FACILITER LA MISE ENPLACE D'ÉTUDES D'UTILISABILITÉ PAR DES OUTILS DE STOCKAGE DES DONNÉES ET D'ANALYSE AUTOMATIQUE DES TRACES D'UTILISATION : UN CAS D'ÉTUDE AVEC UNE APPLICATION MOBILE.

par

Perrine Cribier-Delande

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 1 septembre 2016

Le 1 septembre 2016

le jury a accepté le mémoire de Madame Perrine Cribier-Delande dans sa version finale.

Membres du jury

Professeure Hélène Pigot
Directrice de recherche
Département d'informatique

Professeur Hugo Larochelle
Membre interne
Département d'informatique

Professeur Sylvain Giroux
Président-rapporteur
Département d'informatique

Sommaire

Le laboratoire DOMUS développe des applications pour assister les personnes en perte d'autonomie et les personnes avec des troubles cognitifs. Chaque application est ou a déjà été le sujet de plusieurs études d'utilisabilité qui permettent de les améliorer.

Ces études prennent beaucoup de temps à mettre en place, car l'on rencontre souvent des problèmes de logistique (format et sensibilité des données, chercheurs répartis sur une grande aire géographique). C'est pourquoi un outil appelé GEDOPAL a été développé. Il permet de partager entre chercheurs de différents centres les données créées et utilisées lors de la mise en place des études d'utilisabilité. La conception et la réalisation de cet outil ont nécessité une réflexion en amont sur la nature et la sensibilité de ces données. Cette réflexion est l'objet du Chapitre 3.

Ces études prennent aussi beaucoup de temps lors de l'analyse des résultats. De plus, certaines données créées lors de ces études, telles que les traces d'utilisation ont des volumétries trop importantes pour être analysées manuellement. C'est pourquoi nous avons créé un processus permettant d'analyser ces traces d'utilisation pour y détecter les erreurs utilisateurs dans l'espoir de les relier à des problèmes d'utilisabilité. Ce processus se compose de deux parties : la première est une analyse formelle de l'application, qui sera présentée au Chapitre 4, et la seconde l'application d'un outil d'apprentissage automatique aux traces d'utilisation pour y détecter les erreurs utilisateurs. Cet outil est présenté au Chapitre 5.

Mots-clés: utilisabilité, apprentissage automatique, étude d'utilisabilité, application mobile, alzheimer

Remerciements

Ce travail a été effectué au laboratoire DOMUS de l'Université de Sherbrooke. Mes premiers remerciements vont à ma directrice de recherche, la professeure Hélène Pigot : ses conseils, son appui, son soutien et la confiance qu'elle a eue en moi ont été une aide indispensable ; sans sa présence toujours rassurante et motivante, je n'aurais pas pu finir cette maîtrise. Je remercie aussi Sylvain Giroux pour son indéfectible soutien et la qualité exeptionnelle de son enseignement qui m'a ouvert de nouveaux horizons. Ces deux directeurs m'ont permis de vivre 18 mois au laboratoire DOMUS où la bonne humeur est toujours de mise, sans nuire à une activité studieuse.
Le professeur Hugo Larochelle m'a aidée lors du développement de mes réseaux de neurones, qu'il en soit remercié.
Je remercie aussi la docteure Hélène Imbeault pour son aide précieuse dans la compréhension des problématiques liées à la maladie d'Alzheimer. Le travail de Brigitte Gilbert, Nathalie Bier et de leurs collaborateurs de l'Institut Universitaire de Gériatrie de Montréal nous a permis d'étendre nos tests de AP@LZ au-delà de Sherbrooke, je les en remercie tous très sincèrement.
Mes remerciements vont aussi à l'ensemble des membres du laboratoire DOMUS, à commencer par Marc Chevalaz pour son soutien technique indispensable, Amandine Porcher, Fanny Le Morellec, Yannick Adelise et Marisnel Olivares pour leur soutien moral et leurs conseils tout aussi indispensables, Maud Loarer et Paul Cornec pour leur aide, sans eux GEDOPAL n'aurait pu voir le jour, Wathek Bellah Loued, Taoufik Zayani, Mathieu Gagnon et Jules Randolph pour la bonne humeur et les superbes parties de baby-foot. Enfin, tout ceci n'aurait pu être réalisé sans le soutien financier d'INTER géré de main de maître par sa coordinatrice Nathalie Hamel.

Abréviations

AP@LZ Agenda Personnel pour ALZheimer

AVQ Activité de la vie quotidienne

CIG Composant d'Interface Graphique

DTA Démence de type Alzheimer

DOMUS DOmotique et informatique mobile de l'Université de Sherbrooke

Eclipse RCP Eclipse Rich Client Platform (la plateforme client riche utilisée par Eclipse)

GEDOPAL GEstion des DOnnées de PAtients et des Logs

GOMS Goals, Operators, Methods, and Selection ou Buts, Opérateurs, Méthodes et règles de Sélection en français

IHM Interface Homme-Machine

RNPP Réseau de Neurones à Poids Partagés

RPV Réseau Privé Virtuel

WIMP Windows, Icons, Menus and Pointing device

Table des matières

Liste des figures

Introduction

1 Vieillissement de la population

Dans les sociétés occidentales, la baisse de la natalité conjuguée à l'allongement de la durée de vie provoquent des bouleversements dans la pyramide des âges en augmentant la proportion de personnes âgées. Ce phénomène va s'amplifier dans les prochaines années. En effet, l'Organisation mondiale de la Santé (l'OMS) prédit qu'entre 2000 et 2050, la proportion des plus de 60 ans dans la population va doubler. Comment prendre en charge le nombre grandissant de personnes âgées en perte d'autonomie dans des structures qui sont aujourd'hui à peine suffisantes ?

Dans le cadre du vieillissement, les personnes âgées sont particulièrement vulnérables à certaines pathologies qui diminuent leurs facultés cognitives. Dans son rapport intitulé « Une nouvelle façon de voir l'impact de la maladie d'Alzheimer et des maladies apparentées au Canada » [13], la société Alzheimer Canada estime le nombre de Canadiens vivant avec des troubles cognitifs, dont la maladie d'Alzheimer, à 747 000 personnes en 2012 et à 1,4 million en 2031. Cela représente 14.9% des Canadiens de 65 ans et plus. Ce même rapport indique qu'aujourd'hui « un Canadien sur cinq, âgé de 45 ans ou plus fournit, sous une forme ou une autre, des soins à des aînées ayant des problèmes de santé à long terme ». Ces aidants naturels ont consacré en 2011 « un peu plus de 444 millions d'heures non rémunérées à prendre soin d'une personne atteinte de la maladie d'Alzheimer ». D'ici à 2040, ils devraient consacrer aux soins « le chiffre effarant de 1,2 milliard d'heures non rémunérées par an. »

La maladie d'Alzheimer conduit très souvent à un placement en institution du

patient. La perte d'autonomie devient telle qu'il n'est plus possible pour les proches de faire autrement. En dehors des considérations budgétaires (une institutionnalisation coûte cher), une solution alternative, probablement plus humaine serait de permettre aux personnes de demeurer chez elles plus longtemps en leur fournissant de l'assistance. Une méthode souvent utilisée est de modifier l'environnement de vie des patients pour leur apporter de l'aide durant leurs AVQ (Activités de la Vie Quotidienne). Par exemple, on peut ajouter des capteurs dans leur lieu de vie pour détecter certains problèmes, comme l'oubli de la prise d'un repas afin de leur rappeler le moment venu.

Une des motivations du laboratoire DOMUS est de développer des outils d'assistance cognitive à destination des personnes ayant des troubles cognitifs pour améliorer leur autonomie à domicile. De par la nature de la population cible, les outils créés doivent être faciles d'utilisation ce qui signifie que les interfaces doivent être simples à comprendre et à utiliser.

Parmi les applications créées par le DOMUS, certaines sont à destination des personnes atteintes de démence de type Alzheimer (DTA). Une en particulier va nous servir de cas d'étude : AP@LZ.

Cette application AP@LZ (Agenda pour Personne ALZheimer) est une application Android, disponible sur téléphone intelligent, qui permet aux personnes présentant une démence de type Alzheimer (DTA) de rester autonomes plus longtemps. La fonction principale de cette application est la gestion de leur emploi du temps. Dans cette section, nous présenterons rapidement les fonctionnalités et interfaces de l'application que nous verrons en détail plus loin.

2 Études d'utilisabilité

Plusieurs études d'utilisabilité ont été effectuées avec AP@LZ, avec des personnes âgées saines et des personnes âgées atteintes de la maladie d'Alzheimer [24, 23]. Ces études ont permis d'améliorer l'interface. Cependant, il nous est maintenant difficile de continuer à améliorer AP@LZ sans changer la manière dont nous effectuons nos

études d'utilisabilité[1] car il semblerait que la limite de l'utilité des études que nous effectuons ait été atteinte, et ce pour deux raisons.

La première est due au fait que les tests des études d'utilisabilité ont lieu sous la surveillance des chercheurs. Cela pose deux problèmes : tout d'abord, le participant se sent observé et risque d'agir différemment que s'il n'y avait personne. Ensuite, ces études ont lieu dans un espace-temps limité. En effet, le patient expérimente l'application pendant un temps relativement court et souvent selon un scénario prédéfini. Ces problèmes signifient que le test ne correspond pas tout à fait à une utilisation réelle de l'application dans la vie quotidienne du patient.

La deuxième raison vient des utilisateurs finaux. Il est difficile pour une personne atteinte de la maladie d'Alzheimer de donner son avis ou de se souvenir des problèmes rencontrés durant l'utilisation. Cela rend l'amélioration plus difficile, car l'évaluation de l'utilisation dans la vie courante est difficile.

Ces problèmes sont récurrents dans l'évaluation de l'utilisabilité des applications créées par le DOMUS. C'est pourquoi nous avons décidé d'utiliser d'autres outils pour effectuer des études d'utilisabilité.

Des études d'utilisabilité sont fréquemment effectuées par le DOMUS sur différentes applications et au cours des années, nous avons remarqué qu'elles prennent beaucoup de temps et nécessitent la mise en place de nombreuses ressources. De plus, durant ces études, un grand nombre de données sont produites, souvent difficiles à gérer. Ces données doivent être conservées et accédées régulièrement pour l'analyse par les chercheurs. Il serait très pratique pour ces derniers d'avoir des outils automatiques. C'est pourquoi nous avons décidé de créer des outils permettant de faciliter leur mise en place. Ces outils, qui seront testés sur AP@LZ pourront servir de cadre pour l'analyse d'autres applications créées par le DOMUS.

Nous introduirons dans le premier chapitre (Chapitre 1) les termes courants d'analyse d'interface ainsi que l'état de l'art en analyse d'utilisabilité. Nous présenterons ensuite (Chapitre 2) nos objectifs ainsi que la méthodologie utilisée pour les atteindre. Viendront ensuite trois chapitres présentant les solutions trouvées pour réaliser ces

1. Définitions et compléments d'information sur l'utilisabilité et les études d'utilisabilité au Chapitre 1

objectifs, le premier (Chapitre 3) portera sur les données produites durant les études d'utilisabilité et l'outil que nous avons créé pour les gérer, le second (Chapitre 4) présentera le processus utilisé durant l'analyse formelle de notre cas d'étude, enfin le troisième (Chapitre 5) présentera les outils que nous avons créés pour l'analyse automatique des traces d'utilisation.

