, il faut considérer les cas où t_u est primitive ou non.

Formellement, $\rho(t_u)$ est solution au problème $\mathscr{P} = (s_0, t_u, O, M)$ si et seulement si :

– Cas 1 : si t_u est primitive, alors t_u est une action a. Si a est applicable dans s_0 alors, $\rho(t_u) = \{a\}$ est solution au problème $\mathscr{P} = (s_0, t_u, O, M)$.

  Intuitivement, si t_u est primitive et applicable dans s_0, l'ensemble des linéarisations permettant de trouver une solution est constitué d'un seul élément a.

– Cas 2 : si t_u est non primitive, alors t_u est une méthode totalement instanciée m. Il faut considérer les cas où m est de type séquence ou parallélisme.

  • Cas 2 a : m est de type séquence. Soient $\langle t_1, t_2, \ldots, t_n \rangle$ les éléments de $subAct(m)$. $\rho(t_u)$ est solution au problème $\mathscr{P} = (s_0, t_u, O, M)$ si et seulement si :
    – m est applicable dans s_0 ;
    – $\forall t_i \in subAct(m)$, $\rho(t_i)$ est solution au problème de planification $\mathscr{P} = (\gamma(s_0, t_{i-1}), t_i, O, M)$ avec $\gamma(s_0, t_{i-1}) = s_0$ si $i = 1$.

    Intuitivement, si t_u est une méthode de type séquence, l'ensemble des linéarisations permettant de trouver une solution est constitué d'une seule linéarisation. Trouver une solution revient donc à trouver une linéarisation des sous-tâches de t_u.

    Par exemple, dans la version 1, la méthode *Get_medical_assistance* est solution si elle est applicable dans s_0 et si la linéarisation de ses sous-tâches *Choose_the_city*, *Choose_the_type_of_assistance* et *Contact_the_doctor* sont solution dans les états respectifs s_0, s_1 et s_2 (figure 6.16).

  • Cas 2 b : m est de type parallélisme. Soient $\langle t_1, t_2, \ldots, t_n \rangle$ les éléments de $subAct(m)$. Formellement, $\rho(t_u)$ est solution au problème $\mathscr{P} = (s_0, t_u, O, M)$ si et seulement si :

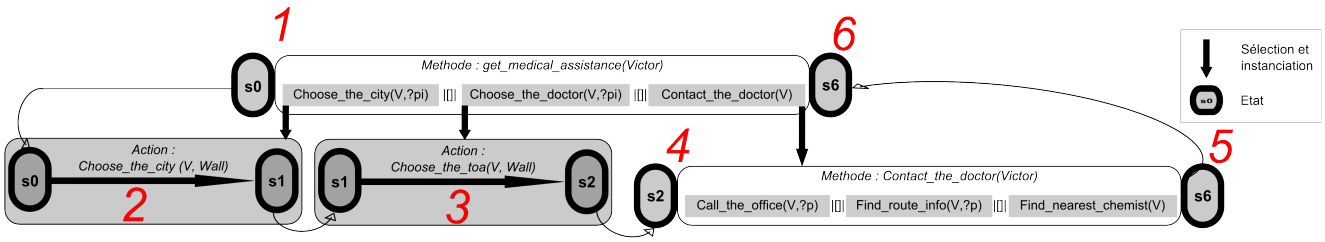


FIGURE 6.16 – *Exemple de solution pour la séquence.*

- m est applicable dans s_0 ;
- $\forall t_i \in subAct(m)$, $\rho(t_i)$ est solution au problème de planification $\mathscr{P} = (s_0, t_i, O, M)$;
- $\forall t_i, t_j \in subAct(m), i \neq j$, t_i est indépendante avec t_j.

Intuitivement, si t_u est une méthode de type parallèle, il faut vérifier que ses sous-tâches sont exécutables dans l'état courant et qu'elles ne sont pas en conflit entre elles.

Par exemple, dans la version 1, la méthode *Contact_the_doctor* est solution si elle est applicable dans s_2 et si ses sous-tâches $Call\_the\_office$, $Find\_route\_info$ et *Find_nearest_chemist* sont solution dans s_2 et ne sont pas en conflit (figure 6.17). L'état s_6 obtenu par l'application de *Contact_the_doctor* est calculé par application successive de $Call\_the\_office$ sur s_2, $Find\_route\_info$ sur s_3 et *Find_nearest_chemist* sur $s_{4'}$.

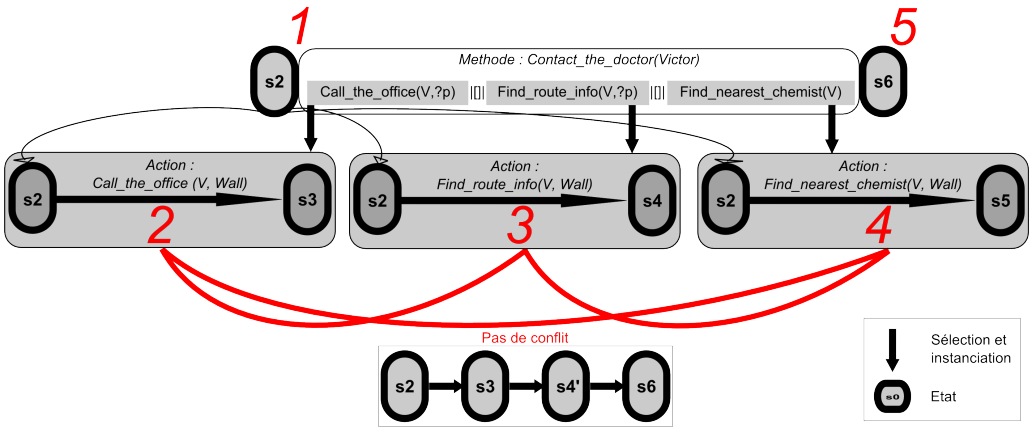


FIGURE 6.17 – *Exemple de solution pour le parallélisme.*

Algorithme de composition d'IHM par planification

L'algorithme est une application de cette définition récursive. Il est implémenté dans le planificateur *Compose Planner*. Il prend en entrée un problème de planification $\mathscr{P} = (s_0, t_u, O, M)$. Si t_u est primitive, t_u est une action a. Si a est applicable alors la solution est $\rho(t_u) = \{a\}$. Si t_u est non primitive, t_u est une méthode m. La procédure décompose le problème en sous problèmes selon les sous-tâches $t'_1 \ldots, t'_n$ à instancier. Pour chaque sous-tâche de t_u, l'étape « ChoixNonDeterministe t_i » correspond à la sélection d'une instance t_i de t'_i. Si la procédure échoue à la suite de la sélection d'une instance de t'_i, elle en choisira une autre. Dans le pire des cas, toutes les actions sont testées. Si aucune action ne convient, le choix non déterministe renvoie un échec. Si m est de type séquence, $\rho(t_u)$ est solution si toutes ses sous-tâches t_i sont exécutables. Si m est de type parallèle, $\rho(t_u)$ est solution si toutes les sous-tâches sont exécutables et indépendantes entre elles. Si c'est le cas, l'union des solutions de ces

sous-tâches est solution de $\mathscr{P} = (s_0, t_u, O, M)$.

Algorithme 3 : **Compose-IHM** (s_0, t_u, O, M) - Procédure de composition d'IHM par planification

```
Data : (s_0, t_u, O, M) un problème de planification HTN.
Result : ρ(t_u) un ensemble d'actions (modèle de tâches solution)
         s'il existe, échec sinon.
if t_u est une action a then
    if a applicable dans s_0 then
        return {a};
    else
        return échec;
    end
else
    //t_u est une méthode m
    if m applicable dans s_0 then
        foreach t'_i ∈ subAct(m) do
            ChoixNonDeterministe t_i instance de t'_i;
            ρ(t_i) ← Compose-IHM(s_0, t_i, O, M);
            if ρ(t_i) = échec then
                return échec;
            end
            if type(m) = " > " then
                if i>1 then
                    s_0 ← γ(s_0, t_{i-1});
                end
                ρ(t_u) ← ρ(t_u) ∪ ρ(t_i);
            else
                //type(m) = "||"
                if t_i indépendant t_x, ∀t_x ∈ ρ(t_u) then
                    ρ(t_u) ← ρ(t_u) ∪ ρ(t_i);
                else
                    return échec;
                end
            end
        end
    end
    return ρ(t_u);
end
```

Dans cette présentation de l'algorithme, nous avons considéré pour simplifier que le problème se définissait avec une tâche t_u qui est une action ou une méthode instanciée. En pratique, cette tâche est un opérateur ou une méthode non instancié. La procédure *Compose-IHM* est exécutée pour chaque instance t_u de cette tâche. Si le planificateur trouve une

solution, il la renvoie. Si aucune instance t_u n'est solution, la procédure échoue.

L'algorithme *Compose-IHM* est illustré sur le cas d'étude (version 1) dans la figure 6.18. Une méthode « *Get_medical_assistance* » est sélectionnée, instanciée puis appliquée. Elle se décompose en sous-tâches « *Choose_the_city* », « *Choose_the_type_of_assistance* », et « *Contact_the_doctor* » sélectionnées et instanciées successivement. L'action « *Choose_the_city* » permet de calculer s_1 à partir de s_0. L'action « *Choose_the_type_of_assistance* » permet de calculer s_2 à partir de s_1. Une méthode « *Contact_the_doctor* » est à son tour sélectionnée et instanciée pour se décomposer en trois actions « *Call_the_office* », « *Find_route_info* » et « *Find_nearest_chemist* ». Ces actions sont applicables dans s_2, l'algorithme vérifie donc qu'il n'existe pas de conflit entre ces tâches exécutables. Une fois assurée de l'absence de conflit, la procédure calcule l'état s_6 par l'application de la séquence d'actions sur s_2. Ainsi, s_3 est produit par l'application de « *Call_the_office* » sur s_2, s'_4 par « *Find_route_info* » sur s_3 et s_6 par « *Find_nearest_chemist* » sur $s_{4'}$. L'état s_6 est l'état final satisfaisant le but.

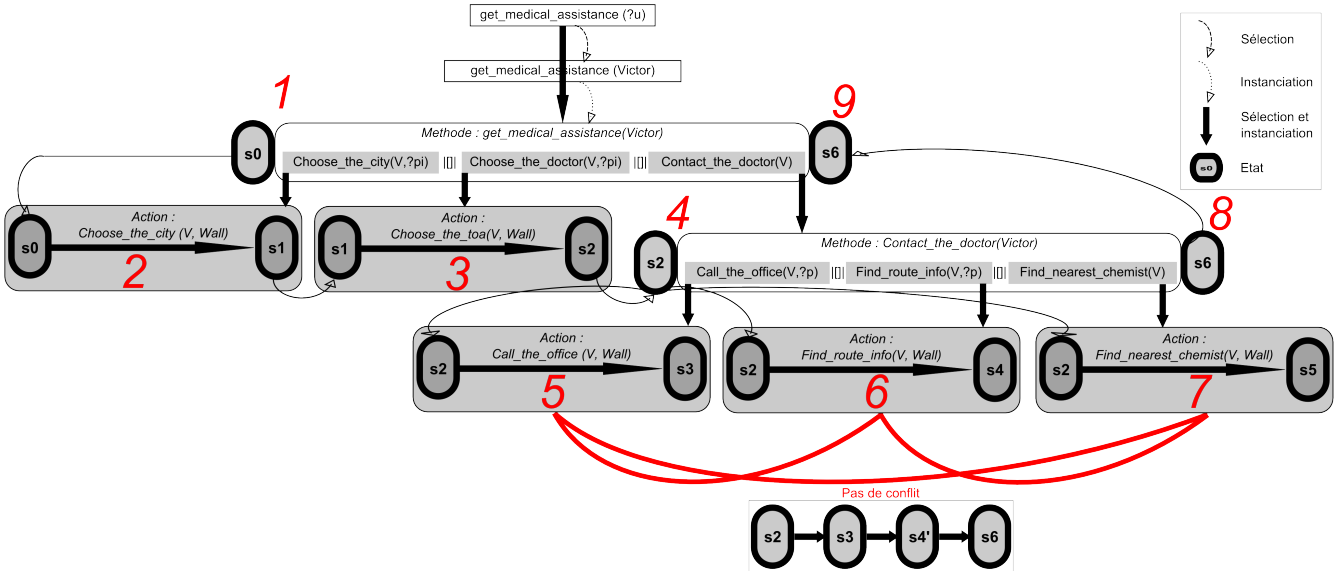


FIGURE 6.18 – *Illustration de la procédure de l'algorithme pour la version 1 du cas d'étude.*

6.4 Conclusion

Nous avons choisi d'étudier la voie de la planification HTN pour sa capacité à exprimer la composition dynamique d'IHM. Cette étude a démontré l'applicabilité des algorithmes de planification à la composition d'IHM. Néanmoins, elle a aussi mis en avant une limite pour notre problématique : seul l'opérateur « puis » peut être exprimé dans le plan solution. Ce constat nous a amenés à proposer une formalisation du problème de composition dynamique d'IHM qui permet d'exprimer le parallélisme.

Nous avons proposé un algorithme pour calculer une séquence d'actions solution tout en vérifiant que les tâches en parallèle ne sont pas en conflit. La notion de conflit permet de s'assurer de la validité de l'IHM par rapport au contexte d'usage et aux propriétés ergonomiques à privilégier. Par exemple, deux IHM vocales ne peuvent pas être utilisées en parallèle car leurs effets sont en conflit.