Chapitre 1

État de l'art

1 Évaluation d'interface personne-machine

1.1 Utilité et utilisabilité

L'utilité et l'utilisabilité sont deux concepts considérés lorsque l'on parle d'acceptabilité de systèmes, c'est-à-dire le fait de savoir si un système est assez bon pour satisfaire les besoins et les exigences de ses utilisateurs. En informatique, nous nous intéressons surtout à la dimension pratique de l'acceptation, par opposition à l'acceptabilité sociale. Nielsen indique que l'acceptabilité pratique porte sur l'intention que le système permet d'atteindre lorsque l'acceptabilité sociale porte plutôt sur le contexte d'utilisation du système [35]. Toujours selon Nielsen, l'acceptabilité pratique d'un système découle de l'utilité et de l'utilisabilité.

Senach définit l'utilité comme ce que le système permet de faire *via* son interface. Elle porte sur des propriétés telles que la capacité fonctionnelle, les performances du système et la qualité de l'assistance technique proposée à l'utilisateur [44].

L'utilisabilité porte sur la qualité de l'interface personne-machine. Aujourd'hui, elle est définie par la norme ISO 9241-11 [26] comme 'le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié'. Cette norme donne les lignes directrices concernant l'utilisabilité et définit les trois critères

de qualité d'utilisation pour le design d'interface personne-machine :

Efficacité : savoir si le but a été atteint selon des critères de réussite qui doivent être prédéfinis.

Efficience : mesurer les efforts nécessaires pour atteindre le but, Plus les efforts sont faibles, plus l'efficience est grande.

Satisfaction : mesurer à quel point l'utilisateur trouve le système agréable.

Ces trois critères sont évalués, comme l'indique la norme ISO, en prenant en compte le système, mais aussi l'utilisateur, la tâche et l'environnement. L'efficacité et l'efficience ne doivent pas être confondues. Elles sont souvent utilisées indifféremment dans le langage courant leur sens est différent aux mots anglais 'efficiency' et 'efficacity'.

Selon Nielsen, pour évaluer ces trois critères, cinq caractéristiques sont à prendre en considération lors de l'évaluation de l'utilisabilité [35] :

Apprentissage : la facilité avec laquelle l'utilisateur accomplit les tâches basiques lors de la première utilisation. Ce critère peut être mesuré par le temps nécessaire à l'utilisateur pour utiliser l'interface.

Mémorisation : la facilité avec laquelle l'utilisateur se souvient comment accomplir les tâches. Ce critère peut être mesuré en fonction des erreurs et du temps nécessaire à l'utilisateur pour reprendre en main l'interface après une période de non-utilisation.

Fiabilité : le nombre d'erreurs que commet l'utilisateur, la sévérité de ces erreurs ainsi que la facilité avec laquelle l'utilisateur se remet de ces erreurs.

Efficience : la rapidité avec laquelle l'utilisateur peut effectuer les tâches qu'il

souhaite. Cela mesure la productivité que peut atteindre l'utilisateur.

Satisfaction : valeur subjective qui mesure à quel point l'utilisateur trouve agréables les interactions avec l'interface.

Nielsen décompose le premier critère de la norme ISO en trois critères pour le rendre mesurable. Il est préférable que les critères prédéfinis auxquels il est fait allusion dans la norme soient définis de la manière la plus objective possible.

1.2 Étude d'utilisabilité

Il existe plusieurs moyens d'évaluer l'utilisabilité d'une IHM (Interface Homme Machine). Ces techniques et méthodes sont divisées en deux classes, celles avec utilisateurs et celles sans utilisateurs. Cette classification a été faite deux fois de manière assez similaire. La première par Nielsen et Molich, distingue les approches prédictives des approches expérimentales [36] tandis que la seconde, par Senach, distingue les approches empiriques des approches analytiques [44]. Nous garderons pour la suite le vocabulaire employé par le second, car les termes sont employés en français ce qui évite des imprécisions de traduction, les deux classifications étant très proches l'une de l'autre. Senach distingue donc :

Les approches analytiques

Dans les approches analytiques, les interfaces n'ont pas besoin d'être implémentées, car elles sont modélisées. Ces approches ne nécessitent pas la présence d'un utilisateur. En revanche, elles doivent être très précisément définies pour ce qui est des interactions sur les écrans. En effet, les méthodes analytiques ont pour variables les composants d'interface et leur placement, par exemple dans le but de calculer précisément les temps d'interaction. Elles sont particulièrement utiles au début du cycle de vie d'une application. Dans ces approches, il est nécessaire d'avoir une description détaillée des tâches, une analyse approfondie du système et une modélisation de l'utilisateur pour identifier les problèmes d'utilisation potentiels. Nous ne parlerons ici que des modèles prédictifs dont le but est de prévoir les performances d'utilisation.

Des exemples de tels modèles sont les méthodes de GOMS (Goal, Operator, Method, Selector) [6] ou la loi de Fitts[3].

Les approches empiriques

Les approches empiriques ne peuvent être utilisées que sur des applications possédant déjà des interfaces fonctionnelles, au moins partiellement. Elles partagent des caractéristiques communes, identifiées et définies par Dumas et Redish : leur but est d'améliorer le système, les participants sont de vrais utilisateurs qui effectuent de vraies tâches en étant observés selon un protocole rigoureux, les données sont ensuite analysées pour faire des recommandations pour l'amélioration du système [15]. Dans certains cas, ces études peuvent être effectuées par des participants qui ne sont pas des utilisateurs finaux, mais des experts qui donnent leur avis sur l'interface.

Senach définit aussi deux grandes catégories d'évaluation lors des études d'utilisabilité [44] :

Expérimental : l'utilisateur est observé et des données sont récoltées durant son utilisation de l'interface.

Subjective : l'utilisateur donne son avis sur l'interface après son utilisation grâce à des questionnaires ou des entretiens.

Aujourd'hui, la méthode la plus couramment utilisée est mixte, l'utilisateur est observé pendant l'expérimentation puis des questionnaires lui sont donnés que des entretiens viennent compléter. Les données récoltées sont de plusieurs natures : de la vidéo, de l'audio, des traces d'utilisation (toutes les interactions que l'utilisateur a avec l'interface), de questionnaires... Ces données permettent ensuite d'inférer (selon la précision et la validité des mesures effectuées) les difficultés que rencontrent les utilisateurs et de développer des solutions les réduisant [44].

Les approches empiriques, quoique très efficaces, ont leurs limites :

Tout d'abord, elles produisent beaucoup de données et une partie des données est stockée sur des supports numériques de différentes natures (vidéo, audio, tableur, base

de données...) tandis qu'une autre partie est stockée sur des supports papier (prises de notes, questionnaires remplis par les testeurs...). Ces habitudes sont parfois source de problèmes, car les différents supports sont facilement endommagés ou perdus.

De plus, ces données prennent souvent beaucoup de temps à analyser, la littérature mentionne des temps d'analyse qui sont souvent de 5 à 10 fois le temps de collecte et peuvent même parfois atteindre jusqu'à 100 fois ce temps [20].

C'est pourquoi plusieurs études cherchent à automatiser ce processus d'évaluation.

1.3 Évaluation automatique des interfaces utilisateurs

Nielsen [37] a créé les premières analyses automatiques à base d'heuristiques à appliquer aux applications pour compléter l'analyse auprès d'utilisateurs. Ces heuristiques ont été affinées avec le temps permettant une analyse toujours plus précise. Malheureusement, si ces méthodes fonctionnent, elles sont longues à mettre en place et donnent parfois des résultats trop subjectifs. D'autres méthodes d'évaluation, plus automatiques, ont été mises en place, avec un certain nombre de guides d'utilisation. Par exemple, Sanderson et Fisher [16] ont décrit différentes méthodes pour collecter et surtout manipuler les évènements utilisateurs créés durant les études d'utilisabilité dans le but d'en améliorer l'analyse. La mise en place de ESDA (Exploratory Sequential Data Analysis) [17] a demandé la conciliation de plusieurs domaines utilisés durant les études d'utilisabilité tels que l'étude comportementale, sociale, cognitive... C'est ainsi qu'ont été créés les premiers guides pour la mise en place d'analyses automatisées.

Pour mieux classer toutes ces solutions, Ivory définit tout d'abord trois étapes [27], les différentes méthodes d'évaluation d'utilisabilité portant sur l'une ou plusieurs de ces étapes. :

Capture : étape de collecte des données d'utilisabilité qui peuvent être : la vitesse à laquelle une tâche a été effectuée, les erreurs commises, les réponses aux questionnaires...

Analyse : étape d'interprétation des données pour identifier les problèmes.

Critique : étape de suggestion de solutions aux problèmes détectés.

Avec l'expansion des nouvelles technologies, les applications sont devenues de plus en plus complexes à évaluer [29] et d'autres méthodes ont été testées. Les traces d'utilisation sont apparues pour l'évaluation de l'utilisabilité des sites Internet et se sont rapidement exportées vers le paradigme WIMP (Windows, Icons, Menus and Pointing device) [5, 27]. La plupart des cadriciels développés dans ce cadre se concentrent sur une évaluation fondée sur des règles ou des modèles qui ont déjà été extensivement testés. D'autres, en revanche, ont proposé d'utiliser ces nouvelles technologies pour améliorer l'évaluation. Kort [29] combine des 'évènements utilisateurs' (user events en anglais) avec des informations qualitatives, obtenant ainsi de meilleurs résultats.

En dehors du gain de temps, il y a plusieurs avantages à automatiser l'évaluation des interfaces [27], le premier étant que différentes équipes utilisant les mêmes méthodes d'évaluation (manuelle) sur la même interface peuvent trouver des résultats qui diffèrent énormément comme cela a été prouvé en différentes occasions [33, 14, 32, 34]. Dix propose comme solution à ces problèmes d'utiliser dans chaque étude, plusieurs techniques d'évaluation et si possible plusieurs équipes d'évaluateurs [11]. Pour chaque méthode utilisée (heuristique ou test avec des utilisateurs), plusieurs équipes comparent leurs résultats pour obtenir une évaluation moins biaisée.

Ivory montre aussi un certain nombre d'autres avantages à l'automatisation de l'évaluation des interfaces [27] tels que :

— la réduction des coûts et du besoin d'expertise en évaluation de la part des évaluateurs. Ainsi, des designers sans grande expérience de l'évaluation des interfaces peuvent maintenant effectuer des études d'utilisabilité
— l'augmentation du nombre de fonctionnalités testées dans le système, car le temps n'est plus une contrainte aussi importante
— cela permet la comparaison de différents designs, cela permet une meilleure intégration de l'évaluation tout au long du processus de développement

Nous nous concentrerons sur les méthodes qui utilisent les évènements utilisateur, ou traces d'utilisation.

Utilisation des évènements utilisateur

Les évènements utilisateurs incluent toutes les interactions que l'utilisateur a avec l'interface : les mouvements de la souris, les entrées clavier, les clics de la souris, les mouvements des doigts sur l'écran... Ces évènements sont facilement enregistrables de façon automatique et permettent de reconstituer assez précisément le comportement de l'utilisateur. Hilbert et Redmiles ont créé un cadriciel pour classer et comparer toutes les méthodes d'évaluation automatique des interfaces à partir des évènements utilisateur [20]. Ils distinguent cinq approches [20] :

Synchronisation et recherche : synchroniser plusieurs médias par exemple la vidéo ou l'audio avec les évènements utilisateur, pour permettre de chercher dans un média une séquence ou un évènement particulier pour le trouver dans un autre (récupérer le contexte pour une séquence en particulier). Cela permet aussi d'annoter les données en temps réel.

Transformation : transformer les séquences d'évènements très bas niveau en évènements de haut niveau.

Sélection : séparer le bruit du reste, en effet, certains évènements ne sont pas intéressants et doivent être supprimés pour que l'analyse puisse être concluante. Cela permet aussi dans certains cas de ne récupérer qu'un seul type d'évènement sur lequel on souhaite se concentrer (par exemple uniquement les clics sur les boutons) ou supprimer les évènements que l'on ne veut pas voir (par exemple les entrées au clavier).

Abstraction : relier les évènements au concept de plus haut niveau auxquels ils correspondent de façon à en faire ressortir le sens. Cela facilite l'analyse. Exemple : remplacer une suite de clics menant à une impression par l'évènement 'demande d'impression'.

Conversion : après avoir effectué la sélection et l'abstraction, transformer les séquences d'évènements obtenus pour créer une nouvelle séquence d'évène-

ments de plus haut niveau (par exemple transformer toutes les demandes d'impression quelle que soit la manière dont elles ont été demandées [suite de clics, raccourci clavier...] en un seul évènement).

Analyse : trouver des séquences d'utilisation pour les analyser plus en détail

Détection : trouver certaines séquences spécifiques pour les isoler : chercher des séquences qui reviennent souvent ou des séquences qui brisent des règles, par exemple des séquences qui ne devrait théoriquement pas être possibles.

Comparaison : comparer des séquences (définies à l'avance ou non) avec ce qui s'est réellement passé pour chercher des différences et trouver les points où le modèle s'écarte des usages réels.

Caractérisation : créer un modèle à partir des séquences d'utilisation réelles. Cela permet de créer un modèle plus proche de la réalité et de trouver des caractéristiques particulières pour le modèle, ou des séquences qui n'avaient pas été imaginées.

Visualisation : profiter des capacités naturelles d'observation des humains pour présenter les résultats de manière plus immédiatement utilisable. Cette visualisation s'avère particulièrement pratique pour analyser une interface, car cela permet par exemple de mettre en évidence très facilement les zones les plus utilisées de l'interface.

Support intégré : intégrer les outils d'analyse dans l'environnement de l'application, pour permettre une analyse plus aisée et plus rapide.

Beaucoup de logiciels existants utilisent une ou plusieurs de ces approches pour faire l'analyse automatique des évènements utilisateurs. Parmi les plus développés, on peut citer Akers [1], qui utilise les trois principales techniques d'analyse pour détecter les erreurs que commettent les utilisateurs dans des applications dont le but est la création. Ces erreurs ayant été détectées, tous les médias utilisés (captures d'écrans, vidéos, audios) sont synchronisés dans un mini-film de quelques dizaines de secondes. Les participants sont alors placés par paires pour revoir leurs propres films et expli-

quer d'où sont venues les erreurs. Une méthode, présentée par Pitkanen [39], demande aux utilisateurs de signaler les moments critiques d'utilisation qu'ils soient de nature positive ou négative. Ce sont ceux où l'utilisateur n'arrive pas à faire ce qu'il souhaite facilement ou au contraire, trouve qu'une autre action est très facile à faire ou qu'une icône est particulièrement facile à comprendre. Les différents médias utilisés peuvent être de différente nature : capture d'écran, des mouvements de la souris, des entrées clavier, parfois des mouvements des yeux de l'utilisateur sur l'écran... Autour des moments critiques, ces médias enregistrent tous les évènements utilisateur. Ces enregistrements sont ensuite revus par un expert.

Plus ancien, mais avec une philosophie toujours d'actualité, EDEM (Expectation-Driven Event Monitoring) [21] tente de combiner plusieurs de ces approches. L'idée est de créer une plateforme qui aide durant la conception à récolter non seulement les évènements utilisateur, mais aussi toutes les informations que souhaitent faire remonter les utilisateurs. Cela permet une meilleure analyse d'utilisabilité. De plus, les utilisateurs sont ainsi plus intégrés dans la conception du système ce qui rend la conception orientée utilisateur plus efficace.

Une des philosophies utilisées pour évaluer l'utilisabilité est de se concentrer sur une conception centrée non sur l'utilisateur, mais sur l'usage. Constantine présente une solution dans laquelle l'application se voit donner une architecture des évènements utilisateurs qui permet d'évaluer l'utilisabilité [7]. Cette architecture est définie dans les premiers stades de la conception et s'affine peu à peu en une solution [7]. Elle permet d'indiquer à l'application quelles sont les séquences d'interactions attendues pour effectuer une tâche. En comparant cette architecture avec des utilisations réelles, le modèle peut être raffiné, mettre en lumière des problèmes d'utilisabilité et permettre un meilleur retour sur l'utilisabilité au fur et à mesure de la conception [41]. Cette approche permet une comparaison automatique de l'utilisation avec la théorie. Contrairement aux approches centrées utilisateur, les chercheurs se concentrent ici sur le développement d'une architecture. Cela permet de réduire le nombre de tests avec les utilisateurs.

Toutes les méthodes d'analyse automatique sont pensées pour des interfaces d'écran d'ordinateur. Or le laboratoire DOMUS utilise des interfaces de types différents et souvent apparus après les écrans d'ordinateur. Par exemple, notre cas d'étude fonctionne sur un téléphone intelligent. Nous utilisons aussi des tablettes tactiles et des interfaces réparties. Ces interfaces ont des points communs avec les interfaces classiques, mais sont cependant différentes. Par exemple, les applications sur téléphone intelligent ont rarement des menus contextuels (un menu dont le contenu diffère selon le composant où l'utilisateur situe son interaction. Il se trouve souvent sous le clic droit dans les interfaces d'ordinateurs). De plus, il est impossible de prévoir dans quelles conditions l'appareil va être utilisé (la luminosité, le bruit, l'agitation...) ce qui influe sur les erreurs qui peuvent être commises par les utilisateurs. C'est pourquoi un certain nombre de méthodes ont été créées pour adapter l'analyse des traces d'utilisation pour les téléphones intelligents. Ainsi un cadriciel a été utilisé pour simplifier la collecte et l'analyse des traces d'utilisation [2]. Parmi les méthodes les plus développées, on peut citer MATE [40]. MATE (Mobile Analysis Tool for usability Experts) définit un certain nombre de mesures qui sont ensuite agrégées pour évaluer les performances et mettre en lumière celles qui sont exceptionnelles. Cela permet aux chercheurs de détecter les problèmes d'utilisabilité.

Il existe donc des méthodes permettant l'analyse automatique des traces d'utilisation. Malheureusement, l'immense majorité d'entre elles sont créées pour analyser des interfaces d'ordinateur qui correspondent donc au paradigme WIMP. Au laboratoire DOMUS, nous en utilisons peu, car la diversité des interfaces créées pour l'informatique mobile répond mieux à nos besoins. De plus, les méthodes présentées au-dessus sont peu généralisables. En effet, si chaque méthode peut-être adaptée, dans son principe, pour différentes applications et différents supports, cela demande beaucoup de travail. Le laboratoire DOMUS a besoin d'outils qui permettent de gagner du temps dans la mise en place d'études d'utilisabilité, en particulier lors de l'analyse. La solution idéale serait un outil qui permettrait l'analyse automatique des traces d'utilisation, quelle que soit l'application les ayant générées. Cet outil idéal n'est pas techniquement réalisable. En revanche, une solution pourrait être de créer un outil qui prend en compte un grand nombre de nos applications. Cela devrait être facilité par

le fait décrit dans l'introduction : nos applications sont à destination de personnes avec troubles cognitifs et ont donc des interfaces assez simples d'utilisation.

2 L'apprentissage automatique

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle. Son but est d'exploiter automatiquement l'information présente dans un jeu de données. Il permet aujourd'hui d'effectuer beaucoup de tâches différentes : la fouille de données, la classification, la sélection de variables, la régression, etc.

2.1 Principes généraux

Quelle que soit la tâche qu'un algorithme d'apprentissage automatique doit effectuer, il doit apprendre, c'est-à-dire progresser durant une phase d'entraînement (aussi appelée apprentissage). Pour cela, les algorithmes d'apprentissage automatique ont des paramètres qui évoluent lors de l'apprentissage pour pouvoir effectuer leur tâche cible de mieux en mieux. Ces paramètres sont 'appris', c'est-à-dire optimisés durant l'apprentissage pour obtenir la meilleure performance possible sur la tâche définie. Russel et Norvig [43] définissent trois grandes catégories d'apprentissage automatique :

Apprentissage supervisé : un jeu de données est utilisé lors de l'apprentissage. Celui-ci est étiqueté, ce qui signifie que pour chaque entrée du jeu, la sortie (et donc la solution que devrait donner l'algorithme) est déjà connue. Le but de l'algorithme est alors d'apprendre les règles qui permettent d'associer les entrées aux sorties pour pouvoir ensuite effectuer cette tâche sur des entrées dont on ne connait pas la sortie.

Apprentissage non supervisé : un jeu de données est aussi utilisé lors de l'apprentissage. Celui-ci n'est pas étiqueté et c'est à l'algorithme de trouver la structure des entrées pour en déterminer les sorties.

Apprentissage par renforcement : il n'y a pas de jeu de données. L'algorithme interagit avec un environnement dynamique pour atteindre un but (par exemple apprendre un jeu en jouant avec un adversaire). Le résultat retourné

par l'environnement à chaque essai (par exemple victoire ou défaite) permet à l'algorithme d'apprendre.

2.2 La classification

Dans le cadre de ce mémoire, nous souhaitons utiliser l'apprentissage automatique pour effectuer une classification. La classification est une tâche très courante de l'apprentissage automatique qui consiste à utiliser des algorithmes dont le but est d'assigner à chaque entrée une classe [8]. Un classificateur prend en entrée un vecteur $\mathbf{x}$ et lui applique une fonction f qui décrit le classificateur et dépend d'un ensemble Θ de paramètres que l'on va optimiser. Le résultat $\mathbf{y} = f(\mathbf{x})$ obtenu permet d'assigner au vecteur $\mathbf{x}$ une classe, donc de le classifier. Par exemple, un algorithme qui détermine si un courriel est un pourriel en fonction des paramètres donnés en entrée (tels que l'expéditeur, l'objet du mail, la présence ou non de certains mots clefs dans le corps du message...) effectue une classification à deux classes (pourriel ou non). LeCun et al. [31] ont aussi utilisé un algorithme d'apprentissage automatique pour classifier des images de chiffres en fonction de ce qu'elles représentent. Dans ce cas, les entrées sont les différents pixels qui codent les images, et la sortie est le chiffre que représente l'image en entrée. Cette classification se fait sur 10 classes (les dix chiffres).

Lors de l'apprentissage, l'idée est qu'il existe une fonction qui associe parfaitement les entrées aux sorties, c'est-à-dire qui correspond exactement à une classification parfaite. Cette fonction cible est infiniment complexe dans certains cas, et ne peut être calculée. Le but de l'apprentissage est que la fonction f qui relie les entrées et les sorties ($f(\mathbf{x}) = \mathbf{y}$) se rapproche le plus possible de cette fonction cible. Pour cela, il faut donc optimiser l'ensemble Θ des paramètres de la fonction f [43].

Lorsqu'on utilise l'apprentissage supervisé, le jeu de données contient les entrées (par exemple des images de chiffres) ainsi que les sorties associées (les chiffres que représentent ces images). Si l'on utilise l'apprentissage non supervisé alors le jeu de données ne contient que l'entrée et éventuellement le nombre de classes de la classification à effectuer.

Dans le cadre de ce mémoire, nous nous intéresserons seulement à l'apprentissage supervisé. L'apprentissage non supervisé n'a pas été envisagé en raison de la complexité de sa mise en place et le fait que les classes seront connues à l'avance grâce à une analyse formelle des applications. Nous nous intéresserons dans ce document plus précisément à une sous-catégorie d'algorithmes d'apprentissage supervisé permettant la classification : les réseaux de neurones artificiels à apprentissage supervisé.

Nous avons décidé de nous intéresser aux réseaux de neurones artificiels pour deux raisons : ils sont des approximateurs universels et leur plasticité leur permettra de s'adapter facilement à différents types d'entrées tant que celles-ci restent dans un format standard. Cela signifie qu'adapter l'algorithme de classification aux différentes applications du laboratoire DOMUS ne nécessitera que peu de changements dans le code. De plus, nous avons tiré profit de l'expertise locale du département d'informatique de l'Université de Sherbrooke grâce aux cours du professeur Larochelle sur les réseaux de neurones.