Notre solution a pour limite principale de ne pas exprimer l'opérateur entre tâches « choix ». Nous identifions différentes perspectives pour intégrer cet opérateur dans la solution :

- Hypothèse d'expressivité équivalente : on peut supposer que si un concepteur souhaite proposer un choix entre plusieurs IHM à l'utilisateur, c'est parce que ces choix ont les mêmes effets. Par exemple, pour s'identifier, l'utilisateur a le choix d'entrer son login ou son nom. Dans les deux cas, ces actions ont pour effet que l'utilisateur est identifié. Ainsi, l'ajout de ce choix (s'il est applicable) ne remet pas en cause le plan solution trouvé.

- Modification de l'algorithme pour calculer autant de plans solutions (modèle de tâches) que de choix. Cela peut entraîner rapidement un fort coût.

- Intégration du choix une fois le plan solution connu en proposant éventuellement des stratégies d'adaptation si le plan est remis en cause.

Démonstrateur *Compose* 7

Résumé

Ce chapitre se structure en deux parties. Dans la première, nous présentons le planificateur *Compose planner* implémenté pour la composition d'IHM. Ce planificateur compose des tâches IHM pour produire un modèle de tâches exécutable dans le contexte d'usage courant et dans le respect des critères d'ergonomie à privilégier. Dans la deuxième partie, nous décrivons le démonstrateur *Compose*. Ce démonstrateur compose des IHM conformément à la solution calculée par le planificateur.

Publications associées : TSI'11 (Gabillon et al. 2011a), SEMAIS'11 (Gabillon et al. 2011b), JFPDA'08 (Gabillon et al. 2008)

Sommaire

Introduction

Ce chapitre présente le prototype **Compose**, un démonstrateur de composition d'IHM par planification. Ce prototype calcule, à partir d'un objectif utilisateur exprimé en langage naturel, une IHM composite. *Compose* utilise un planificateur développé pour la composition d'IHM : le planificateur *Compose planner*. Ce planificateur calcule un modèle de tâches apte à satisfaire l'objectif utilisateur dans le contexte d'usage courant. Chaque tâche de ce modèle de tâches est représentée par une IHM. A partir du modèle de tâches et des IHM préexistantes, *Compose* sélectionne les IHM préexistantes à composer conformément aux consignes du planificateur.

Nous présentons le planificateur *Compose Planner* puis le démonstrateur *Compose* intégrant ce planificateur.

7.1 Planificateur *Compose Planner*

Le planificateur calcule, à partir des descriptions des IHM, un modèle de tâches répondant à l'objectif utilisateur. Ce planificateur est nommé *Compose Planner*, il est implémenté en JAVA.

En pratique, le planificateur prend en entrée un opérateur ou une méthode non instancié noté t'_u. Pour chaque t_u instance de t'_u, la procédure exécute l'algorithme $Compose\text{-}IHM(s_0, t_u, O, M)$ (algorithme 3 présenté dans le chapitre 6). Si ce dernier trouve une solution, il la renvoie. Sinon, il renvoie *échec*.

Algorithme 4 : **ComposePlanner (**s_0, t'_u, O, M**)** - Algorithme du planificateur Compose

Data : (s_0, t'_u, O, M).
Result : $\rho(t_u)$ un ensemble d'actions (modèle de tâches) s'il existe, *échec* sinon.
foreach t_u *instance de* t'_u **do**
  if $Compose\text{-}IHM(s_0, t_u, O, M) \neq$ *échec)* **then**
    return $\rho(t_u)$;
  end
end
return *échec*;

7.2 Prototype *Compose*

Compose est un démonstrateur de composition d'IHM par planification. La figure 7.1 en présente une décomposition fonctionnelle. Seules les fonctions « Model-based composer » et « Code composer » sont implémentées. L'implémentation est effectuée en JAVA.

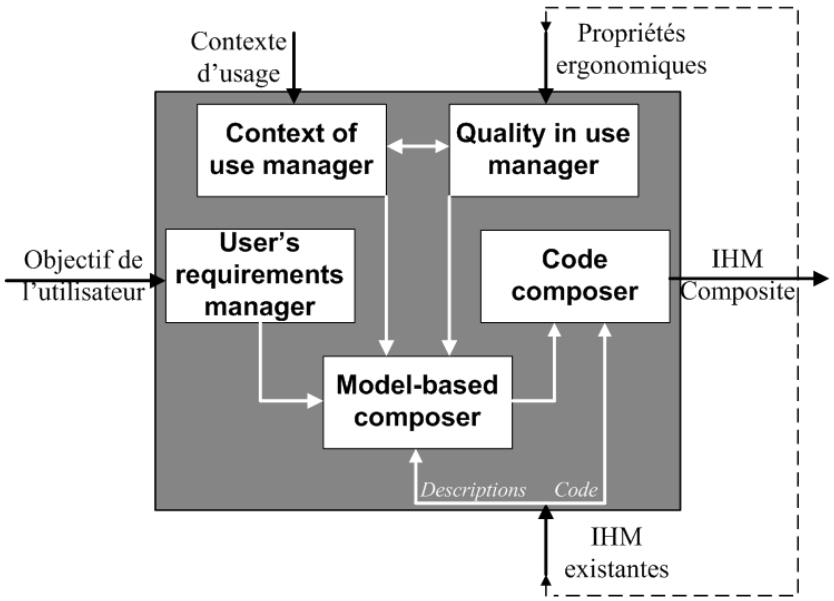


Figure 7.1 – *Décomposition fonctionnelle de Compose.*

La fonction « Context of use manager » est chargée de percevoir et d'interpréter le contexte d'usage courant. La fonction « Quality in use manager » est chargée de gérer les critères d'ergonomie à privilégier (Scapin et Bastien 1997). Dans *Compose*, le contexte d'usage et les critères d'ergonomie sont modélisés manuellement. Ces modèles sont traduits en propositions logiques utilisant les objets du contexte. Par exemple, Victor et le mur numérique sont représentés par des objets. Pour exprimer le fait que Victor a accès au mur, on utilise une proposition « $has(Victor, Mur)$ ». Cette traduction est automatique à partir d'un modèle du contexte d'usage. La fonction « User's requirement manager » est chargée de percevoir l'objectif de l'utilisateur et de l'interpréter en un objectif connu du système. Par exemple, l'objectif « Je veux aller voir un médecin » (figure 7.2) est interprété en un but modélisé par la méthode « Get medical assistance ». Nous utilisons le moteur de recherche open source Lucene implémenté en JAVA.

Les fonctions « Model-based composer » et « Code composer » constituent le cœur de *Compose*. Le « Model-based composer » est effectué par le planificateur. Le « Code composer » utilise le modèle de tâches

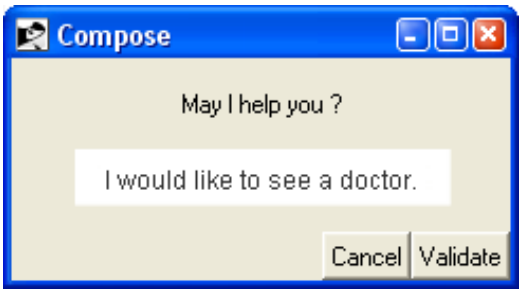


FIGURE 7.2 – *IHM de Compose pour spécifier l'objectif de l'utilisateur*

produit par le planificateur et le traduit en une IHM composite. Une fois que le niveau tâches est composé par le planificateur, les COMETs (Demeure et al. 2008), associées respectivement aux opérateurs ou méthodes instanciés, sont déployées sur la ou les plates-formes sélectionnées par le planificateur. Les COMETs couvrent tous les niveaux d'abstraction d'une IHM. La capitalisation de l'IHM composite est souhaitée par les utilisateurs (Gabillon et al. 2009) pour une raison de stabilité.

La première section présente les IHM existantes utilisées dans le cas d'étude. Ces IHM existent sous la forme de COMETs et de descriptions en planification. La seconde section présente les modèles de tâches calculés par le planificateur selon les versions. Dans une troisième section, le « Code composer » est décrit. Il compose les codes des IHM existantes à partir de leurs modèles composés par le planificateur.

7.2.1 IHM existantes

Les IHM sont implémentées par des COMETs. Chaque COMET est décrite par un opérateur ou une méthode en planification.

Les IHM élémentaires sont décrites par les opérateurs de mêmes noms 7.3 :

- « Choose the doctor » et « Choose the city » : les paramètres sont un utilisateur et une plate-forme. Ces opérateurs requièrent que la plate-forme soit connectée à internet (*internet?p*). Le médecin et la ville sélectionnés sont ajoutés à l'état du monde (*isChosenVictorDoctorChosen* et *isChosenVictorCityChosen*).

- « Call the office », « Find route information » et « Find nearest chemist » : ces opérateurs décrivent respectivement l'IHM (c), (d) et (e). Les paramètres sont un utilisateur et sa plate-forme. Ces opérateurs requièrent que l'utilisateur ait une plate-forme dotée de

capacités téléphoniques ($phone?p$) pour la première ou connectée à internet pour les autres.

Les IHM des composeurs sont décrites par les méthodes de mêmes noms. Dans le cas d'étude, trois COMETs (Figure 7.4) sont utilisées. La COMET (f) est un canevas avec un agencement vertical ($Union = \{Equal, MeetI\}$). Dans la version pour le mur numérique, le canevas contient les COMETs (c), (d) et (e). La COMET (g) représente une séquence temporelle entre COMETs ($Union = \{Meet, EqualI\}$). Dans le cas d'étude, cette COMET est utilisée entre les COMET (a) et (b) et les COMETs (b) et (f). Quand la COMET (a) se termine, la COMET (b) se déploie. Quand la COMET (b) se termine, la COMET (f) se déploie. La COMET (h) représente un onglet entre COMETs ($Union = \{Equals, EqualI\}$). Dans la version pour le Smartphone, le canevas contient les COMETs (c) et (d). Les IHM sont décrites par des méthodes de mêmes noms :

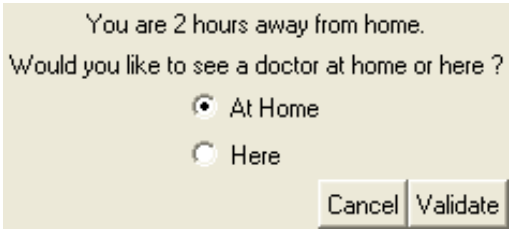
- « Get medical assistance » : les paramètres sont un utilisateur ($?u$). Les sous-tâches sont successivement « Choose the city », « Choose the doctor » et « Contact the doctor ».

- « Contact the doctor » : les paramètres sont un utilisateur et sa plate-forme. La méthode requiert que l'écran de la plate-forme supportant l'interaction soit petit ($isSmallScreen?p$) et le critère « Charge de travail » privilégié dans sa dimension d'actions minimales ($Workload\_Brevity\_MinimalActions$). Cette méthode est utilisée dans les cas où l'espace d'affichage est petit comme la version 2 du cas d'étude. Ses sous-tâches sont parallèlement (concurrence) « Call the office », « Find route information ».

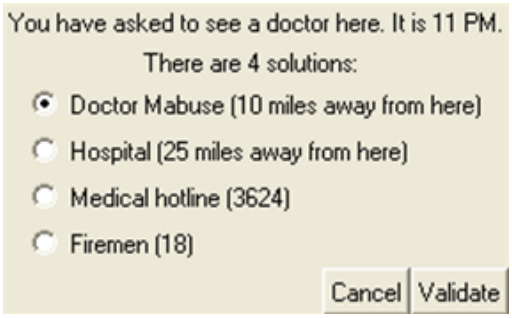
- « Contact the doctor » : les paramètres sont identiques à la méthode précédente. La méthode requiert par contre que l'écran de la plate-forme soit grand ($isLargeScreen?p$) et que le critère « Charge de travail - Densité informationnelle » soit prioritaire ($high\_Workload\_InformationDensity$). Cette méthode est utilisée lorsque l'espace d'affichage est suffisant comme dans les versions 1 et 3 du cas d'étude. Ses sous-tâches sont parallèlement « Call the office », « Find route information » et « Find nearest chemist ».

7.2.2 Modèles de tâches calculés

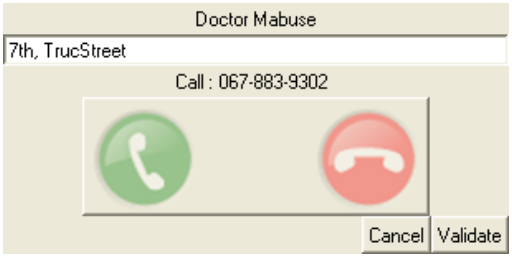
Le « Model-based composer » est le planificateur *Compose Planner* développé pour la composition d'IHM. Il utilise les opérateurs et les mé-



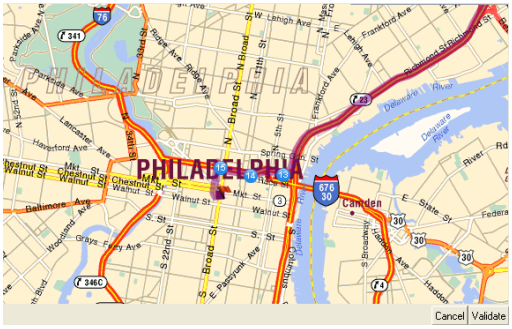
a) COMET « Choose an option » instanciée pour « Choose the city ».



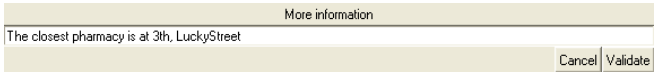
b) COMET « Choose an option » instanciée pour « Choose the doctor ».



c) COMET « Call somebody » instanciée pour « Call the doctor ».