L'origine de ces algorithmes est une tentative d'imitation du fonctionnement et des caractéristiques du cerveau humain. Russel et Norvig [43] ont présenté un modèle mathématique simple pour représenter un neurone. Une représentation graphique est montrée en Figure 1.1. Dans ce cas, un neurone est proche de l'algorithme du perceptron décrit par Rosenblatt [42] en 1957.

Le perceptron Le perceptron est donc un algorithme d'apprentissage supervisé. Dans ce modèle, un neurone a n entrées (représentées par le vecteur $\mathbf{x}$ sur la figure) et à chacune d'entre elles est affecté un poids (vecteur $\mathbf{w}$ sur la figure). Le neurone effectue une somme des entrées pondérées par les poids avant d'y appliquer une fonction d'activation g. Lorsque l'on effectue une classification, celle-ci est binaire (il n'y a que deux classes de classification) et la fonction d'activation utilisée est dans la majorité des cas la fonction échelon ou la fonction sigmoïdale. Cette fonction d'activation (ou fonction de seuil) indique si oui ou non le neurone est activé, c'est-à-dire si sa sortie dépasse le seuil. Si oui, la classification de l'entrée sera '1', sinon ce sera '0'. Pour permettre de décaler l'entrée sur l'axe des x, un biais (b sur la figure) peut être ajouté à la somme des entrées. Si l'on ajoute ce biais en tant que poids, dont l'entrée (virtuelle) serait égale à '1', le vecteur de poids $\mathbf{w}$ forme l'ensemble Θ des paramètres

de cet algorithme.

La fonction de sortie f de ce neurone est donc :

$$y = g\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{1.1}$$

Si l'on pose $w_0 = b$ et $x_0 = 1$ (l'entrée fictive du biais), alors la formule est un produit scalaire :

$$y = g\left(\mathbf{w}^T \cdot \mathbf{x}\right) \tag{1.2}$$

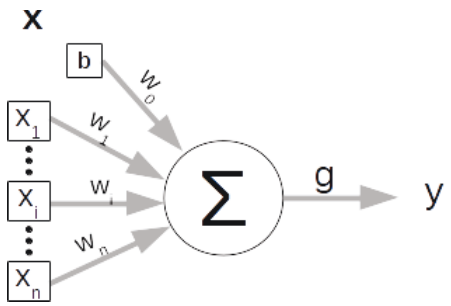


Figure 1.1 – Modèle simple du neurone avec en entrée le vecteur $\mathbf{x}$, le biais b et la fonction d'activation g et en sortie $y = g\left(\mathbf{w}^T \cdot \mathbf{x}\right)$ (en posant $w_0 = b$ et $x_0 = 1$)

2.3 Les réseaux de neurones

Pour créer un réseau de neurones, il faut connecter plusieurs neurones ensemble comme sur les figures 1.2, 1.3 et 1.4. Il existe deux grandes catégories de réseaux de neurones : les réseaux à propagation avant (ou acycliques) et les réseaux récurrents (cycliques). Un réseau est dit cyclique s'il possède au moins une boucle de rétroaction (cf Figure 1.3). Ces boucles permettent au réseau d'avoir une mémoire à court terme. Comme cela n'est pas nécessaire pour la détection d'erreur dans les traces d'utilisation, nous nous intéressons ici seulement aux réseaux à propagation avant dont des exemples sont présentés aux figures 1.2 et 1.4. Les modèles de réseaux de neurones sont très nombreux suite à l'explosion de ce champ de recherche dans les années 90, lorsque les ordinateurs ont commencé à être assez performants pour que les réseaux

puissent contenir un grand nombre de neurones. Nous ne présenterons pas ici tous ces modèles, mais seulement les plus pertinents pour notre travail.

Les réseaux de neurones multicouches

Nous nous intéresserons tout d'abord aux réseaux de neurones multicouches. Cela signifie que les neurones du réseau sont organisés en couches selon le modèle présenté par Bishop [4]. Tous les réseaux ne sont pas organisés en couches comme sur la Figure 1.4 par exemple. Sur cette figure, on peut voir que le neurone A est connecté aux neurones B et C alors que le neurone B est lui aussi connecté au neurone C. On ne peut donc pas définir de couches.

Il existe dans la littérature, deux façons de compter les couches d'un réseau de neurones multicouche. La première considère la première couche (les x_i sur les figures), appelée *couche d'entrée*, comme une couche à part entière et l'autre non. Nous utiliserons la seconde façon de compter dans ce document, ce qui signifie que le réseau de la Figure 1.2 contient trois couches. La dernière couche du réseau s'appelle la *couche de sortie*. Une couche est dite *cachée* s'il ne s'agit pas de la couche d'entrée ou de la couche de sortie. Le réseau de la Figure 1.2 a donc deux couches cachées, une couche d'entrée et une couche de sortie. On dit qu'une couche est *complètement connectée* quand chaque entrée de la couche est connectée à chaque neurone. La connexion n'étant pas forcément utile pour le réseau de neurones, le poids les reliant peut être égal à zéro, ce qui correspond à effacer la connexion. Malgré tout, si la connexion peut exister, cela suffit à dire qu'il s'agit d'une couche complètement connectée.

Par exemple, le réseau de la Figure 1.2 possède une couche d'entrée (à gauche, les x_i), une couche de sortie (à droite, contenant un seul neurone) et deux couches cachées (au milieu, la première contenant quatre neurones et la seconde deux).

Le réseau de la Figure 1.3 est cyclique, car il existe une rétroaction des neurones B vers A et D vers C.

Le réseau de la Figure 1.4 n'est pas organisé en couches, car le neurone A est connecté au neurone B et C qui sont aussi connectés entre eux.

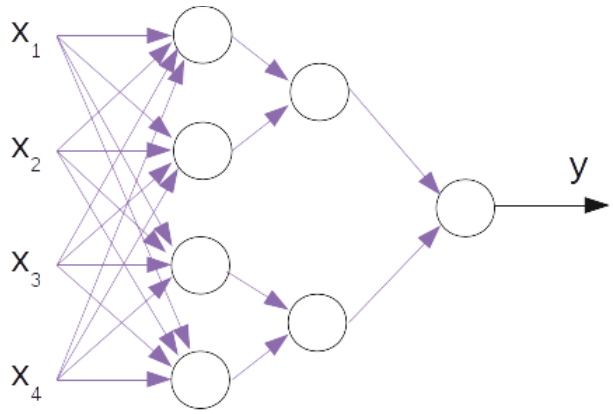


Figure 1.2 – Réseau de neurones à propagation avant (acyclique) multicouche.

Pour calculer $\mathbf{y}$, la sortie du réseau de neurones, on va appliquer la fonction f qui décrit ce réseau à l'entrée $\mathbf{x}$. Pour cela, chaque neurone du réseau va effectuer la même opération qu'un perceptron, c'est-à-dire la somme pondérée des entrées et du biais avant d'y appliquer une fonction d'activation. Pour représenter le réseau de neurones, on va créer une matrice par couche cachée plus une matrice pour la couche de sortie. Ces matrices, appelées matrices de poids et notées W_i, représentent la connexion entre deux couches. Par exemple W_1 représente les connexions entre la couche d'entrée et la première couche cachée. Cette matrice contient les poids utilisés par les neurones de la première couche cachée pour pondérer les entrées. Il s'agit en fait de concaténer en une matrice rectangulaire les différents vecteurs $\mathbf{w}$ représentant chaque neurone. Les paramètres Θ de cette fonction f sont les matrices de poids (les W_i) et les biais ($\mathbf{b_i}$) qui sont en fait les poids reliant les biais égaux à 1 aux neurones. Le mécanisme permettant de calculer $f(\mathbf{x}) = \mathbf{y}$ s'appelle la propagation avant, car les sorties des neurones sont calculées couche par couche de la première couche cachée jusqu'à la couche de sortie. La propagation avant sera présentée plus en détail plus bas.

Les réseaux de neurones multicouches permettent de réaliser un grand nombre d'opérations grâce à leur capacité d'apprentissage. Hecht-Nielsen a démontré en 1987 que n'importe quelle fonction mathématique continue peut être représentée par un réseau de neurones comportant deux couches cachées [38]. Hornik a prouvé quelques

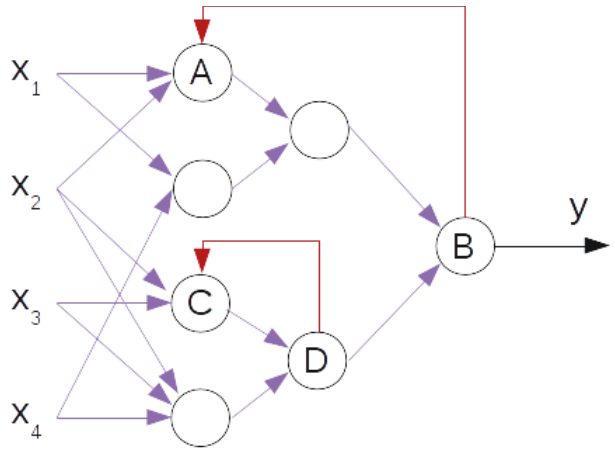


Figure 1.3 – Réseau de neurones cyclique (il existe une rétroaction des neurones B vers A et D vers C).

années plus tard que cela était vrai aussi pour n'importe quel réseau de neurones ne comportant qu'une seule couche cachée, du moment que le réseau possède assez de neurones dans cette couche cachée [22]. C'est pourquoi les réseaux de neurones multicouches sont des approximateurs universels, ils peuvent approximer n'importe quelle fonction mathématique.

Malgré cela, il est parfois difficile de faire converger un réseau de neurones vers la solution optimale (ou fonction cible), c'est-à-dire les paramètres pour lesquels la fonction correspondant au réseau de neurones correspond à celle reliant la sortie et les entrées. Les problèmes rencontrés durant l'entraînement peuvent avoir plusieurs causes [12] :

— un mauvais ajustement des paramètres de l'algorithme d'apprentissage ;
— un nombre insuffisant de neurones dans le réseau.

Dans la majorité des réseaux de neurones multicouches, on utilise pour optimiser les paramètres Θ de la fonction f la rétropropagation de l'erreur grâce à la descente du gradient. Cette technique a été introduite par Werbos [45]. Cet algorithme comporte deux étapes :

— **La propagation avant** : dans cette étape, la sortie du réseau de neurones

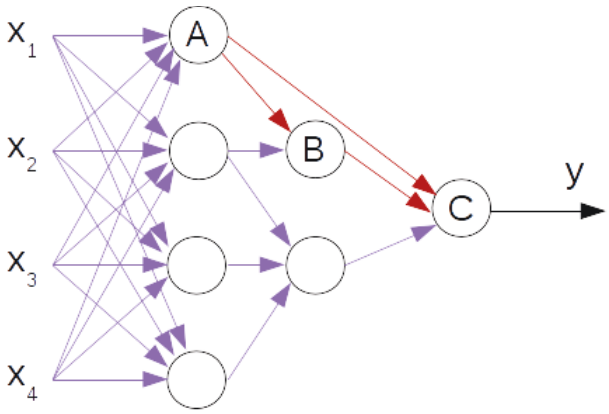


Figure 1.4 – Réseau de neurones à propagation avant (acyclique) non multicouche.