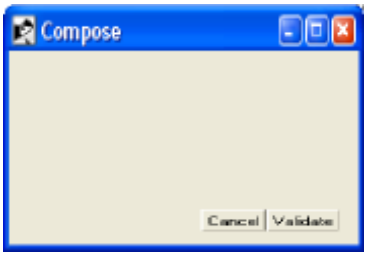


d) COMET « Find route ».

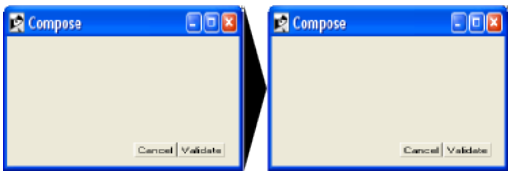


e) COMET « Display text » instanciée pour « Find nearest chemist ».

FIGURE 7.3 – *Cinq IHM élémentaires pour le cas d'étude.*



f) COMET « Vertical layout » instanciée pour « Contact the doctor ».

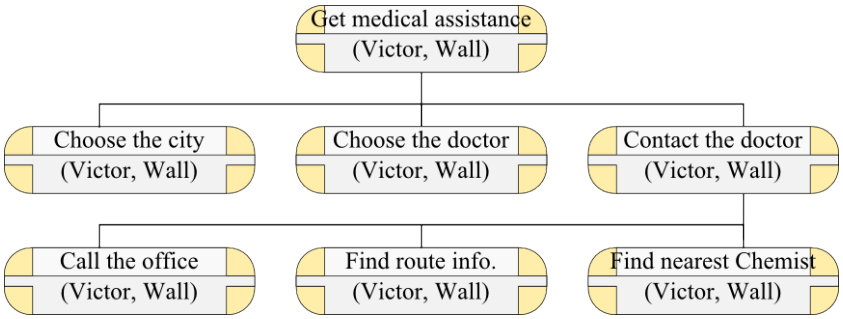


g) COMET « Next » instanciée pour « Get medical assistance ».

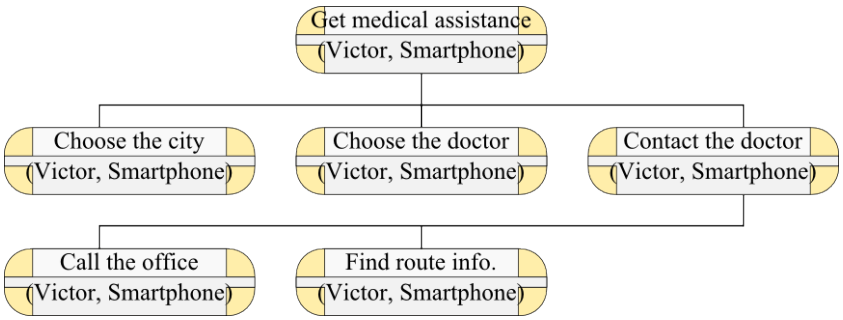
h) COMET « Interleaving » instanciée pour « Contact the doctor ».

FIGURE 7.4 – *Trois IHM d'utilité publique pour le cas d'étude.*

thodes. Il produit un modèle de tâches selon le contexte d'usage courant et l'objectif de l'utilisateur. Pour le cas d'étude, pour le Smartphone et le mur numérique, les modèles de tâches de la figure 7.5 sont calculés par le planificateur. Pour le mur numérique, sept tâches sont composées. Pour le Smartphone, le modèle de tâches obtenu est différent. Seules six tâches sont composées. La tâche optionnelle « Find nearest chemist » est omise et les sous-tâches de « Goto doctor » sont composées dans un onglet. Dans le cas du mur numérique, la méthode instanciée « Contact the doctor » qui a trois sous-opérateurs est sélectionnée car l'écran de la plate-forme est grand (Figure 7.5a). Dans le cas du Smartphone, c'est la méthode instanciée « Contact the doctor » avec deux sous-opérateurs qui est sélectionnée car l'écran est petit (Figure 7.5b).



a) pour le mur numérique.



b) pour le Smartphone.

FIGURE 7.5 – *Modèles de tâches obtenus par le planificateur pour le mur numérique et pour le Smartphone.*

Le modèle de tâches est fourni au « Code composer » qui va remplacer ces opérateurs et méthodes instanciés par les COMETs correspondantes.

7.2.3 Code composer

Le « Code composer » traduit le modèle de tâches obtenu par le planificateur en un modèle de COMETs. A l'exécution, les opérateurs

et les méthodes instanciés sélectionnés sont statiquement associés aux COMETs correspondantes. Ces COMETs sont déployées en fonction des consignes du planificateur. Par exemple, dans *Compose*, la méthode « Get medical assistance » avec l'opérateur séquence est transformée en une COMET. Cette COMET, déployée sur le mur numérique, affiche ses sous-opérateurs dans des fenêtres en séquence. Le « Code composer » compose les COMETs pour produire une IHM composée. Les tâches du modèle de tâches calculées pour le mur numérique sont liées à leurs COMETs respectives (Figure 7.6). La méthode Get medical assistance est liée à la « COMET (g) » qui affichera dans des fenêtres en séquence les « COMETs (a), (b) et (f) ». Une fois que toutes les tâches sont liées, *Compose* fusionne des éléments qui sont identiques (dans l'implémentation, deux éléments sont identiques s'ils ont les mêmes noms) en supprimant ces éléments des sous-IHM. Par exemple, la COMET (f) supprime les boutons de navigation de ses sous-IHM c, d et e. L'IHM composée à l'exécution est déployée pour répondre à l'objectif de l'utilisateur.

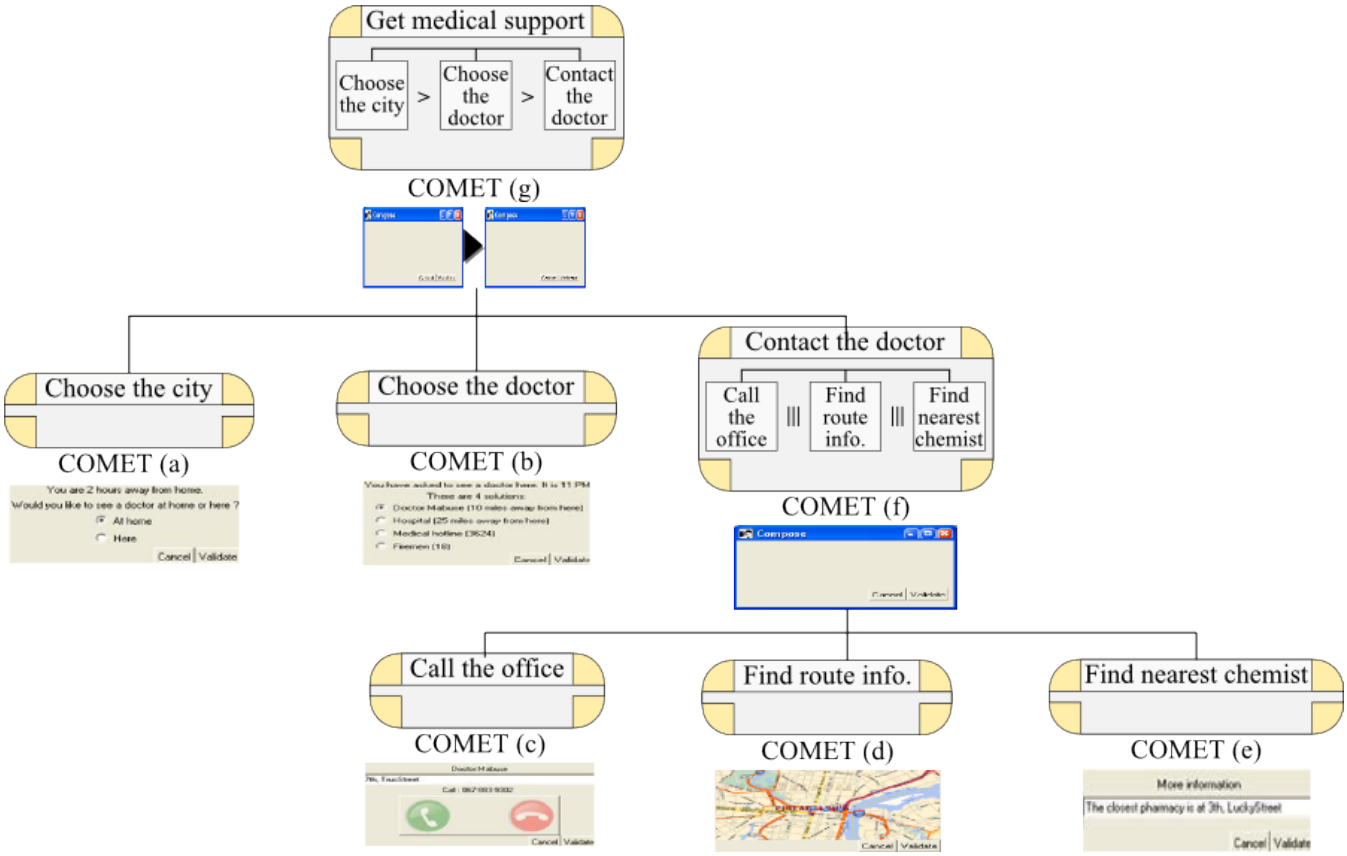
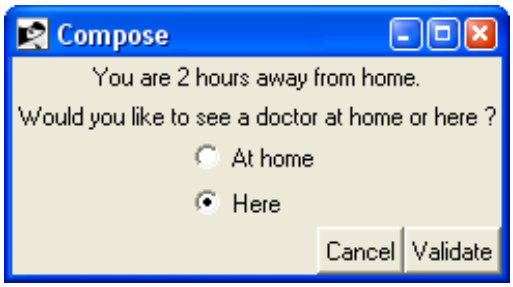


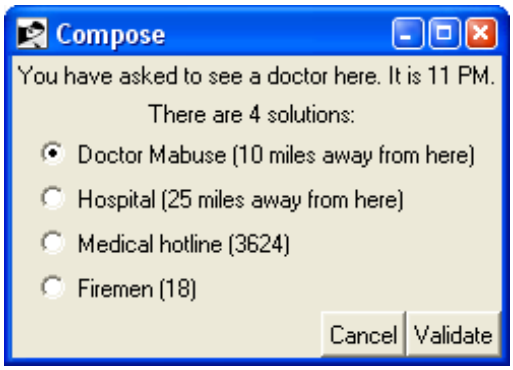
FIGURE 7.6 – *Modèle de tâches calculé pour le mur numérique. Chaque tâche est transformée en une COMET.*

Les IHM pour choisir une ville et choisir un médecin sont compo-

sées en séquence (figure 7.7) dans toutes les versions du cas d'étude. Par contre, différentes IHM sont composées pour contacter un médecin selon la version du cas d'étude. Pour la version 1, les trois IHM (appeler un médecin, afficher un itinéraire et la pharmacie) sont composées sur le mur numérique (figure 7.8). Dans la version 2, l'affichage de la pharmacie la plus proche est omis pour s'adapter à la taille de l'écran du Smartphone (figure 7.9). Dans la version 3, seule l'IHM pour appeler un médecin est déployée sur le Smartphone, les autres sont déployées sur le mur numérique (figure 7.10).



a) Deux possibilités de lieu pour l'assistance médicale.



b) Quatre solutions médicales.

FIGURE 7.7 – *Deux IHM composées en contexte par Compose pour affiner le besoin de l'utilisateur « Get medical assistance ».*

7.3 CONCLUSION

Le prototype *Compose* illustre la composition dynamique par planification sur notre cas d'étude. Il utilise le planificateur *Compose Planner* développé pour la composition dynamique d'IHM. Les descriptions comme la modélisation des IHM sont laissées au concepteur. Ainsi, ce dernier peut décrire lui-même les composeurs qu'il souhaite. Il est garant des contraintes ergonomiques qu'il spécifie.