$\mathbf{y} = f(\mathbf{x})$ est calculée. S' il y a n entrées, et p sortie, alors la fonction f est définie par

$$f : \begin{cases} \mathbb{R}^n & \to & \mathbb{R}^p \\ \mathbf{x} & \mapsto & \mathbf{y} \end{cases} \tag{1.3}$$

— **La propagation arrière (ou rétropropagation)** : on calcule ensuite la différence entre la sortie obtenue après la propagation avant et la sortie désirée (c'est-à-dire la classe à laquelle on sait que le vecteur $\mathbf{x}$ appartient). Cette différence est ensuite rétropropagée de la couche de sortie vers la couche des entrées en ajustant au fur et à mesure les différents paramètres. C'est dans cette partie que l'on utilise l'algorithme de la descente du gradient.

La propagation avant La propagation avant se fait en suivant l'équation 1.2 pour chacun des neurones du réseau.

Pour chaque couche cachée, en partant de la première, on calcule la sortie de chaque neurone, c'est-à-dire la somme pondérée des entrées et du biais suivi de l'application de la fonction d'activation g. En appelant W la matrice des poids de la première couche cachée, $\mathbf{x}$ le vecteur d'entrée et $\mathbf{b}$ le vecteur des biais, le vecteur $\mathbf{a}$

représentant la somme pondérée effectuée pour chaque neurone s'écrit :

$\mathbf{a} = W\mathbf{x} + \mathbf{b}$

En applicant la fonction d'activation g à chaque composante du vecteur $\mathbf{a}$, on obtient le vecteur $\mathbf{h}$ qui représente la sortie de chaque neurone de la première couche cachée et donc le vecteur d'entrée de la couche suivante. On procède ainsi pour chaque couche cachée jusqu'à atteindre la couche de sortie. La seule différence dans le calcul de la sortie de la couche de sortie est que l'on n'applique pas la fonction d'activation g à la somme pondérée, mais la fonction de sortie o.

En effet, lors d'une classification, il est d'usage que la couche de sortie comporte un nombre de neurones égal au nombre de classes. La couche de sortie représente alors pour chaque classe la probabilité que l'entrée $\mathbf{x}$ appartienne à cette classe. Dans ce cas, pour la couche de sortie, la fonction de sortie o est la fonction softmax :

$$\text{softmax}(a) = \left[\frac{\exp(a_1)}{\sum_C \exp(a_c)}, ..., \frac{\exp(a_c)}{\sum_C \exp(a_c)}\right] \tag{1.4}$$

Pour un vecteur $\mathbf{a}$ en entrée, la fonction softmax donne en sortie un vecteur dont les composantes sont positives et de somme unité. Chaque composante correspond à la probabilité que $\mathbf{x}$ appartienne à cette classe. Il est ensuite très simple de sélectionner comme classe de résultat celle ayant la probabilité la plus forte (cela correspond à une fonction argmax). S'il y a p classes, une classe peut donc être vue comme un vecteur de p composantes contenant que des '0' sauf un '1' pour la composante représentant cette classe.

La figure suivante (Figure 1.5) représente un réseau de neurones à trois couches, où on appelle $\mathbf{x}$ le vecteur d'entrée. Pour chaque couche i, la somme pondérée des entrées de la couche s'appelle $\mathbf{a_i}(\mathbf{x})$. La sortie de chaque couche i s'appelle $\mathbf{h_i}(\mathbf{x})$ (On a donc ($\mathbf{h_i}(\mathbf{x}) = g(\mathbf{a_i}(\mathbf{x}))$ sauf pour la dernière couche où $\mathbf{h_3}(\mathbf{x}) = \mathbf{y} = o(\mathbf{a_3}(\mathbf{x}))$)

Pour les deux couches cachées, on calcule successivement pour $i = 1$ puis $i = 2$:

$$\mathbf{a_i}(\mathbf{x}) = \mathbf{b_i} + W_i\mathbf{h_{i-1}}(\mathbf{x}) \tag{1.5}$$

avec $\mathbf{h_0}(\mathbf{x}) = \mathbf{x}$.

Puis

$$\mathbf{h_i}(\mathbf{x}) = g(\mathbf{a_i}(\mathbf{x})) \tag{1.6}$$

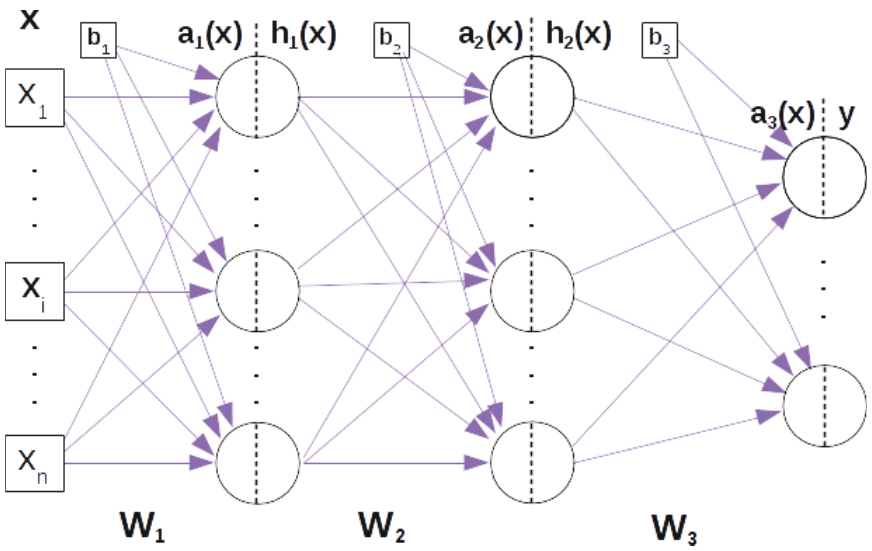


Figure 1.5 – Exemple générique de réseau de neurones multicouche avec 3 couches.

où g est la fonction d'activation choisie, appliquée à chaque composante de $\mathbf{a_i}(\mathbf{x})$.

Enfin pour la couche de sortie, la somme pondérée est la même :

$$\mathbf{a_3}(\mathbf{x}) = \mathbf{b_3} + W_3\mathbf{h_2}(\mathbf{x}) \tag{1.7}$$

Puis la fonction d'activation est différente :

$$y = \mathbf{h_3}(\mathbf{x}) = \text{softmax}(\mathbf{a_3}(\mathbf{x})) \tag{1.8}$$

Ce processus permet donc de calculer $f(\mathbf{x}) = \mathbf{y}$ la sortie du réseau de neurones. Lors du fonctionnement normal du réseau, c'est-à-dire une fois que celui-ci a été entraîné, il suffit d'appliquer la fonction argmax au vecteur $\mathbf{y}$ pour obtenir la classe assignée par le réseau au vecteur d'entrée $\mathbf{x}$.

Lors de l'entraînement, en revanche, il faut comparer ce résultat avec celui attendu et 'corriger' le réseau pour lui permettre d'apprendre. Pour atteindre la solution optimale, on va chercher à minimiser le risque, c'est-à-dire la probabilité que le réseau de neurones 'se trompe' dans la classe de sortie. Il serait logique de minimiser l'erreur de classification, mais ce n'est pas possible, car ce n'est pas une fonction continue. C'est

pourquoi on utilise la fonction de risque, qui dépend de la fonction de perte (appelée loss) :

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{T} \sum_{t} \text{loss}(f_\Theta(\mathbf{x}^t), y^t) \tag{1.9}$$

où Θ est l'ensemble des paramètres que l'on peut optimiser de la fonction f,

t représente l'indice des exemples utilisés pour entraîner le réseau de neurones,

T le nombre d'exemples contenus dans l'ensemble d'entraînement,

$\mathbf{x}^t$ le t^{ieme} exemple de l'ensemble d'entraînement et

y^t la classe à laquelle appartient cet exemple.

Il faut faire attention à différencier $\mathbf{y} = f(\mathbf{x})$, le vecteur de sortie calculé par le réseau et y, la classe à laquelle appartient réellement le vecteur $\mathbf{x}$.

Dans le cas de la classification, la fonction de perte utilisée est la log probabilité négative :

$$\text{loss}(f(\mathbf{x}), y) = -\sum_{C} 1_{y=c} \log(f(\mathbf{x})_c) = -\log(f(\mathbf{x})_y) \tag{1.10}$$

où C est l'ensemble des classes auquel $\mathbf{x}$ peut appartenir

$1_{y=c}$ représente la fonction indicatrice qui vaut 1 si $y = c$, c'est-à-dire si $\mathbf{x}$ appartient à la classe c et 0 sinon

et $f(\mathbf{x})_c$ étant la probabilité déterminée par le réseau que $\mathbf{x}$ appartiennent à la classe c sachant $\mathbf{x}$. Cela correspond en fait à la c^{eme} composante de $f(\mathbf{x}) = \mathbf{y}$.

$f(\mathbf{x})_c = p(y = c|\mathbf{x})$

Pour que le réseau apprenne, il faut donc chercher à optimiser la fonction de perte. Nous allons minimiser cette fonction de perte grâce à l'algorithme de la descente du gradient. Cette phase s'appelle la propagation arrière ou rétropropagation.

Propagation arrière Cet algorithme permet de mettre à jour les paramètres du réseau de neurones pour chaque exemple. On va minimiser la fonction de perte en fonction des paramètres Θ, ici les poids W_i et les biais $\mathbf{b_i}$. Nous utiliserons dans la suite de ce document, la notation suivante : $\nabla_\theta(f)$ représente le gradient par rapport au paramètre θ de la fonction f.

L'algorithme est le suivant :

for N iterations **do**
  for each example (x^t, y^t) **do**

$$\Delta = -\nabla_\Theta(\text{loss}(\mathbf{x}^t, y^t)) \tag{1.11}$$

where the gradient is calculated with respect of each parameter Then each parameter is adjusted :

$$\theta \leftarrow \theta + \alpha \, \Delta \tag{1.12}$$

where α is the step size (it will be explained later)
  end for
end for

On appelle *epoch* une itération sur tous les exemples de l'ensemble. Il y a donc N epochs.

Pour calculer le gradient de la fonction de perte, il faut tout d'abord calculer $\mathbf{y} = f(\mathbf{x})$ (donc faire une propagation avant) puis calculer l'erreur. Ensuite, il faut calculer le gradient de l'erreur avant de le propager en arrière pour mettre à jour les paramètres. Comme expliqué ci-dessus, on cherche à optimiser le risque, donc la fonction de perte. On va donc calculer le gradient de cette fonction en sortie.

On obtient :

$$\nabla_{f(\mathbf{x})}((\text{loss}(\mathbf{x}^t, y^t)) = \nabla_{f(\mathbf{x})}(-\log(f(\mathbf{x})_y) = -\frac{\mathbf{e}(y)}{f(\mathbf{x})_y} \tag{1.13}$$

car $\dfrac{\partial}{\partial f(\mathbf{x})_c}(-\log(f(\mathbf{x})_y) = -\dfrac{1_{y=c}}{f(\mathbf{x})_y}$

où nous rappelons que $f(\mathbf{x})_y$ est la probabilité déterminée par le réseau que $\mathbf{x}$ appartiennent à la classe y (que nous savons être la bonne) sachant $\mathbf{x}$, c'est-à-dire $f(\mathbf{x})_y = p(y = c|\mathbf{x})$

Il faut maintenant calculer le gradient de la fonction de perte par rapport à tous les paramètres, puis le rétropropager afin de mettre les paramètres à jour.

Il faut d'abord calculer le gradient de la fonction de perte en fonction des $\mathbf{a_i}(\mathbf{x})$, puis en déduire les gradients pour les W_i et $\mathbf{b_i}$. Ensuite il faut calculer les gradients des $\mathbf{h_{i-1}}(\mathbf{x})$ puis des $\mathbf{a_{i-1}}(\mathbf{x})$.

On remonte ainsi chaque couche jusqu'à avoir propagé les gradients jusqu'à la couche d'entrée. Une fois les gradients de chaque paramètre calculés, on les met à jour en suivant la formule de la descente du gradient.