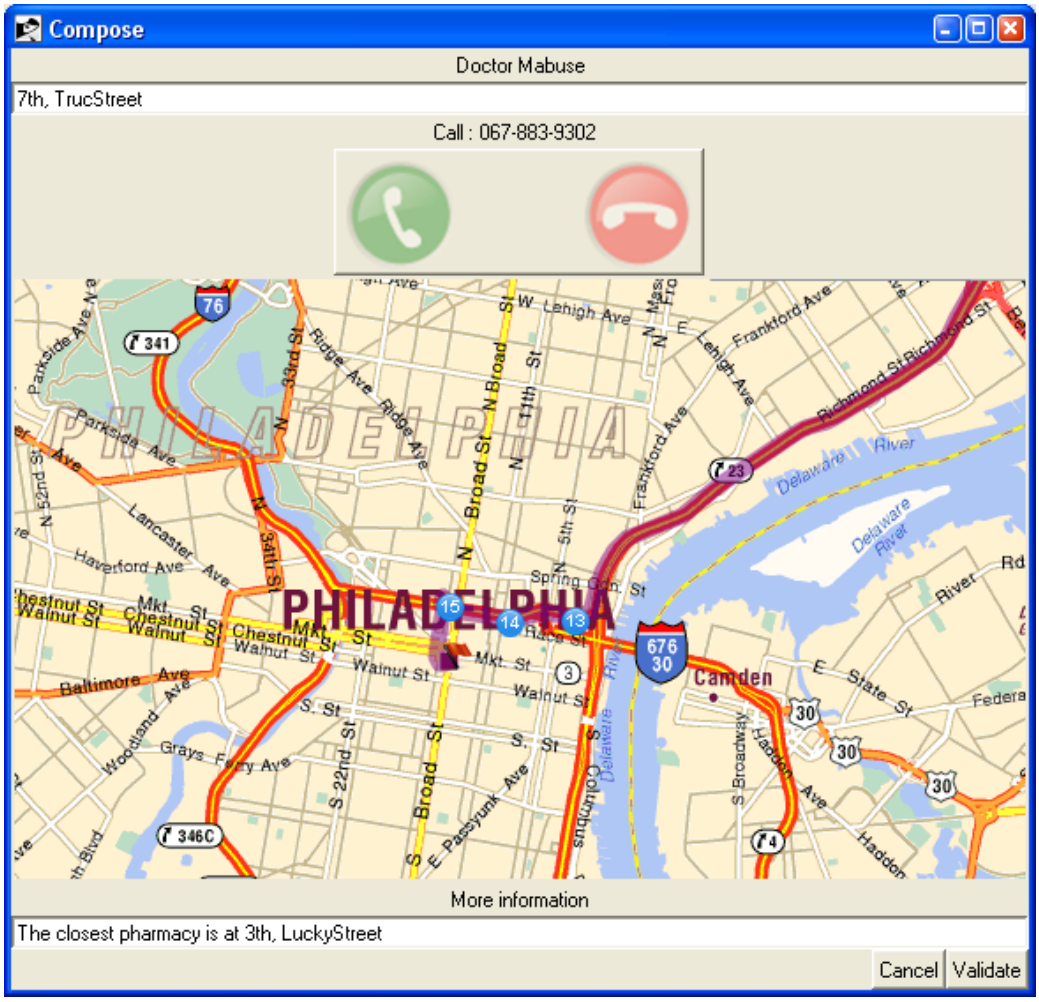


FIGURE 7.8 – *IHM composée pour la version 1 sur le mur numérique.*

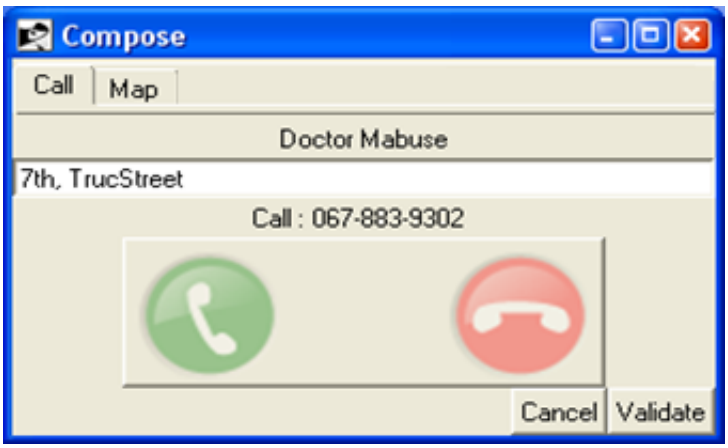
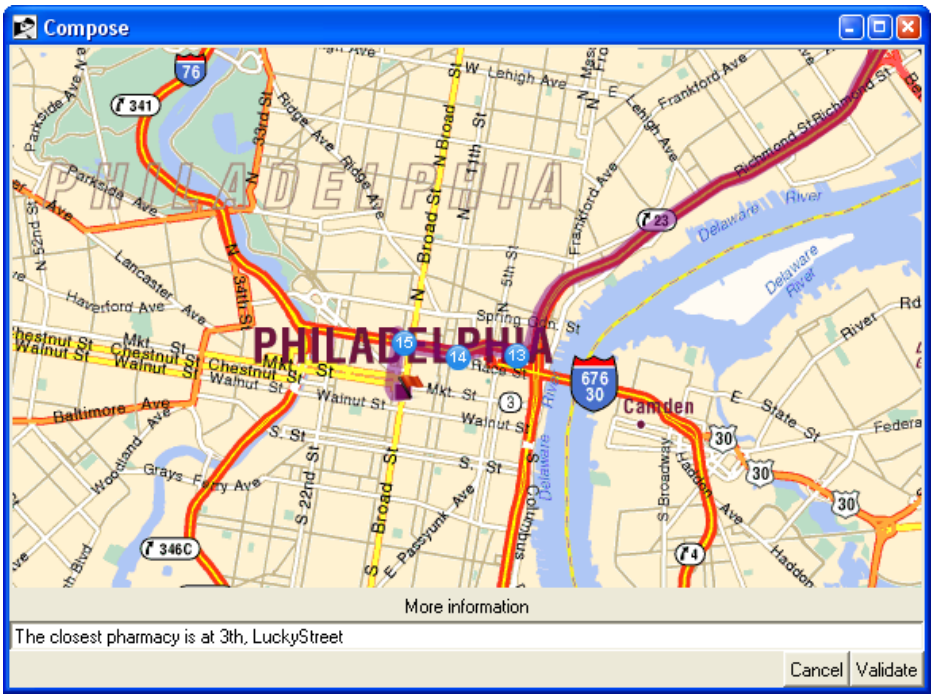
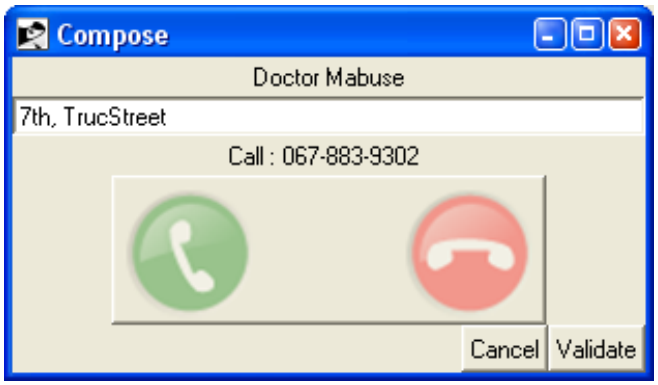


FIGURE 7.9 – *IHM composée pour la version 2 sur le Smartphone.*



a) sur le mur numérique.



b) sur le Smartphone.

FIGURE 7.10 – *IHM composée pour la version 3 sur le mur numérique et le Smartphone.*

Conclusion

8

Ce travail a traité de la composition d'IHM pour répondre à un objectif utilisateur émergent. Son originalité est de composer le modèle de tâches dynamiquement. La composition s'appuie sur une base d'IHM réutilisables, un contexte d'usage donné et des propriétés ergonomiques à privilégier décrites par le concepteur. L'approche que nous avons explorée est la planification automatique.

Nos recherches ont permis de démontrer la faisabilité de cette approche pour la composition d'IHM. Nous en rappelons les contributions puis en présentons les perspectives à moyen et long terme.

8.1 Résumé des contributions

Dans une démarche centrée utilisateur, nous avons tout d'abord réalisé une **étude sociologique** sous la forme d'interviews et de groupes de discussion dans le but de recueillir les exigences des utilisateurs vis-à-vis d'un système de composition dynamique tel que *Compose*.

Les principales contributions de cette étude sont les suivantes :

- L'*intérêt des utilisateurs pour Compose :* un système tel que *Compose* est majoritairement apprécié par les utilisateurs (23 sujets sur 26).

- La *variété des cas d'étude :* les utilisateurs apprécient *Compose* dans des situations d'exception (panne de voiture, assistance médicale, etc.) et du quotidien (simplification de recherche, assistance technique).

- L'*incidence pour le système de composition :*
  - sur la décomposition fonctionnelle : la capitalisation des systèmes interactifs composés doit être prévue ;
  - sur le mode d'interaction : le langage naturel écrit est le moyen préféré des utilisateurs pour spécifier leur objectif ;

- sur les services : les utilisateurs veulent connaître les services sélectionnés et en particulier leur caractère commercial ou non.

D'un point de vue conceptuel, notre objectif a été de comprendre comment composer dynamiquement des IHM, ainsi que d'établir des patrons de composition à tous niveaux d'abstraction. Nous avons contribué à cet objectif en proposant :

– Un **espace problème** de la composition dynamique d'IHM structuré en trois classes de problème selon qu'ils relèvent du but, des éléments à composer ou du calcul de la composition. Il permet de décomposer le problème de la composition dynamique pour caractériser l'état de l'art en composition d'IHM.

– Une **taxonomie des composeurs** caractérisant les opérateurs entre IHM. Elle se singularise par sa prise en compte du niveau tâches pour la composition d'IHM. Cette taxonomie constitue une aide au concepteur pour concevoir des composeurs en séparant les préoccupations en niveaux d'abstraction.

D'un point de vue logiciel, notre approche a été l'application à la composition d'IHM des techniques de l'intelligence artificielle, et plus particulièrement les algorithmes de planification automatique. Cette thèse permet de démontrer la faisabilité algorithmique de l'approche. Elle a en particulier contribué aux points suivants :

– Une **application des algorithmes HTN pour la composition d'IHM :** cette étude a permis de constater le manque d'expressivité du plan par rapport à nos besoins. En effet, le plan solution est une séquence d'actions. Or, le parallélisme est nécessaire pour valider l'exécution en concurrence de deux IHM.

– Une **adaptation des algorithmes HTN pour la composition d'IHM :** nous avons proposé une procédure de planification pour composer des IHM en séquence et en parallèle. La composition d'IHM en parallèle repose sur l'idée que ces IHM ne doivent pas être en conflit. Les algorithmes de planification permettent de garantir que l'IHM est compatible avec le contexte d'usage courant et les propriétés ergonomiques à privilégier.

Ces contributions logicielles sont illustrées par le prototype *Compose* utilisant le planificateur que nous avons développé pour la composition d'IHM : *Compose Planner*.

8.2 Perspectives

A l'instar de la composition dynamique du NF, la composition dynamique d'IHM est une problématique complexe qui est loin d'être résolue. Ce travail engendre de nombreuses perspectives à moyen terme et à long terme.

8.2.1 A moyen terme

Nos contributions bénéficieraient des extensions suivantes à moyen terme :

- **Ajout des décorations dans notre algorithme :** les décorations de tâches ne sont pas prises en compte par notre planificateur. Il serait intéressant d'ajouter ces décorations dans la définition d'actions et de méthodes pour que le planificateur puisse les intégrer dans le calcul d'une solution. Par exemple, si une tâche a pour précondition une connexion internet et comme effet de supprimer cette connexion à internet, alors elle ne pourra être itérative.

- **Évaluation des performances de notre algorithme :** notre algorithme de planification est spécifiquement développé pour la composition d'IHM. Pour cette raison, il est difficile de le comparer avec d'autres algorithmes de planification. Cependant, il serait nécessaire d'évaluer les performances de l'algorithme en terme de temps d'exécution en faisant varier le nombre d'actions, de méthodes, la profondeur du modèle de tâches à calculer, etc. La difficulté pour ce genre d'évaluation est le besoin d'une base de problème de planification.

- **Adaptation du langage de planification :** le langue utilisé par notre planificateur est un langage de logique du premier ordre. Même si cette représentation symbolique permet de raisonner à haut niveau d'abstraction, le raisonnement sur la qualité de l'IHM est complexe et peut nécessiter l'utilisation d'un langage numérique pour décrire précisément le contexte. Cette extension entraînerait l'adaptation de la notion de conflit pour l'exprimer dans un langage numérique.

- **Ajout de la sémantique :** nous avons supposé dans notre travail l'alignement sémantique effectué. Cependant, il est nécessaire d'ajouter la description sémantique des IHM. Les travaux sur la composition des services Web sémantiques ont démontré la capacité des algorithmes de planification à être couplés à OWL-S pour

exprimer cette sémantique.

- **Hétérogénéité des IHM à composer :** notre travail suppose que les IHM à composer sont décrites de manière homogène. En pratique, les IHM peuvent être conçues à différents niveaux d'abstraction ou peuvent être incomplètes. Il est donc nécessaire d'étudier l'homogénéisation des IHM à composer. Cette homogénéisation doit être effectuée à la suite du calcul de planification, une fois que les IHM à composer ont été sélectionnées.

- **Capitalisation de l'IHM composée :** la capitalisation de l'IHM composée signifie la nécessité de conserver l'IHM composite et de générer sa description en planification automatiquement. L'IHM composite doit être décrite par une action pour être réutilisable comme telle par le planificateur.

- **Ajout du choix :** le modèle de tâches solution calculé ne produit pas de choix pour l'utilisateur. Les choix sont exprimés par l'opérateur entre tâches de même nom en IHM. Cependant, pour trouver une solution, il est nécessaire de calculer autant de modèles de tâches que de choix possibles pour l'utilisateur. Calculer ce choix engendrerait donc un fort coût de calcul. Ce coût peut être évité en proposant ces choix à l'utilisateur.

8.2.2 A long terme

Nos contributions conceptuelles et logicielles pourraient bénéficier des extensions suivantes à long terme :

- **Méta-IHM :** une méta-IHM pourrait être développée pour éventuellement faire intervenir l'utilisateur directement dans le calcul du planificateur. Cette méta-IHM permettrait de satisfaire l'exigence de qualité et simplifier la combinatoire. Son avantage est la possibilité de recueillir les habitudes des utilisateurs. Le système de composition aurait ainsi la capacité d'« apprendre » leurs préférences. La méta-IHM permettrait aussi de répondre à la méfiance des utilisateurs pour les systèmes entièrement automatisés.

- **Propriétés de la composition :** la composition logicielle est identifiée comme un problème important et d'actualité. Le raisonnement compositionnel sur les propriétés des systèmes assemblés et composites est l'un des grands défis à relever pour composer dynamiquement des systèmes (A. Finkelstein, UCL, London,

décembre 2009). Des propriétés comme la composabilité et la compositionnalité (De Roever et al. 2001) sont définies en système (Krakowiak 2010). La composabilité signifie que les propriétés de chaque composant sont préservées dans le système composé. La compositionnalité signifie que les propriétés du système composé se déduisent de celles des composants et des règles de composition. La planification est une approche formelle qui facilite l'étude de ces propriétés.

- **Ergonomie :** la spécification et la satisfaction des exigences ergonomiques sont une difficulté majeure. Les planificateurs raisonnent sur l'ergonomie sous la forme de prédicats décrits par le concepteur. Cependant, il est nécessaire d'automatiser la prise en compte des critères ergonomiques pour composer dynamiquement des IHM ergonomiques. Les critères d'ergonomie s'encrant au niveau tâches peuvent être intégrés dans notre algorithme de planification. Au contraire, les critères d'ergonomie s'encrant à plus bas niveau d'abstraction doivent être intégrés dans les composeurs. Ici, la difficulté est de taille, les critères d'ergonomie souffrant d'un manque de formalisation.

- **Couplage avec la composition du NF :** dans cette thèse, nous nous sommes concentrés sur l'IHM sans considérer le NF. La composition dynamique de systèmes nécessite de composer dynamiquement le NF, l'IHM et les relations entre les deux. Cette étude sur le couplage entre ces travaux et ceux de la composition de services sera facilitée par l'utilisation de la même technique : la planification automatique.

- **Recomposition :** notre algorithme de planification doit être étendu pour lui permettre de calculer une IHM en tenant compte de la précédente IHM composite, et ainsi produire un résultat stable pour l'utilisateur (critère « Homogénéité / Cohérence »).

- **Passage à l'échelle :** le temps de calcul dépend du nombre d'IHM à composer. L'approche par COMETs favorise la séparation des préoccupations (une description par niveau d'abstraction isolant, en particulier, le niveau tâches du rendu) et prépare en conséquence le passage à l'échelle. Cependant, la vérification est nécessaire en pratique. Pour cela, il nous faut des bases de données d'IHM réutilisables, bases dont la communauté IHM ne dispose pas. Les travaux menés en langages de description d'IHM (par exemple, UsiXML) devraient faciliter ces capitalisation et réutilisation.

– **End User Programming :** l'utilisateur pourrait souhaiter un contrôle sur le processus de composition. Dès lors, la problématique de la composition d'IHM s'ouvre sur la programmation par l'utilisateur final (End-User Programming) (Myers 2009), domaine en pleine expansion aujourd'hui. On voit là toute la richesse mais aussi toute la difficulté du sujet.

Annexes

A

A.1 Grille d'entretien

Analyse qualitative de COMPOSE
Recrutement et Grille d'entretien
Version du 21 Octobre 2008

Participants à la définition des critères de recrutement : Yoann, Gaëlle, Humbert, Patrick, Sonia, Nadine

Objectifs :
Mesurer les attitudes (cognitif, affectif, comportemental), les attentes et les besoins (primaires, fondamental et secondaire) vis-à-vis de l'**informatique ambiante** en situations d'exception et situations nominales avec un ciblage sur **« Assistance aux personnes actives** », pour se démarquer des projets européens sur les personnes âgées. Les contextes d'usage sont : domicile/travail/voiture/vacances/ …

Lexique à utiliser dans l'entretien :
- **Nouvelle technologie** / Domotique/ informatique
- **Assistant / en support à la vie quotidienne/** système d'assistance / système d'aide / compagnon de l'usager
- **difficultés / aléas / vous rendre service**/ situation d'exception/ situation d'urgence
- **quotidien/ habituel/** situation nominale /
- **habituel/imprévu /** situation d'exception

Profil des sujets :

- **pas d'informaticien**
- **28 sujets répartis de la manière suivante selon le genre et l'âge**

	Homme	**Femme**	**Total**
18-25	**3**	**3**	**6**
26-40	**4**	**4**	**8**
40-60	**4**	**4**	**8**
60-70	**3**	**3**	**6**
Total	**14**	**14**	**28**

Autres critères de recrutement : à équilibrer au moment du recrutement

Lieux d'habitation	Urbain/Périurbain/Rurbain/Rural Essayer d'avoir 3 personnes de chaque catégorie
Le nombre d'enfants de moins de 7 ans et de moins de 18 ans	Avoir une répartition avec ou sans jeunes enfants de moins de 7 ans et moins de 18 ans (au moins 3 de chaque)
Type d'habitat	Maison/Appartement : avoir une répartition (au moins 4 de chaque)

Guide d'entretien

Introduction

Tout d'abord, je tiens à vous remercier d'avoir accepté de participer à cet entretien.
Le sujet qui nous intéresse s'inscrit dans le cadre de travail de recherche en informatique. Votre participation va contribuer à l'enrichissement de deux thèses.
Nous nous intéressons à ce que vous pensez des nouvelles technologies et ce qu'elles représentent pour vous. Surtout laissez libre cours à votre imagination. Votre avis nous intéresse d'autant plus que le terme de nouvelles technologies est une notion vague et mal définie.

Mesure de la connaissance sur les nouvelles technologies et sur son usage

- Aujourd'hui, on parle beaucoup de nouvelles technologies. Qu'est ce que cela évoque pour vous ? A quoi vous pensez quand vous entendez le terme « nouvelles technologies » ?
- On dit aussi les nouvelles technologies sont partout. Qu'en pensez-vous ? A votre avis sont-elles vraiment partout ?
- Avez-vous l'habitude d'utiliser des produits issus des nouvelles technologies ?
  - Si oui, poser les questions suivantes
    - Lesquels ?
    - Pour quels types usages (fréquence et habitude)

Mesure de la perception et opinions sur les nouvelles technologies (phase 1)

- Quel est l'objet qui représente les plus les nouvelles technologies d'aujourd'hui ?
- Est-ce que vous l'utilisez ? Pourquoi ?
- Quels sont les avantages de cet outil ?
- Quels sont les inconvénients de ces outils ?
- Quels sont les autres objets qui pour vous représentent les nouvelles technologies ?

POUR L'ENQUETEUR : Noter la liste des outils

- Quels sont les avantages de ces outils ?
- Quels sont les inconvénients de ces outils ?

Mesure de la connaissance sur les services en ligne et sur leur usage

- Avez-vous l'habitude d'utiliser un ordinateur ? pour quels usages ?

Situation N°1 Si la personne a l'habitude d'utiliser un ordinateur à domicile ou au travail

- Est-ce que vous avez l'habitude d'utiliser internet ?
- Est-ce que par exemple vous l'utilisez pour préparer des vacances, pour organiser un déplacement …
- Quels sont les avantages de ces services en ligne ?
- Quels sont les défauts de ces services en ligne ?

Situation N°2 : La personne n'a pas l'habitude d'utiliser un ordinateur à domicile ou au travail

- **Si j'ai bien noté ce que vous m'avez dit, vous** utilisez rarement un ordinateur. Est-ce que malgré cela vous utilisez internet ?
- Est-ce que par exemple vous l'utilisez pour préparer des vacances, pour organiser un déplacement …
- Est-ce qu'une autre personne le fait pour vous ?
- Mais finalement, à votre avis quels peuvent être les avantages de ces services en ligne ?
- Quels sont les défauts de ces services en ligne ?
- Et à votre avis pourquoi internet a-t-il autant de succès ?

Mesure de l'utilité des nouvelles technologies (en situation d'exception)

Situation N° 1 : La personne connaît ces outils sous la forme informatique ambiante

- Parmi les objets que vous m'avez cités, lequel préférez-vous ? pourquoi ?
- Lequel est pour vous le plus utile ? pourquoi ?
- A quelle fréquence utilisez-vous ces outils ? (mesure du niveau d'utilisation)

POUR L'ENQUETEUR : fait ici le résumé de ce qui lui a été dit afin de rompre le déroulement de l'entretien.

- Est-ce que vous pourriez me raconter une situation dans laquelle les nouvelles technologies vous ont été utiles ? pourquoi ?
- Est-ce que vous en avez vécu d'autres ?
- Est-ce que vous pourriez me raconter une situation dans laquelle les nouvelles technologies ont été insuffisantes ?
- Quels outils auraient pu vous aider ?

Situation N°2 : La personne ne connaît pas vraiment ces outils

POUR L'ENQUETEUR : Montrer des photos de nouveaux outils : vitrines tactiles, GPS, …. Et donner la définition suivante :
« Les nouvelles technologies sont des moyens récents pour communiquer, trouver et utiliser de l'information »

- Est-ce que vous possédez des appareils de ce type chez vous ?
- Lesquels ?
- Niveau d'utilisation des outils ?
- Faire émerger les difficultés. Lesquelles et pourquoi ?
- Faire raconter une histoire vécue où une assistance technique aurait pu être utile ou un des outils présentés aurait pu être utile ?

Le système COMPOSE

POUR L'ENQUETEUR : à lire avec précision

Dans le cadre de nos travaux de recherche, nous développons un assistant personnel appelé Aladin. (montrer un PDA ou un outil de ce type, ainsi l'interviewer voit immédiatement quels types d'informations il peut obtenir ?) Cet assistant a deux modes de fonctionnement :

Nous allons commencer par le premier mode :

Aladin peut connaître vos habitudes et vous proposer de manière automatique les outils que vous avez l'habitude d'utiliser. Il se charge des tâches répétitives que vous lui demandez. Par exemple, votre agenda le matin et la mise à jour de votre réveil le soir…il sait aussi que le mercredi en début d'après midi vous téléphonez chez vous. Donc, le mercredi de manière automatique en début d'après midi il vous prévient de téléphoner chez vous et il compose même le numéro, à vous de décrocher …. …. Il peut aussi prendre l'initiative et trouver de nouveaux services qui peuvent vous être utiles, comme par exemple vous transmettre un rappel de votre agenda si vous n'êtes pas devant votre ordinateur.

- Que pensez-vous de ce type d'assistant ?
- Pourquoi ?
- Est-ce que vous imagineriez d'autres situations où il serait intéressant ?
- Quels sont les avantages ?
- Quels sont les inconvénients ?

Maintenant, le deuxième mode de fonctionnement d'Aladin :

Il peut aussi en cas de difficultés ou de problèmes vous proposer un ensemble de services qui vous apporteront une solution pratique. Vous lui posez une question et il vous fournit la solution la plus pertinente. Par exemple, votre fille vient de déchirer son costume de fée clochette pour le carnaval. Vous devez en trouver un autre dans la soirée. L'assistant vous renvoie l'adresse et les horaires d'ouverture de ce type de magasin proches de votre domicile.

Il est 20h un soir d'hiver. Demain vous partez en voiture à Autrans. Vous voulez connaître l'état des routes. Votre assistant vous renvoie le service météo, le service bison futé et vous donne l'adresse de magasins d'équipements automobiles car la neige est annoncée …

- Que pensez-vous de ce type d'assistant ?
- Pourquoi ?
- Est-ce que vous imagineriez des situations dans lesquelles un tel système pourrait être intéressant ? lui demander pour chacune d'elles comment il souhaite interagir avec COMPOSE (Oral, clavier, souris, stylet, ….)
- Situations à proposer :
    - à la maison
    - en vacances
    - en voiture pour des raisons personnelles
    - en voiture pour des raisons professionnelles
    - pour répondre à un problème administratif, fournitures de documents à l'école

A.2 Scénarios

Nous présentons dans cette section les scénarios exposés lors des groupes de discussion. Ces scénarios ont été réalisés à partir des préférences utilisateur exprimées lors des interviews réalisées.

A.2.1 Médecin

Vous êtes en vacances loin de chez vous, par exemple, à Montpellier. Il fait beau. Le soleil est au zénith mais hélas vous ne vous sentez pas bien. Vous ne connaissez pas la région et avez besoin de vous soigner.

A.2.2 Mulhouse

Vous devez vous rendre à une réunion professionnelle à Mulhouse les 5 et 6 juillet. Tous vos collègues partiront la veille en train. Malheureusement, votre fille donne son spectacle de fin d'année la veille au soir. Vous vous renseignez sur les avions et réservez un vol onéreux quittant Lyon le matin même à 6h30.

Le 5 juillet au matin, vous vous levez très tôt. Vous vous faites conduire à l'aéroport. L'avion est prévu à l'heure. Vous vous installez en salle d'embarquement, commencez à travailler jusqu'à ce qu'une annonce vous interpelle. Vous espérez avoir mal compris : le vol est annulé. Vous vous renseignez : le vol est effectivement annulé. La compagnie vous propose comme seule solution de prendre l'avion suivant, l'après-midi. Cette solution ne vous convient pas. Vous devez être impérativement à la réunion à 14H.

A.2.3 Le taxi

Vous avez une réunion à Mulhouse le 26 juillet. Malheureusement l'heure de fin est prévue à 18H, ce qui ne vous permet pas de rentrer le soir même sur Lyon. Vous réservez donc votre vol retour pour le lendemain matin à 6h30.

En arrivant à Mulhouse, vous prenez un taxi à l'aéroport et demandez au chauffeur s'il pourra vous véhiculer le surlendemain à 5H. Le chauffeur surpris par l'horaire vous demande à quelle heure est votre avion. Il s'insurge disant que c'est beaucoup trop tôt : 5h30 est largement suffisant. Vous êtes contrarié : l'aéroport est loin de la ville mais le chauffeur se dit professionnel. Vous finissez par accepter. Le lendemain, vous

descendez dans la rue à 5h20, espérant que la voiture soit là. Personne. 5h30 : toujours personne. Vous appelez la compagnie : la ligne est sur répondeur.