Pour un réseau de neurones contenant L couches cachées et donc $L+1$ couches en comptant la couche de sortie, nous obtenons :

$$\nabla_{a_{L+1}}(-\log(f(\mathbf{x})_y)) = -(\mathbf{e}(y) - f(\mathbf{x})) \tag{1.14}$$

Pour plus de lisibilité, jusqu'à la fin du paragraphe, la notation $\nabla_\theta(-\log(f(x)_y)$ sera remplacée par ∇_θ et la fonction $(-\log(f(x)_y)$ sera omise.

Pour propager le gradient, il faut calculer pour chaque couche, de la dernière à la première, le gradient des différents paramètres.

Pour le gradient de la matrice de poids W_i :

$$\nabla_{W_i} = \nabla_{\mathbf{a_i}} * \mathbf{h_{i-1}}(\mathbf{x})^T \tag{1.15}$$

Pour les gradients des vecteurs de biais $\mathbf{b_i}$:

$$\nabla_{\mathbf{b_i}} = \nabla_{\mathbf{a_i}} \tag{1.16}$$

Pour propager le gradient d'une couche à l'autre, il faut calculer :

$$\nabla_{\mathbf{h_{i-1}}} = W_i^T(\nabla_{\mathbf{a_i}}) \tag{1.17}$$

et

$$\nabla_{\mathbf{a_{i-1}}} = \nabla_{\mathbf{h_{i-1}}} \odot \begin{pmatrix} ... \\ g'((\mathbf{a_{i-1}}(\mathbf{x}))_j) \\ ... \end{pmatrix} \tag{1.18}$$

où $\odot$ est la multiplication membre à membre entre vecteurs et g' la dérivée de la fonction d'activation qui est appliquée à chaque membre du vecteur $\mathbf{a_{i-1}}$.

À chaque fois que le gradient de la fonction de perte est calculé par rapport à un paramètre, celui-ci est mis à jour selon la formule de la méthode de descente du gradient présenté plus haut.

Pour vérifier que le réseau apprend réellement, il faut vérifier que la classification effectuée est de plus en plus précise, avec de moins en moins d'erreurs. La solution la plus souvent utilisée pour cette vérification est de contrôler que le taux d'erreur de classification du réseau de neurones diminue au fur et à mesure de l'apprentissage. Une fois entraîné, le réseau de neurones doit pouvoir classifier correctement des exemples non étiquetés, c'est pourquoi ce taux doit être calculé sur des exemples qui n'ont pas été utilisés durant l'entraînement, donc que l'algorithme n'a jamais vus. Il est donc nécessaire de diviser l'ensemble des exemples étiquetés en deux parties. La première, celle qui contient le plus d'exemples, appelée ensemble d'entraînement, servira à entraîner le réseau de neurones. La seconde, plus petite, appelée ensemble de test, permettra de tester que le réseau de neurones progresse. Il est usuel d'effectuer la mesure du taux d'erreur de classification entre chaque epoch pour détecter le moment où le réseau ne progresse plus et arrêter l'entraînement.

On détecte la fin de la progression lorsque la différence entre l'objectif d'optimisation sur l'ensemble d'entraînement et sur l'ensemble de test augmente (indication d'un sur-ajustement des données). Cela signifie que le réseau de neurones est toujours de plus en plus performant sur l'ensemble d'entraînement, mais qu'il devient de moins en moins performant sur l'ensemble de test, donc qu'il généralise de moins en moins bien. En nommant 'look_ahead' la condition précédemment décrite, l'algorithme d'entraînement devient le suivant :

while look_ahead **do**
for each example (x^t, y^t) **do**
forward propagation (computation of the output $\mathbf{y} = f(\mathbf{x})$
Computation of the gradient for each parameter :

$$\Delta = -\nabla_{\Theta}(\text{loss}(\mathbf{x}^t, y^t)) \tag{1.19}$$

where the gradient is calculated with respect of all the parameters.
Then each parameter is adjusted :

$$\theta \leftarrow \theta + \alpha \, \Delta \tag{1.20}$$

where α is the step size, a hyperparameter that will be explained in the next paragraph.
end for
end while

Ainsi, à chaque epoch, on garde en mémoire les résultats de la propagation avant sur les exemples de l'ensemble de test qui nous permettent de calculer l'erreur au sens de la fonction de log-probabilité et du taux d'erreur de classification.

Les hyperparamètres Comme l'optimisation se fait sur une fonction non convexe, il est possible que la descente du gradient converge vers un minimum local. Pour éviter un résultat sous-optimum, une solution est d'initialiser les paramètres Θ du réseau de neurones de manière aléatoire et de comparer le résultat de l'apprentissage pour chaque initialisation. Ce principe est celui des hyperparamètres.

Les hyperparamètres sont des caractéristiques du réseau de neurones qui ne sont pas optimisées lors de l'apprentissage. Ils peuvent être de deux types :

1. Caractéristiques structurelles du réseau : ce sont les paramètres intrinsèques du réseau qui sont utilisés durant l'apprentissage, mais aussi après lors de la classification des nouveaux exemples. Dans notre exemple, le nombre et la taille des couches ainsi que la fonction d'activation utilisée sont des hyperparamètres.

2. Caractéristiques de l'apprentissage : ce sont les paramètres utilisés uniquement durant l'apprentissage et qui n'ont plus aucune pertinence après. Dans notre exemple, le pas α de la descente du gradient ainsi que l'initialisation des poids sont des hyperparamètres.

D'autres hyperparamètres sont souvent utilisés dans les réseaux de neurones multicouches :

Par exemple, le pas α est souvent décomposé en deux paramètres, le taux d'apprentissage tx et la constante de décroissance dc du taux d'apprentissage.

$$\alpha = \frac{tx}{1 + n * dc} \tag{1.21}$$

où n est le nombre d'epochs déjà terminées. Cela permet de donner plus d'importance à la première itération des exemples lors de l'apprentissage.

Un autre type d'hyperparamètres très utilisé sont ceux qui concernent la régularisation.

La fonction de risque telle que présentée plus haut est assez instable. Non seulement à cause des minimas locaux, mais aussi parce qu'elle dépend beaucoup des échantillons. Cela mène souvent à des valeurs de poids inutilement importantes. On ajoute alors une régularisation pour défavoriser les grands poids et rendre le résultat plus robuste vis-à-vis de l'ensemble d'entraînement [30].

La régularisation consiste à ajouter un terme à la fonction de risque qui devient alors le risque empirique défini par :

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{T} \sum_{t} \text{loss}(f(\mathbf{x}^t), y^t) + \lambda \Omega(\Theta) \tag{1.22}$$

où Ω est une fonction des hyperparamètres et des paramètres Θ.

Cette fonction est habituellement constituée de la somme de différentes fonctions de régularisation. Les deux plus courantes sont la régularisation par les normes L1 et L2. La norme L1 d'une matrice correspond à la somme des valeurs absolues de tous les termes de cette matrice.

$$L1(M) = \sum_{i} \sum_{j} |M_{i,j}| \tag{1.23}$$

La régularisation qui lui correspond est la somme des valeurs absolues de tous les termes de toutes les matrices de poids :

$$\Omega(\theta) = \sum_k L1(W_k) \tag{1.24}$$

où $L1(W_k)$ est la norme L1 de la matrice W_k.

La norme L2 d'une matrice est égale à la somme des carrés de tous les termes de cette matrice :

$$L1(M) = \sum_i \sum_j M_{i,j}^2 \tag{1.25}$$

De même, la régularisation qui lui correspond est la somme des carrés de tous les termes de toutes les matrices de poids :

$$\Omega(\theta) = \sum_k L2(W_k) \tag{1.26}$$

Les hyperparamètres correspondant à ces normes sont souvent appelés $l1$ et $l2$ et correspondent au poids que l'on va donner à ces normes dans la fonction Ω :

$$\Omega(\theta) = l1 * \sum_k L1(W_k) + l2 * \sum_k L2(W_k) \tag{1.27}$$

L'ajout de la fonction Ω à la fonction de risque empirique change légèrement les formules d'application de la descente du gradient. Il faut maintenant ajouter (et donc calculer) le terme de la régularisation de $\triangle$ qui devient :

$$\triangle = -\nabla_\Theta(\text{loss}(\mathbf{x}^t, y^t)) - \lambda\nabla(\Omega(\Theta)) \tag{1.28}$$

Lorsque l'on met à jour les poids, il faut donc ajouter le gradient de la fonction de régularisation. En utilisant les mêmes notations que plus haut, le gradient des matrices de poids W_i devient :

$$\nabla_{W_i} = \nabla_{a_i} * h_{i-1}(\mathbf{x})^T + l1 * \text{signe}(W_i) + l2 * 2W_i \tag{1.29}$$

où $\text{signe}((W_i)_{n,p} = \dfrac{W_{n,p}}{|W_{n,p}|}$

Notre algorithme devient donc :

while look_ahead **do**
  for each example (x^t, y^t) **do** forward propagation (computation of the output $\mathbf{y} = f(\mathbf{x})$)
Computation of the gradient for each parameter (with regularization) :

$$\Delta = -\nabla_{\Theta}(\text{loss}(\mathbf{x}^t, y^t)) - \lambda \nabla(\Omega(\Theta)) \tag{1.30}$$

where the gradient is calculated with respect of all the parameters.
Then each parameter is adjusted :

$$\theta \leftarrow \theta + \alpha \, \Delta \tag{1.31}$$

where α and λ are functions of the hyperparameters.
  end for
end while

Il existe deux manières de choisir les hyperparamètres. La recherche sur une grille consiste à donner à chaque hyperparamètre un ensemble de valeurs possibles puis à essayer toutes les combinaisons possibles des différentes valeurs de chaque hyper paramètre. La recherche stochastique est une recherche aléatoire. Pour chaque hyper-paramètre, un intervalle de valeur est donné et à chaque essai, les hyperparamètres sont initialisés aléatoirement dans leur intervalle.

Pour savoir quelle combinaison d'hyperparamètres est la meilleure, il est nécessaire de comparer le taux d'erreur de chaque combinaison. Pour éviter que la comparaison des taux d'erreur soit faite sur des exemples déjà vus par le réseau de neurones, il est nécessaire de créer un troisième ensemble d'exemples appelé l'ensemble de validation. Il y a donc trois ensembles :

L'ensemble d'entraînement : C'est l'ensemble qui sert à entraîner le réseau de neurones, sur lequel on effectue la propagation puis la rétropropagation avec mise à jour des paramètres.

L'ensemble de validation : C'est l'ensemble qui permet de choisir la meilleure configuration d'hyperparamètres possible, sur lequel on effectue uniquement la propagation avant pour déterminer le taux d'erreur de classification.

L'ensemble de test : C'est l'ensemble qui permet d'évaluer la performance du réseau de neurones, sur lequel on effectue uniquement la propagation avant pour déterminer le taux d'erreur de classification et vérifier que le réseau classifie correctement des exemples jamais rencontrés. Ce sont les résultats sur cet ensemble qui permettent de mesurer la performance de l'algorithme. Ces résultats permettent en effet de comparer les performances d'algorithmes de différentes natures.

Les réseaux de neurones multicouches ont malgré tout des inconvénients. Par exemple, il n'existe pas de méthode déterministe pour choisir la structure du réseau (combien de neurones et comment sont-ils organisés, combien de couches cachées...). Cette structure (qui fait partie des hyperparamètres) est très souvent déterminée de manière empirique. De plus, quelles que soient la structure du réseau et la méthode utilisée pour l'optimiser, il y a toujours un risque de converger vers un minimum local. Une solution pour limiter ce risque est d'utiliser un hyperparamètre qui permet l'initialisation des paramètres du réseau à différentes valeurs.