A.2.4 La vidéo illisible

Durant le week-end du premier mai, vous profitez du jour férié pour inviter Paul, un ami d'enfance. Il arrive à 20h, vous discutez longuement autour d'un verre. La conversation est variée et le cinéma arrive naturellement comme sujet. Paul est passionné et profite de l'occasion pour vous faire découvrir sa nouvelle trouvaille sur Internet. Il vous propose de télécharger légalement le film « Final Cut » avec Jude Law, pendant que vous continuez à discuter. Au bout de quelques minutes, le téléchargement est terminé. Vous êtes impatient de le voir. Vous lancez la vidéo en double cliquant sur le fichier « Finalcut.avi », mais seul un écran noir apparaît. Vous n'êtes pas un spécialiste de l'informatique, Paul non plus.

A.2.5 L'envoi de mail

Vous écoutez tranquillement de la musique. C'est la fin de journée, vous êtes fatigué. Le téléphone sonne. Vous décrochez. C'est votre ami Victor. Il vous appelle car il aimerait que vous lui envoyiez les photos de vacances que vous avez prises en Corse. Il souhaite réunir les photos de toutes les personnes qui étaient présentes pour faire un montage aujourd'hui et il ne lui manque plus que les vôtres. Un peu contrarié de devoir vous occuper de cela maintenant, vous allumez votre ordinateur et dites à Victor que vous lui envoyez cela tout de suite. Vous voilà seul devant votre ordinateur que vous ne maîtrisez pas. Vous commencez à écrire votre mail, et ajoutez les pièces jointes une à une en espérant que toutes fonctionnent. Au bout de la 5ème, vous commencez déjà à en avoir marre, mais il en reste encore 15 et vous ne savez pas comment faire autrement alors ... Une fois toutes vos photos ajoutées, vous êtes content. Vous cliquez pour envoyer votre email. Un message d'erreur survient : « Trop de fichiers ou taille trop importante pour envoyer votre email ».

A.2.6 Autrans

C'est la période de Noël, le 23 décembre, vous êtes en congé. Comme chaque année, vous avez décidé de profiter des vacances en famille à

Autrans. Vous venez de vous lever à 10H pour partir dans la foulée. Un peu endormi, vous ouvrez vos volets et êtes surpris de l'importance de la neige. Cette nuit a dû être fraîche et mouvementée. Le temps est encore incertain et vous commencez à douter de pouvoir arriver sain et sauf en voiture, surtout que vous n'êtes pas équipé en pneus neige.

A.2.7 La panne d'essence imminente

Il est 5h du matin, le 3 juin. Vous êtes en voiture. Vous roulez depuis des heures pour aller faire une marche en Cévennes. Vous êtes un marcheur intensif. Vous avez prévu de passer une semaine avec un groupe d'habitués. Le rendez-vous est à Alès à 8h. Vous êtes à 1h d'Ales donc en avance mais impatient d'arriver pour pouvoir vous préparer tranquillement. Un voyant attire votre attention : la jauge d'essence. Vous n'y croyez pas. Il vous reste peu de carburant. Aucune station essence n'est signalée.

A.3 Les critères ergonomiques de Scapin et Bastien

L'évaluation de la qualité ergonomique est faite par un ergonome. Le travail de l'ergonome s'articule autour d'« outils » qui lui servent à juger de la facilité d'utilisation d'un système informatique. Parmi ces outils, Scapin et Bastien (1997) ont établi une liste de 18 dimensions à satisfaire répartis en 8 critères.

1. **Guidage**

   D'un point de vue général, il s'agit de l'ensemble des moyens mis en œuvre pour conseiller, orienter, informer et conduire l'utilisateur lors de ses interactions avec l'ordinateur. Un bon guidage facile l'apprentissage et l'utilisation du système. La dimension de guidage comprend 4 sous-critères :

   1.1 **Incitation** : conseille d'inciter l'utilisateur à effectuer des actions spécifiques en lui fournissant des indices. Par exemple, guider les entrées de données en indiquant le format adéquat et les valeurs acceptables pour exprimer le besoin de l'utilisateur (figure A.1 de l'IHM de l'assistant figure 1.2) : « *(exemple :*

Je dois être à la gare dans 30 min) ».

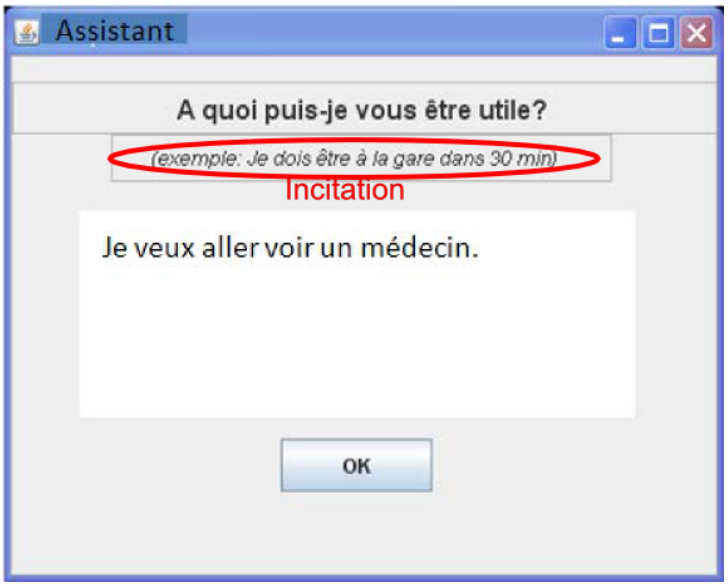


FIGURE A.1 – *Illustration du critère « Guidage » dans sa dimension « Incitation ».*

1.2 **Groupement/Distinction entre Items** : préconise de grouper des différents éléments visuels de façon cohérente et ordonnée. Ce critère comprend 2 sous-critères :

1.2.1 **Groupement/Distinction par la Localisation** : prône de positionner les items les uns par rapport aux autres afin d'indiquer leur appartenance, ou non, à une classe donnée d'objets. Par exemple, les informations pour appeler le docteur Mabuse sont groupées et séparées de celles pour se déplacer (figure A.2) de l'exemple 1 du cas d'étude figure 1.4.

1.2.2 **Groupement/Distinction par le Format** : recommande de donner aux éléments des caractéristiques graphiques particulières afin d'indiquer leur appartenance, ou non, à une classe donnée d'objets.

1.3 **Feedback immédiat** : signifie que dans tous les cas, l'ordinateur doit répondre à l'utilisateur en fonction des actions et des requêtes de ce dernier. Par exemple, dans les cas où les traitements sont longs, comme composer dynamiquement une IHM pour répondre à l'objectif de l'utilisateur, une information indiquant à l'utilisateur que les traitements sont en cours

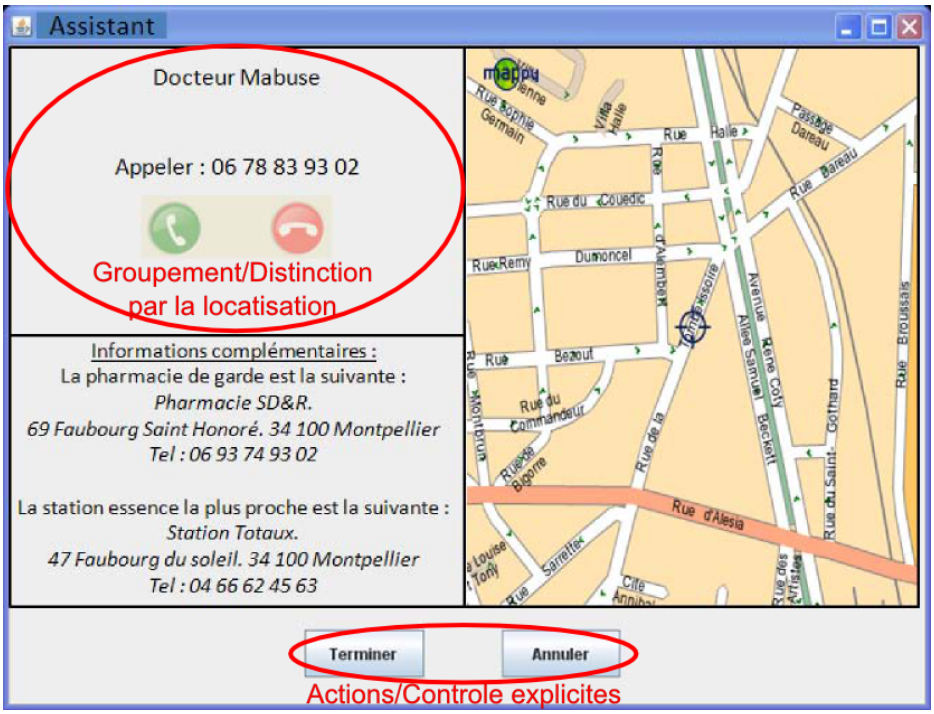


FIGURE A.2 – *Illustration de 3 dimensions de critères.*

devrait lui être fournie immédiatement.

1.4 **Lisibilité** : préconise que les caractéristiques lexicales de présentation des informations sur l'écran doivent faciliter la lecture de ces informations.

2. **Charge de travail**

Le critère Charge de travail concerne l'ensemble des éléments de l'interface qui ont un rôle dans la réduction de la charge perceptive ou mnésique des utilisateurs et dans l'augmentation de l'efficacité du dialogue. La dimension de charge de travail comprend 2 sous-critères :

2.1 **Brièveté** : conseille de limiter le travail de lecture, d'entrée et les étapes par lesquelles doivent passer les usagers. Ce critère comprend 2 sous-critères :

2.1.1 **Concision** : encourage de réduire la charge de travail au niveau perceptif et mnésique pour ce qui est des éléments individuels d'entrée ou de sortie. Par exemple, ne pas demander aux utilisateurs d'entrer des données qui

peuvent être déduites par le système comme le numéro de téléphone pré-composé du docteur Mabuse dans la figure A.3 du cas d'étude figure 1.5.

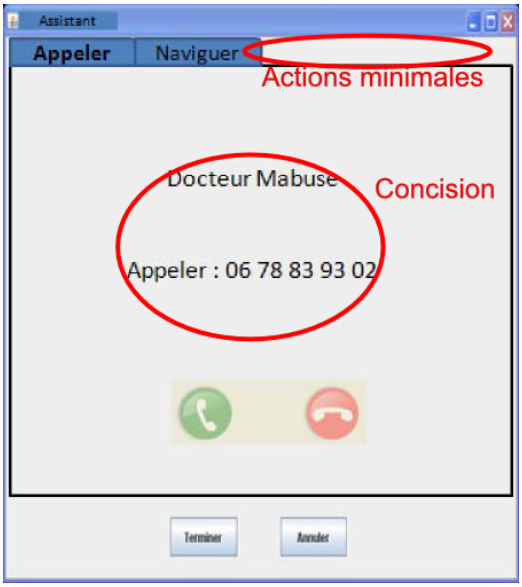


FIGURE A.3 – *Illustration de deux critères.*

2.1.2 **Actions Minimales** : prône de limiter les étapes par lesquelles doivent passer les utilisateurs. Pour illustrer, dans l'exemple 2 du cas d'étude, l'action de consulter les informations complémentaires sur la pharmacie et la station essence n'est pas disponible dans la figure A.3 pour privilégier ce critère.

2.2 **Densité Informationnelle** : préconise de réduire la charge de travail du point de vue perceptif et mnésique, pour des ensembles d'éléments et non pour des items. Par exemple, limiter la densité informationnelle de l'écran, en affichant seulement les informations nécessaires.

3. **Contrôle explicite**

Ce critère concerne la prise en compte par le système des actions explicites des utilisateurs et le contrôle qu'ont les utilisateurs sur le traitement de leurs actions. Cette dimension comprend 2 sous-critères :

3.1 **Actions Explicites** : recommande d'expliciter la relation entre le fonctionnement de l'application et les actions des utilisateurs. Par exemple, l'entrée de commandes doit se terminer par une indication de fin (le bouton « Terminer » de la figure A.2).

3.2 **Contrôle Utilisateur** : conseille que l'utilisateur doit pouvoir contrôler le déroulement des traitements informatiques en cours. Par exemple, autoriser l'utilisateur à revenir en arrière dans toute opération en cours (bouton « Annuler »).

4. **Adaptabilité**

Ce critère concerne la capacité à réagir selon le contexte et selon les besoins et les préférences des utilisateurs. La dimension d'adaptabilité comprend 2 sous-critères :

4.1 **Flexibilité** : prône de mettre à la disposition des utilisateurs des moyens pour personnaliser l'interface afin de rendre compte de leurs stratégies ou habitudes de travail et des exigences de la tâche. Le critère correspond aussi au nombre de façons différentes mises à disposition des utilisateur pour atteindre un objectif donné. Par exemple, pour privilégier ce critère, Victor aurait pu avoir le choix entre la carte et le GPS.

4.2 **Prise en Compte de l'Expérience de l'Utilisateur** : le système doit respecter le niveau d'expérience de l'utilisateur. Par exemple, prévoir des choix d'entrées pas-à-pas ou multiples selon l'expérience des utilisateurs. Par exemple, un utilisateur novice et expert peuvent disposer de fonctionnalités différentes.

5. **Gestion des erreurs**

Ce critère concerne les moyens permettant d'une part d'éviter ou de réduire les erreurs, d'autre part de les corriger lorsqu'elles surviennent.

5.1 **Protection contre les erreurs** : conseille de mettre en place des moyens pour détecter et prévenir les erreurs. Par exemple,

toutes les actions possibles sur une interface doivent être envisagées et plus particulièrement les appuis accidentels des touches du clavier afin que les entrées non-attendues soient détectées. Le choix d'une case à sélectionner limite les erreurs comparé à un champ texte permettant à l'utilisateur d'écrire la ville qu'il souhaite (figure A.4).

5.2 **Qualité des Messages d'Erreurs** : a pour objectif de s'assurer que l'information donnée aux utilisateurs sur la nature des erreurs commises (syntaxe, format, etc.) et sur les actions à entreprendre pour les corriger, soit pertinente, facile à lire et exacte. Par exemple, utiliser un vocabulaire neutre, non personnalisé, non réprobateur dans les messages d'erreurs ; éviter l'humour.

5.3 **Correction des Erreurs** : indique la nécessité de mettre à la disposition des utilisateurs des moyens pour corriger leurs erreurs.

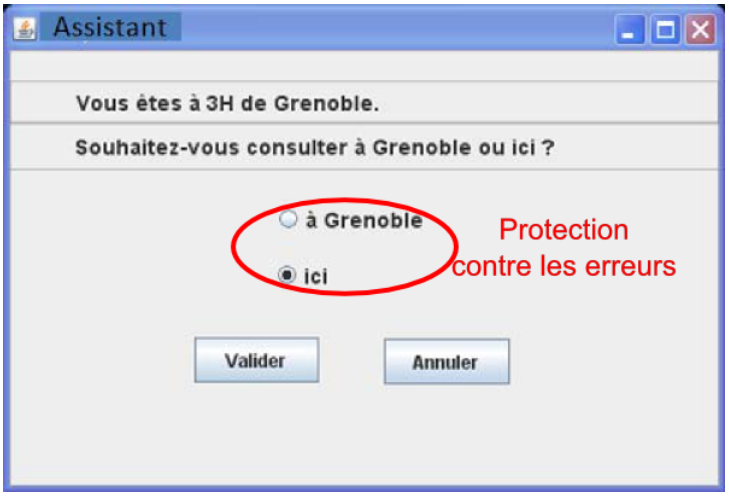


FIGURE A.4 – *Illustration du critère « Gestion des erreurs » dans sa dimension « Protection des erreurs ».*

6. **Homogénéité / Cohérence**

Les choix de conception d'interface doivent être conservés pour des contextes identiques, et doivent être différents pour des contextes différents.

7. **Signifiance des codes et Dénomination**

   Ce critère concerne l'adéquation entre l'objet et l'information affichée ou entrée, et son référent. Des codes et dénominations « signifiant » disposent d'une relation sémantique forte avec leur référent. Par exemple, rendre les règles d'arbitraire d'abréviation explicites.

8. **Compatibilité**

   Il faut qu'il y ait accord entre les caractéristiques des utilisateurs et des tâches d'une part, et l'organisation des sorties, des entrées et du dialogue d'une application donnée d'autre part.

Bibliographie

CORBA : Architecture and Specification. Object Management Group, Inc., 1995. (Cité page 58.)

J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, 1983. (Cité pages 36 et 44.)

A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, et al. Daml-s : Semantic markup for web services. Dans *The emerging semantic web : selected papers from the first Semantic Web Working Symposium*, volume 75, page 131. Ios Pr Inc, 2002. (Cité page 59.)

L. Balme, A. Demeure, G. Calvary, et J. Coutaz. Sedan-bouillon : A plastic web site. Dans *INTERACT 2005 Workshop on Plastic Services for Mobile Devices*, pages 1–3, 2005. (Cité page 72.)

C. Bessière, J. Euzenat, R. Jeansoulin, G. Ligozat, et S. Schwer. Raisonnement spatial et temporel. *Actes 6e journées nationales du PRC-GDR intelligence artificielle*, 1997. (Cité page 35.)

A. Blum et M. Furst. Fast Planning Through Planning Graph Analysis. Dans *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1636–1642, 1995. (Cité page 88.)

E. Bruneton, T. Coupaye, et J.B. Stefani. The fractal composition framework specification. final release, version 1.0. *ObjectWeb Consortium*. (Cité page 57.)

A. Bruseberg et D. McDonagh-Philp. Focus groups to support the industrial/product designer : a review based on current literature and designers' feedback. *Applied Ergonomics*, 33(1) :27–38, 2002. (Cité page 18.)

A. Bucchiarone et S. Gnesi. A survey on services composition languages and models. Dans *in Proceedings of International Workshop on Web Services Modeling and Testing (WS-MaTe2006)*, pages 51–63, 2006. (Cité pages ix et 58.)

J. Bézivin. On the unification power of models. *Software and Systems Modeling*, 4(2) :171–188, 2005. (Cité page 40.)

G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, et J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3) :289–308, 2003. (Cité pages 1 et 28.)

S. Card, T. Moran, et A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, U.S.A., 1983. (Cité page 1.)

F. Casati, S. Ilnicki, L.J. Jin, V. Krishnamoorthy, et M.C. Shan. eflow : a platform for developing and managing composite e-services. Dans *Research Challenges, 2000. Proceedings. Academia/Industry Working Conference on*, pages 341–348. IEEE, 2000. (Cité page 63.)

R. Chinnici, J.J. Moreau, A. Ryman, et S. Weerawarana. Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C Working Draft*, 26, 2004. (Cité page 60.)

T. Clerckx, K. Luyten, et K. Coninx. Generating context-sensitive multiple device interfaces from design. *Computer-Aided Design of User Interfaces IV*, pages 283–296, 2005. (Cité page 63.)

Corba components. Joint revised submission. *Object Management Group*, 1999. (Cité page 57.)

J. Coutaz. Meta-user interfaces for ambient spaces. Dans Karin Coninx, Kris Luyten, et Kevin A. Schneider, éditeurs, *TAMODIA*, volume 4385 de *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006. ISBN 978-3-540-70815-5. (Cité page 9.)

J. Coutaz, L. Nigay, D. Salber, et A. Blandford. Four easy pieces for assessing the usability of multimodal interaction : the care properties. Dans *Proceedings of IFIP INTERACT'95 : Human-Computer Interaction*, pages 115–120, 1995. (Cité page 34.)

A. Coyette et J. Vanderdonckt. A sketching tool for designing anyuser, anyplatform, anywhere user interfaces. *Human-Computer Interaction-INTERACT 2005*, pages 550–564, 2005. (Cité page 19.)

F. Curbera, Y. Goland, J. Klein, F. Leymann, S. Weerawarana, et al. Business process execution language for web services, version 1.1. 2003. (Cité page 59.)

O. Daassi. *Les COMETts : une nouvelle génération d'Interacteurs pour la Plasticité des Interfaces Homme-Machine.* PhD thesis, 2007. Thèse de doctorat Informatique préparée au Laboratoire d'Informatique de Grenoble (LIG), Université Joseph Fourier. 134 pages. (Cité pages viii, 26 et 27.)

L.G. De Michiel, L.U. Yalçinalp, et S. Krishnan. Enterprise javabeans tm specification, version 2.0. *On-line at< http ://java. sun. com/products/ejb/docs. html>, year2001*. (Cité page 57.)

W. P. De Roever, F. S. De Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, et J. Zwiers. *Concurrency Verification : Introduction to Compositional and Noncompositional Methods*, volume 54 de *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001. ISBN 0-521-80608-9. (Cité page 129.)

A. Demeure, G. Calvary, et K. Coninx. COMET(s), A software architecture style and an interactors toolkit for plastic user interfaces. Dans T. C. Nicholas Graham et Philippe A. Palanque, éditeurs, *DSV-IS*, volume 5136 de *Lecture Notes in Computer Science*, pages 225–237. Springer, 2008. ISBN 978-3-540-70568-0. (Cité pages 34 et 116.)

M. Devy, B. Espiau, F. Genot, M. Ghallab, F. Lamiraux, JP Laumond, P. Morin, P. Rives, C. Samson, et S. Sekhavat. *La robotique mobile, chapitre Planification et décision*. J.P. Laumond (ed.), Hermes, 2001. (Cité pages 80 et 81.)

A. Dey, T. Sohn, S. Streng, et J. Kodama. icap : Interactive prototyping of context-aware applications. *Pervasive Computing*, pages 254–271, 2006. (Cité page 16.)

S. Edelkamp et J. Hoffmann. Pddl 2.2 : the language for the classical part of ipc-04. *ICAPS-2004-International Planning Competition*, pages 2–6, 2004. (Cité page 86.)

M. J. Egenhofer et R. D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5 :161–174, 1991. (Cité pages 37 et 47.)

R. Ennals et D. Gay. User-friendly functional programming for web mashups. Dans *Proceedings of the 12th ACM SIGPLAN international conference on Functional programming*, pages 223–234. ACM, 2007. (Cité page 63.)

K. Erol. *Hierarchical task network planning :–formalization, analysis, and implementation*. PhD thesis, research directed by Dept. of Computer Science, 1995. (Cité pages 11, 88 et 93.)

J-M. Favre. Foundations of model (driven) (reverse) engineering : Models - episode I : Stories of the fidus papyrus and of the solarus. Dans Jean Bézivin et Reiko Heckel, éditeurs, *Language Engineering for Model-Driven Software Development*, volume 04101 de *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2004. (Cité page 40.)

M. Feldmann, F. Martens, G. Berndt, J. Spillner, et A. Schill. Rapid development of service-based interactive application using service ANNOTATIONS. pages 319–322, 2010. (Cité pages ix, 72 et 74.)

R.E. Fikes et N.J. Nilsson. STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4) : 189–208, 1971. ISSN 0004-3702. (Cité page 86.)

M. Fox et D. Long. PDDL2. 1 : An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20 (1) :61–124, 2003. (Cité page 86.)

J. Fujima, A. Lunzer, K. Hornbæk, et Y. Tanaka. Clip, connect, clone : combining application elements to build custom interfaces for information access. Dans Steven Feiner et James A. Landay, éditeurs, *UIST*, pages 175–184. ACM, 2004. ISBN 1-58113-957-8. (Cité page 70.)

Y. Gabillon. Planification pour la composition dynamique d'interfaces homme-machine. Dans *Proc. of Rencontres Jeunes Chercheurs en IHM RJC-IHM 2008*, Cap d'Agde, France, 13 - 15 mai 2008. (Cité page 1.)

Y. Gabillon, G. Calvary, et H. Fiorino. Composition dynamique de systèmes interactifs : l'alliance de l'ihm et de la planification. Dans *Journées Francophones sur de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA'08)*, pages 155–163, Metz, France, 19 - 20 juin 2008. (Cité pages 79 et 113.)

Y. Gabillon, G. Calvary, et H. Fiorino. Composition d'interfaces homme-machine en contexte : approche par planification automatique. *Technique et science informatiques*, 30(10) :25, 2011a. (Cité pages 39, 55, 79 et 113.)

Y. Gabillon, G. Calvary, N. Mandran, et H. Fiorino. Composition dynamique d'interfaces homme-machine : besoin utilisateur ou défi de chercheur ? Dans *IHM '09 : Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine*, pages 61–64, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-461-4. (Cité pages 1, 15 et 116.)

Y. Gabillon, M. Petit, G. Calvary, et H. Fiorino. Automated planning for user interface composition. Dans *2nd International Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS'11) of the 2011 International Conference on Intelligent User Interfaces (IUI'11)*, pages 1–5. Springer HCI, 2011b. (Cité pages 79 et 113.)

K. Gajos et D.S. Weld. SUPPLE : automatically generating user interfaces. Dans *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 93–100. ACM, 2004. ISBN 1581138156. (Cité page 70.)

K.Z. Gajos, J.O. Wobbrock, et D.S. Weld. Automatically generating user interfaces adapted to users' motor and vision capabilities. Dans *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 231–240. ACM, 2007. (Cité page 70.)

A. Gerevini et D. Long. Plan constraints and preferences in PDDL3. *ICAPS 2006*, page 7, 2006. (Cité page 86.)

M. Ghallab, D. Nau, et P. Traverso. *Automated Planning : Theory and Practice*. Morgan Kaufmann Publishers Inc., Amsterdam, 2004. ISBN 978-1-55860-856-6. (Cité pages 80 et 93.)

C. Gram et G. Cockton, éditeurs. *Design Principles for Interactive Software*. Chapman & Hall, 1996. ISBN 0-41272470-7. (Cité page 26.)

D. Grolaux, J. Vanderdonckt, et P. V. Roy. Attach me, detach me, assemble me like you work. Dans Maria Francesca Costabile et Fabio Paternò, éditeurs, *INTERACT*, volume 3585 de *Lecture Notes in Computer Science*, pages 198–212. Springer, 2005. ISBN 3-540-28943-7. (Cité pages ix, 72 et 73.)

P. R. Halmos. *Naive Set Theory*. Van Nostrand, Princeton, New Jersey, 1960. (Cité page 43.)

T. Hanh. *Coordination adaptative de services à base de contrats*. PhD thesis, Institut polytechnique de Grenoble, 2009. (Cité pages 60 et 62.)

R.R. Hassen, L. Nourine, et F. Toumani. Protocol-based web service composition. Dans *Service oriented computing : 6th international conference ; proceedings*, volume 5364, page 38. Springer, 2008. (Cité page 60.)

D. Hindus, S.D. Mainwaring, N. Leduc, A.E. Hagstr
"om, et O. Bayley. Casablanca : designing social communication devices for the home. Dans *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 325–332. ACM, 2001. (Cité page 16.)