Les réseaux de neurones à convolution

Il existe d'autres types de réseaux de neurones comme les réseaux de neurones à convolution (*Convolutional Neural Network*) qui ont été présentés par Le Cun et Bengio ainsi que De Ridder [9, 10]. Ces réseaux (aussi appelé RNPP pour Réseaux de Neurones à Poids Partagés) ont d'abord été utilisés dans la reconnaissance vocale puis l'usage s'est généralisé dans la reconnaissance d'images, en particulier dans la reconnaissance de caractères manuscrits.

Ces réseaux ont été créés, car, à l'époque, faire des couches complètement connectées avec des entrées en deux dimensions telles que des images produisait un grand nombre de paramètres, ce qui dépassait les capacités des ordinateurs. Comme les entrées sont des matrices et non plus des vecteurs, la quantité d'information est beaucoup plus importante. Il a donc fallu trouver une manière de sélectionner l'information importante pour en réduire la quantité. La structuration des entrées en matrices permet par exemple d'ajouter une dimension temporelle, indispensable dans la reconnaissance vocale.

Ces réseaux sont aussi structurés en couches, mais celles-ci ne sont pas complètement connectées. Les poids qui relient les couches sont communs à tous les neurones. Le réseau est constitué d'une alternance de couches de convolution et d'échantillonnage.

Couche de convolution L'opération de convolution discrète de la fonction f par le filtre h est définie par :

$$g(x) = (f \star h)(x) = \sum_{i=-\infty}^{\infty} f(x-i)h(i) \tag{1.32}$$

Très souvent, h et f ne sont pas défini sur tout $\mathbb{Z}$. Dans ce cas, pour la convolution classique (ou étroite), g est définie pour toute valeur de x telle que $f(x-i)$ est définie pour tout i appartenant au domaine de définition de h.

Par exemple, la convolution du vecteur **[5 4 2 3 5 4 2]** par le filtre **[3 1 2]** donne le vecteur **[20 19 22 23 20]**. Cet exemple est illustré en Figure 1.6. En règle général, pour un vecteur de taille n et un filtre de taille m, le résultat de la convolution sera de taille $n - m + 1$ (sauf si $m>n$, auquel cas, la convolution n'est pas définie). C'est

parce que le résultat est toujours de taille inférieure ou égale à l'entrée que ce type de convolution est dit étroit.

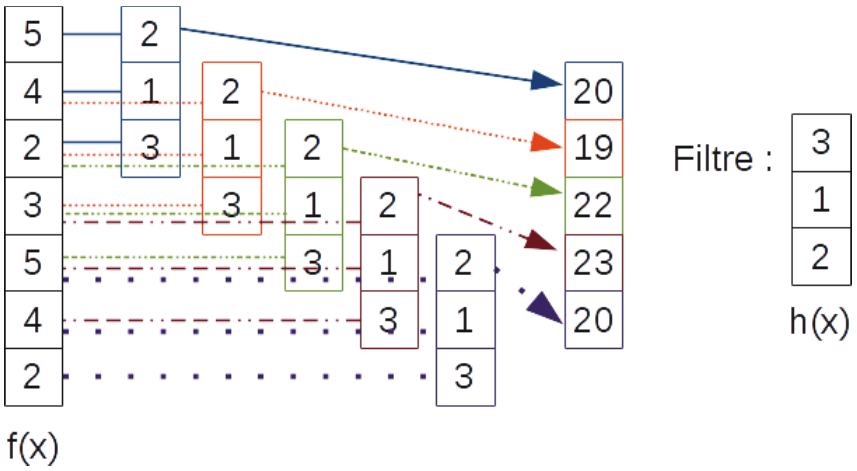


Figure 1.6 – Exemple de convolution sur une entrée vectorielle. On y voit une propriété importante de la convolution : l'invariance par translation. La première et la dernière valeur du résultat sont égales, car elles sont connectées aux trois mêmes valeurs.

Une opération semblable à la convolution est la corrélation. Le résultat g de l'opération de corrélation discrète de la fonction f par le filtre g est définie par :

$$g(x) = (f \heartsuit h)(x) = \sum_{i=-\infty}^{\infty} f(x+i)h(i) \tag{1.33}$$

De même, l'ensemble de définition de g est tous les x pour lesquels $f(x+i)$ est définie pour tout i du domaine de définition de h.

En reprenant l'exemple, la corrélation du vecteur [**5 4 2 3 5 4 2**] par le filtre [**3 1 2**] donne le vecteur [**23 20 19 22 13**]. Cet exemple est illustré en Figure 1.7.

Si l'on définit le filtre $\widetilde{h}$ comme l'inverse du filtre h (pour tout i : $\widetilde{h}(i) = h(-i)$), alors une opération simple de changement d'indice de i en $-i$, permet de voir que, faire une corrélation par le filtre h est équivalent à faire une convolution par le filtre $\widetilde{h}$.

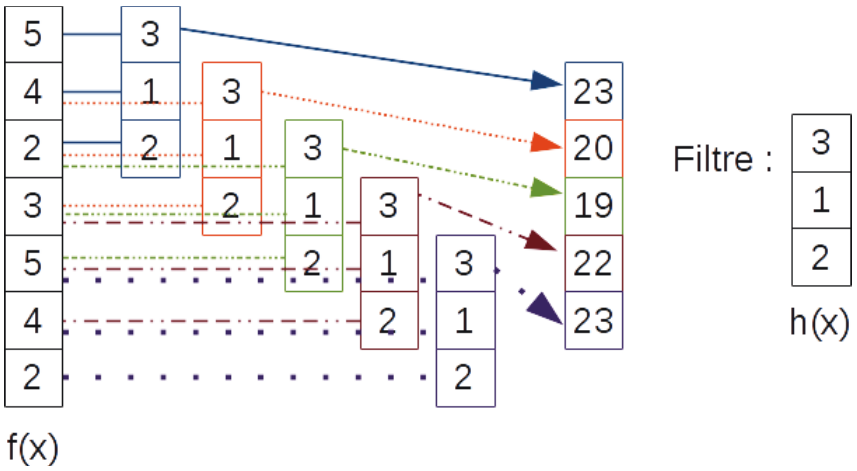


Figure 1.7 – Exemple de corrélation sur une entrée vectorielle. Le filtre utilisé est le même que précédemment. On constate que les opérations sont très proches. Il suffit d'inverser le filtre pour qu'elles soient identiques.

$$\begin{aligned}(f \heartsuit h)(x) &= \sum_{i=-\infty}^{\infty} f(x+i)h(i) \\ &= \sum_{i=-\infty}^{\infty} f(x-i)h(-i) \\ &= \sum_{i=-\infty}^{\infty} f(x-i)\widetilde{h}(i) \\ &= (f \star \widetilde{h})(x)\end{aligned} \tag{1.34}$$

En particulier, pour un filtre symétrique, effectuer une convolution reviens au même qu'effectuer une corrélation. Dans le cas où f et h ne sont pas défini sur tout $\mathbb{R}$, alors la convolution par h et la corrélation par $\widetilde{h}$ sont décalés, elles n'ont pas le même domaine de définition.

La convolution (tout comme la corrélation) peut se faire en deux dimensions. Les définitions deviennent alors :

$$(f \star h)(x,y) = \sum_{i,j=-\infty}^{\infty} f(x-i, y-j)h(i,j) \tag{1.35}$$

$$(f \heartsuit h)(x,y) = \sum_{i,j=-\infty}^{\infty} f(x+i, y+j)h(i,j) \tag{1.36}$$

De même, en changeant les indices, on voit que faire une convolution par un filtre h correspond à faire une corrélation sur le filtre $\widetilde{h}$ inverse du filtre h sur ses lignes et ses colones (pour tout i,j : $\widetilde{h}(i,j) = h(-i,-j)$).

Un exemple de convolution en deux dimensions est présenté en Figure 1.8.

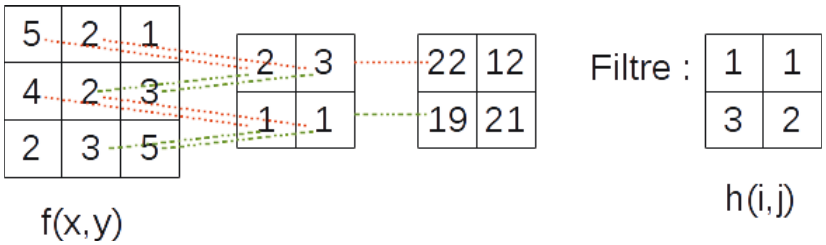


Figure 1.8 – Exemple de convolution sur une entrée matricielle. On voit que le filtre est inversé sur les lignes et les colones pour effectuer la convolution.

L'avantage de la convolution (tout comme la corrélation) est qu'elle conserve les propriétés locales et effectue un traitement qui est invariant par translation. Dans le traitement d'image, cela est nécessaire pour reconnaitre deux fois la même forme si elle se trouve à différents endroits dans l'image. Dans la reconnaissance vocale, cela permet de reconnaitre deux fois le même son à deux moments différents.

Cette étape de convolution restructure l'information pour la diluer afin que l'étape suivante permette de ne sélectionner que l'information importante.

Pour chaque entrée, on effectue plusieurs opérations de convolution chacune avec un filtre différent. Les résultats de ces opérations forment ce qu'on appelle en anglais les *feature maps* et qui peut être traduit 'cartes des patrons' en français.

Par exemple, pour des entrées sous la forme d'images carrées de côté n, avec p filtres de convolution carrés de coté m, on obtient p cartes des patrons de taille $n-m+1$. Cet exemple est illustré en Figure 1.9.

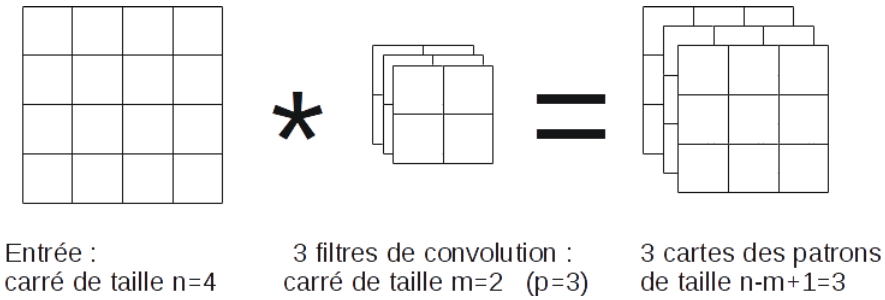


Figure 1.9 – Exemple de la couche de convolution sur une entrée matricielle avec $n = 4$, $p = 3$ et, $m = 2$.

Couche d'échantillonnage C'est dans cette étape qu'a lieu la sélection de l'information. Sur chaque carte des patrons, on va sélectionner uniquement les informations importantes localement. Pour ce faire, chaque carte est découpée en plusieurs voisinages. Sur chacun de ces voisinages, on va effectuer une opération de 'pooling' (que nous traduirons par échantillonnage) qui peut être de plusieurs natures différentes et permet de ne sélectionner qu'une seule valeur par voisinage. On reconstitue à partir de ces voisinages une nouvelle entrée pour la couche de convolution suivante. Cette entrée est en deux dimensions, de taille réduite et contient moins d'information que l'entrée précédente, mais ne contient normalement que de l'information pertinente.

Les deux solutions les plus utilisées lors de l'échantillonnage sont l'échantillonnage par le maximum (aussi appelé sous-échantillonnage) et l'échantillonnage moyen. Dans le premier cas, on sélectionne la valeur la plus élevée du voisinage et les voisinages ne doivent pas se recouper pour ne pas risquer de sélectionner plusieurs fois la même valeur. Dans le second cas, on fait une moyenne de toutes les valeurs du voisinage.