O. Ilghami et D. Nau. A general approach to synthesize problem-specific planners. Juin 30 2003. (Cité pages 61 et 88.)

ISO/IEC CD 25000.2. *Software and Systems Engineering, Software product quality requirements and evaluation (SquaRE)), Guide to SquaRE, 13 january 2003*. (Cité page 26.)

C. Joffroy. *Composition d'applications et de leurs Interfaces homme-machine dirigée par la composition fonctionnelle*. PhD thesis, Université de Nice Sophia-Antipolis, Juin 06 2011. (Cité pages 67 et 69.)

P. Johnson, S. Wilson, P. Markopoulos, et J. Pycock. ADEPT – advanced environment for prototyping with task models. Dans *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems*, Demonstrations, page 56, 1993. (Cité page 69.)

N. Kavantzas. Web services choreography description language (ws-cdf) version 1.0. *http ://www. w3. org/TR/ws-cdl-10/*, 2004. (Cité page 59.)

B. Kitchenham et S.L. Pfleeger. Software quality : The elusive target. *IEEE Software*, 13(1) :12–21, Janvier 1996. (Cité page 26.)

M. Klusch, A. Gerber, et M. Schmidt. Semantic web service composition planning with owls-xplan. Dans *Proceedings of the AAAI Fall Symposium on Semantic Web and Agents, Arlington VA, USA, AAAI Press*, 2005. (Cité page 61.)

S. Krakowiak. Architecture des systèmes : avancées et défis. `http://www.liglab.fr/spip.php?article890`, 02 décembre 2010. (Cité page 129.)

S. Lepreux, A. Hariri, J. Rouillard, D. Tabary, J.-C. Tarby, et C. Kolski. Towards multimodal user interfaces composition based on usiXML and MBD principles. Dans Julie A. Jacko, éditeur, *HCI (3)*, volume 4552 de *Lecture Notes in Computer Science*, pages 134–143. Springer, 2007. ISBN 978-3-540-73108-5. (Cité pages viii, 29 et 64.)

S. Lepreux et J. Vanderdonckt. Towards A support of user interface design by composition rules. Dans Gaëlle Calvary, Costin Pribeanu, Giuseppe Santucci, et Jean Vanderdonckt, éditeurs, *CADUI*, pages 231–244. Springer, 2006. ISBN 978-1-4020-5819-6. (Cité pages ix, 64 et 66.)

A. Lewandowski, S. Lepreux, et G. Bourguin. Tasks models merging for high-level component composition. Dans Julie A. Jacko, éditeur, *HCI (1)*, volume 4550 de *Lecture Notes in Computer Science*, pages 1129–1138. Springer, 2007. ISBN 978-3-540-73104-7. (Cité pages ix, 67, 68 et 75.)

J. Lin, J. Wong, J. Nichols, A. Cypher, et T.A. Lau. End-user programming of mashups with vegemite. Dans Cristina Conati, Mathias Bauer, Nuria Oliver, et Daniel S. Weld, éditeurs, *IUI*, pages 97–106. ACM, 2009. ISBN 978-1-60558-168-2. (Cité pages 63 et 70.)

C. Marin. *Une approche orientée domaine pour la composition de services*. PhD thesis, Université Grenoble 1 Joseph Fourier, 2008. (Cité page 56.)

D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, et al. Owl-s : Semantic markup for web services. *W3C Member Submission*, 22 :2007–04, 2004. (Cité page 60.)

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, et D. Wilkins. PDDL-the planning domain definition language. *The AIPS-98 Planning Competition Comitee*, 1998. (Cité pages 61 et 86.)

S. McIlraith et T.C. Son. Adapting golog for composition of semantic web services. Dans *PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-*, pages 482–496. Citeseer, 2002. (Cité page 61.)

B. Medjahed. *Semantic web enabled composition of web services*. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University, 2004. (Cité page 61.)

G. Mori, F. Paternò, et C. Santoro. CTTE : Support for developing and analysing task models for interactive system design. 28 :797–813, 2002. (Cité pages 29, 63 et 69.)

B. A. Myers. Engineering more natural interactive programming systems : keynote talk. Dans T. C. Nicholas Graham, Gaëlle Calvary, et Philip D. Gray, éditeurs, *EICS*, pages 1–2. ACM, 2009. ISBN 978-1-60558-600-7. (Cité pages 63, 69 et 130.)

D. Nau, Y. Cao, A. Lotem, et H. Muñoz-Avila. SHOP : Simple hierarchical ordered planner. Dans Thomas Dean, éditeur, *IJCAI*, pages 968–975. Morgan Kaufmann, 1999. ISBN 1-55860-613-0. (Cité page 88.)

J. Nichols, Z. Hua, et J. Barton. Highlight : a system for creating and deploying mobile web applications. Dans Steve B. Cousins et Michel Beaudouin-Lafon, éditeurs, *UIST*, pages 249–258. ACM, 2008. ISBN 978-1-59593-975-3. (Cité page 72.)

J. Nichols, B. Myers, et B. Rothrock. UNIFORM : automatically generating consistent remote control user interfaces. Dans *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems*, volume 1 de *Automatic generation and usability*, pages 611–620, 2006. (Cité page 70.)

J. Nielsen. Usability engineering. Chapitre 2 : What is Usability, pages 23–48. Academic Press, Cambridge, MA, 1993. (Cité page 26.)

L. Nigay et J. Coutaz. A generic platform for addressing the multimodal challenge. Dans *CHI*, pages 98–105, 1995. (Cité page 34.)

V. Normand. *Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation*. PhD thesis, HAL - CCSD, Novembre 24 1992. (Cité page 31.)

M. P. Papazoglou. Service-oriented computing : Concepts, characteristics and directions. Dans *WISE*, pages 3–12. IEEE Computer Society, 2003. ISBN 0-7695-1999-7. (Cité pages 56, 57 et 58.)

F. Paternò, O. Mancini, et S. Meniconi. Concurtasktrees : a diagrammatic notation for specifying task models. Dans *Proceedings of IFIP INTERACT'97 : Human-Computer Interaction*, pages 362–369, 1997. (Cité page 44.)

F. Paternò, C. Santoro, et A. Scorcia. Preserving rich user interface state in web applications across various platforms. Dans Peter Forbrig et Fabio Paternò, éditeurs, *TAMODIA/HCSE*, volume 5247 de *Lecture Notes in Computer Science*, pages 255–262. Springer, 2008. ISBN 978-3-540-85991-8. (Cité page 72.)

F. Paternò, C. Santoro, et L. D. Spano. MARIA : A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Transactions Computer-Human Interact*, 16(4), 2009. (Cité page 52.)

A-M. Pinna-Dery, J. Fierstone, et E. Picard. Component model and programming : A first step to manage human computer interaction adaptation. Dans Luca Chittaro, éditeur, *Mobile HCI*, volume 2795 de *Lecture Notes in Computer Science*, pages 456–465. Springer, 2003. ISBN 3-540-40821-5. (Cité pages ix, 65 et 66.)

J. Rao, P. Küngas, et M. Matskin. Application of linear logic to web service composition. Dans Liang-Jie Zhang, éditeur, *IWCS*, page 3. CSREA Press, 2003. ISBN 1-892512-49-1. (Cité page 62.)

J. Rao et X. Su. A survey of automated web service composition methods. *Semantic Web Services and Web Process Composition*, pages 43–54, 2005. (Cité pages 61 et 63.)

D.L. Scapin et J.M.C. Bastien. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour and Information Technology*, 16(4) :220–231, 1997. ISSN 0144-929X. (Cité pages 26, 27, 31, 115 et 140.)

B. Shackel. Usability - context, framework, definition, design and evaluation. *Interacting with Computers*, 21(5-6) :339–346, 2009. (Cité page 26.)

J.-S. Sottet. *Méga-IHM : malléabilité des Interfaces Homme-Machine dirigées par les modèles*. PhD thesis, 2008. Thèse de doctorat Informatique préparée au Laboratoire d'Informatique de Grenoble (LIG), Université Joseph Fourier. (Cité page 40.)

J-S. Sottet, V. Ganneau, G. Calvary, J. Coutaz, A. Demeure, J-M. Favre, et R. Demumieux. Model-driven adaptation for plastic user interfaces. Dans Maria Cecília Calani Baranauskas, Philippe A. Palanque, Julio Abascal, et Simone Diniz Junqueira Barbosa, éditeurs, *INTERACT (1)*, volume 4662 de *Lecture Notes in Computer Science*, pages 397–410. Springer, 2007. ISBN 978-3-540-74794-9. (Cité pages viii et 33.)

W. Stürzlinger, O. Chapuis, D. Phillips, et N. Roussel. User interface facades : towards fully adaptable user interfaces. Dans Pierre Wellner et Ken Hinckley, éditeurs, *UIST*, pages 309–318. ACM, 2006. ISBN 1-59593-313-1. (Cité pages 63 et 72.)

P. A. Szekely. Retrospective and challenges for model-based interface development. Dans François Bodart et Jean Vanderdonckt, éditeurs, *DSV-IS*, pages 1–27. Springer, 1996. ISBN 3-211-82900-8. (Cité page 28.)

C. Szyperski, D. Gruntz, et S. Murer. *Component software : beyond object-oriented programming*. Addison-Wesley Professional, 2002. (Cité page 56.)

D.S. Tan, B. Meyers, et M. Czerwinski. Wincuts : manipulating arbitrary window regions for more effective use of screen space. Dans *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems*, volume 2 de *Late breaking result papers*, pages 1525–1528, 2004. (Cité pages 63 et 72.)

D.A. Taylor. *Object technology : a manager's guide*. Addison-Wesley, 1998. (Cité page 56.)

The OSGi Alliance. OSGi service platform — core specification, Août 2005. Release 4. (Cité page 58.)

D. Thevenin. *L'adaptation en Interction Homme-Machine : le cas de la plasticité*. PhD thesis, 2001. Thèse de doctorat Informatique préparée au Laboratoire de Communication Langagière et Interaction Personne-Système (IMAG), Université Joseph Fourier, 238 pages. (Cité page 2.)

D. Thevenin et J. Coutaz. Plasticity of user interfaces : Framework and research agenda. Dans *Proceedings of IFIP INTERACT'99 : Human-Computer Interaction*, Mobile Systems, pages 110–117, 1999. (Cité pages 2 et 32.)

W.M.P. Van der Aalst, T. Basten, H.M.W Verbeek, P.A.C. Verkoulen, et M. Voorhoeve. Adaptive workflow-on the interplay between flexibility and support. pages 353–360, 1999. (Cité page 63.)

R. Waldinger. Web agents cooperating deductively. *Formal Approaches to Agent-Based Systems*, pages 250–262, 2001. (Cité page 62.)

M. Weiser. The computer for the 21st century. *Scientific American*, 265(3) : 94–104, Septembre 1991. (Cité page 1.)

M. Weiser. Ubiquitous computing. *IEEE Computer Hot Topics*, 26(10) : 71–72, Octobre 1993. (Cité page 1.)

D. Wu, E. Sirin, J. Hendler, D. Nau, et B. Parsia. Automatic web services composition using shop2. Dans *Workshop on Planning for Web Services*. Citeseer, 2003. (Cité page 61.)

V. Wulf, V. Pipek, et M. Won. Component-based tailorability : Enabling highly flexible software applications. *Int. J. Hum.-Comput. Stud*, 66(1) : 1–22, 2008. (Cité page 72.)

Titre : Composition d'Interfaces Homme-Machine par planification automatique

Résumé : En informatique ambiante, les objectifs de l'utilisateur peuvent émerger opportunément. Il devient, dès lors, nécessaire de générer à la volée des systèmes interactifs. Un système interactif est composé d'un noyau fonctionnel et d'une Interface Homme-Machine (IHM). Cette thèse traite de la composition d'IHM pour un objectif utilisateur et un contexte d'usage (utilisateur, plate-forme, environnement) donnés. Elle en propose un espace problème fondé sur les exigences utilisateur recueillies par une étude qualitative. Un état de l'art positionne notre travail et en montre la complémentarité par rapport aux travaux existants : la composition du modèle de tâches. La composition de l'IHM concrète est déléguée à une boîte à outils d'interacteurs définis au niveau tâches. La composition du modèle de tâches se fait par planification automatique. L'étude montre que les planificateurs existants répondent partiellement au problème. Aussi, un planificateur a été spécifiquement développé pour l'IHM. Son utilisation est illustrée dans un prototype Compose. Le travail est original à deux titres : d'une part, son approche « Composition de modèles de tâches » est une extension de la littérature ; d'autre part, la composition d'IHM est un nouveau cadre applicatif pour les algorithmes de planification.

Mots-clés : Interface Homme-Machine, Planification automatique, Composition, Contexte d'usage, Critères d'ergonomies

Title : Composition of user interfaces by automated planning

Abstract : In ubiquitous computing, user needs may opportunistically emerge along the variation of the context of use. Thus, there is a need for dynamically composing interactive systems. An interactive system is made of a functional core and a User Interface (UI). This work deals with the composition of UIs to support opportunistic user needs in a given context of use (user, platform, environment). It proposes a problem space of UI composition based on a social study. A state of the art shows the originality of the work : the composition of the task model. The composition of the concrete UI is delegated to a toolkit of interactors defined at the task level. The composition of the task model is done by automated planning. The work shows that current planners do not fulfill Human Computer Interaction (HCI) requirements. Therefore, a specific planner has been developed to compose UIs. This planner is used in Compose, our proof of concept. The work is original in two points : first, by the

high level of abstraction the composition is performed at ; secondly, by the use of automated planning in HCI.

Keywords : User Interface, Automated planning, Composition, Context of use, Ergonomic criteria