Les propagations avant et arrière seront présentées en détail dans le Chapitre 5 (Sous-section 2.1) où des réseaux de neurones à convolution ont été utilisés et implémentés.

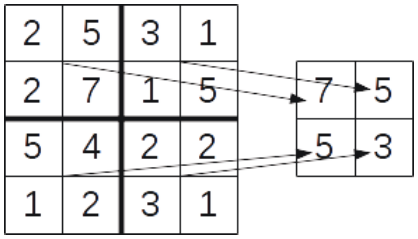


Figure 1.10 – Exemple de sous-échantillonnage, aussi appelé échantillonnage par le maximum.

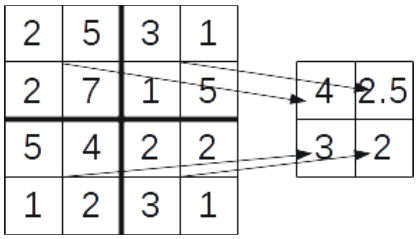


Figure 1.11 – Exemple d'échantillonnage moyen.

Chapitre 2

Objectifs et méthodologie

La revue de littérature présente un besoin de concevoir des outils pour aider à l'analyse automatique de l'utilisabilité des interfaces du laboratoire DOMUS. Dans le cadre des recherches au laboratoire DOMUS, ces interfaces proviennent de prototypes conçus spécifiquement pour des personnes avec des troubles cognitifs, de plus elles sont conçues pour des systèmes souvent plus récents que l'ordinateur de bureau classique tels que les téléphones intelligents, les tablettes ou les interfaces réparties. En raison de la nature de la population cible, les interfaces (comme les applications) doivent donc être simples d'utilisation. Pour chacune des applications du laboratoire DOMUS, une ou plusieurs études d'utilisabilité ont été effectuées par les chercheurs du laboratoire DOMUS (en collaboration le plus souvent avec des chercheurs cliniques). Ces études d'utilisabilité peuvent avoir lieu en laboratoire ou au domicile des participants. Ces différentes études sont très longues et demandent beaucoup d'organisation et de méthode, car il faut faire travailler ensemble des chercheurs n'ayant pas les mêmes domaines de compétence et qui de plus se trouvent souvent dans des lieux différents (et parfois très éloignés). Le but de ce mémoire est donc d'offrir aux chercheurs des outils pour faciliter la mise en place d'études d'utilisabilité. Deux volets ont été traités : d'un côté la sauvegarde et l'accès aux données issues des études d'utilisabilité, de l'autre l'analyse des traces d'utilisation pour détecter les erreurs les plus communes.

Les trois objectifs de cette recherche sont donc :

O1 : créer un outil pour faciliter le stockage, l'accès et la visualisation des données produites durant les études d'utilisabilité.

O2 : utiliser les méthodes d'analyse formelle pour analyser une application du laboratoire DOMUS qui nous servira de cas d'étude : AP@LZ.

O3 : créer un outil permettant l'analyse automatique des traces d'utilisation pour déceler les erreurs des utilisateurs et améliorer l'utilisabilité. Cet outil permettra de gagner du temps durant l'analyse des résultats d'une étude d'utilisabilité. Cet outil sera testé avec notre cas d'étude : AP@LZ.

1 Offrir des outils de stockage et de visualisation des données

Le premier volet de la recherche (qui correspond à O1) porte sur la récolte et le stockage des traces d'utilisation ainsi que des autres données produites durant les études d'utilisabilité. La première étape a lieu sur le support même de l'application (ordinateur, tablette, téléphone intelligent, interfaces réparties...). Dans la plupart des cas, les données récoltées par les expérimentateurs doivent être transmises aux chercheurs qui vont les analyser. Dans un troisième temps, ces données seront visualisées. Ces données incluent les traces d'utilisation, mais aussi des données générées par les applications utilisées (des données de configuration par exemple).

1.1 Cueillette des données

La première étape consiste à récolter les évènements utilisateur dans leur ordre d'apparition. Deux questions principales se posent : quelles sont les informations à récolter lors de ces interactions et comment les récolter ? Plus de précisions seront présentées dans le Chapitre 3 sur les données récoltées et l'outil qui permet de les stocker et de les visualiser.

Informations à recueillir

Il s'agit ici de trouver un compromis entre la quantité, la qualité des informations récoltées et le respect de la vie privée des utilisateurs.

Nous devons respecter la vie privée des patients et nous ne pouvons pas risquer d'enregistrer des données personnelles sans rapport avec l'expérimentation. Malgré cela, les informations récoltées doivent permettre de reconstituer les actions de l'utilisateur et de les analyser. Comme expliqué dans Chapitre 1, il existe des méthodes pour collecter ces traces d'utilisation, qu'il faudra adapter aux besoins particuliers du laboratoire DOMUS et des supports des applications.

Méthode de cueillette

Un autre problème à résoudre est la manière de récupérer ces données. La diversité des supports matériels utilisés au laboratoire DOMUS, rend difficile de recueillir des données de manière unique. En effet, le laboratoire DOMUS utilise des supports très divers : tablette, téléphone intelligent, interface répartie... Ces différents supports fonctionnent sur des systèmes d'opération différents et avec des langages de programmation différents. De plus, les informations à recueillir ne sont pas les mêmes selon la nature de l'interface et le but de l'application. C'est pourquoi, la plupart du temps, nous n'utilisons pas des cadriciels intégrés ou programmés par d'autres organisations, mais plutôt nos propres manières de faire qui, si elles sont moins robustes, nous permettent plus de flexibilité. Ces données sont ensuite stockées dans des bases de données relationnelles.

1.2 Stockage centralisé et visualisation

Les données sont stockées sur l'appareil utilisé par le patient. Il faut les récupérer et les stocker de manière à ce qu'elles soient accessibles par les différents chercheurs. Elles ne doivent pas être dispersées, au risque d'être égarées ou abîmées, et doivent être conservées de manière confidentielle.

Il faut donc tout d'abord recueillir les données sur l'appareil puis les transférer sur un autre support (sans oublier que les supports utilisés par le laboratoire DOMUS sont extrêmement variés). Il faut ensuite transférer les données de l'appareil jusqu'au

laboratoire DOMUS. Les chercheurs ayant besoin d'accéder aux données devront ensuite pouvoir les visualiser depuis différents lieux (à leur domicile, à leur bureau...) Ces étapes doivent toutes être sécurisées pour assurer la confidentialité.

Un outil permettant le stockage et la visualisation de ces données et respectant ces contraintes est nécessaire.

Différents types de données

Les données sur lesquelles se concentre le travail présenté ici sont les traces d'utilisation pour lesquelles nous avons donc créé une description uniforme pour le laboratoire DOMUS. Il existe d'autres types de données utiles aux chercheurs qui n'ont pas forcément de rapport avec les traces d'utilisation, par exemple les informations sur les patients qui ont généré ces données. La solution devra donc permettre aux chercheurs d'accéder aux différents types de données. Selon la nature de la recherche effectuée, certaines données devront être anonymisées avant d'être transmises tandis que d'autres ne devront pas être transmises du tout.

2 Offrir des outils d'analyse

Le second volet, qui correspond aux objectifs O2 et O3, se concentre sur la création d'outils permettant l'analyse des données produites durant les études d'utilisabilité pour permettre de faire des recommandations dans le but d'améliorer l'utilisabilité. Il existe différentes manières d'analyser une interface, présentées au Chapitre 1. La première, l'approche empirique, s'appuie sur la récolte de données avec des utilisateurs. La seconde est l'approche analytique à l'aide de différentes méthodes telles que la méthode GOMS. Les outils créés seront appliqués à notre cas d'étude pour en vérifier la validité.

2.1 Approche analytique

L'approche analytique utilise des méthodes formelles pour modéliser les comportements des utilisateurs face à une interface. Nous avons choisi d'utiliser la méthode GOMS pour analyser nos interfaces. L'analyse des tâches nous permettra d'obtenir

les différentes tâches qu'une application permet d'effectuer ainsi que les différentes manières qui permettent de les effectuer. Cela nous donnera non seulement les classes de notre classification automatique (voir plus loin, 2.2), mais cela nous permet aussi de mieux comprendre l'application et de mieux déterminer quels sont les éléments importants à remarquer lors de l'analyse des traces d'utilisation. Cette analyse sera présentée au Chapitre 4

2.2 Approche empirique

L'analyse des données préalablement uniformisées et stockées requiert le développement d'outils qui seront présentés au Chapitre 5

Analyse statistique

La première méthode d'analyse est une simple analyse statistique. C'est une méthode souvent utilisée durant les études d'utilisabilité. Le but est de savoir quelles fonctionnalités sont utilisées, avec quelle fréquence, quels endroits de l'interface sont les plus fréquemment sollicités... De telles analyses peuvent être faites à l'aide d'un tableur à partir des données uniformisées. Toutefois, des méthodes plus élaborées permettant une représentation plus facile à déchiffrer au premier coup d'œil en utilisant les facultés naturelles d'analyse visuelle des êtres humains (repérer un schéma dans une figure complexe) sont aussi un moyen de faciliter l'analyse.

Analyse des traces d'utilisation

Le but de cette analyse est d'essayer de détecter les erreurs commises par les utilisateurs à partir des traces d'utilisation. L'analyse des traces d'utilisation est fastidieuse, même en utilisant différentes solutions telles que celle présentée dans l'état de l'art, car un certain nombre de tâches répétitives doivent être faites par les chercheurs en amont ou en aval. Pour permettre une analyse automatique des traces d'utilisation, nous avons décidé d'utiliser une méthode inédite : l'apprentissage automatique.

L'algorithme sélectionné pour cet apprentissage est le réseau de neurones. Il fait partie des algorithmes d'apprentissage supervisé. Cela signifie que pour l'utiliser, il est nécessaire d'avoir un ensemble d'entrainement avec des données déjà classifiées.

Étiqueter «à la main» les séquences de traces d'utilisation prend beaucoup de temps, c'est pourquoi nous avons décidé de créer une application qui permet à la fois de classifier et d'annoter les traces existantes. Cet outil permet de gagner du temps lors la classification des données d'entrainement, mais aussi de revoir sous forme de «films» les différentes interactions de l'utilisateur avec son interface (du moment que l'application possède pour seule interface, une interface non distribuée de type téléphone intelligent, tablette ou ordinateur).

Notre but est de classifier les traces d'utilisation pour détecter les erreurs. Pour cela, nous avons défini les classes comme les différentes tâches que l'application permet d'effectuer. La liste des tâches réalisables avec une application ainsi que les schémas qui leur correspondent sont obtenus durant l'analyse analytique.

Chapitre 3

Représentation, stockage et accès aux données

Pour atteindre notre objectif 1 : créer un outil pour faciliter le stockage, l'accès et la visualisation des données produites durant les études d'utilisabilité, il a d'abord fallu définir quelles étaient les données produites durant les études d'utilisabilité. Ces données sont de natures très différentes et les difficultés qu'elles peuvent poser sont donc très diverses.

1 Représentation des traces d'utilisation

Les traces d'utilisation sont très riches en information et, comme expliqué dans le Chapitre 2, elles ont déjà été utilisées pour des études d'utilisabilité. Elles permettent une analyse automatique pour gagner du temps durant la partie analyse d'une étude d'utilisabilité.

Ces données soulèvent malgré tout quelques questions. Tout d'abord, il faut trouver un équilibre entre le respect de la vie privée des patients et la qualité des données récoltées.

En effet, pour avoir le plus d'informations possible, il serait très pratique de pouvoir filmer l'utilisateur à chaque interaction avec l'écran, mais cela est impossible pour

plusieurs raisons.

Tout d'abord, si les vidéos contiennent beaucoup d'informations, elles sont aussi très longues à analyser, car il n'y a pas d'outils automatiques. Nous souhaitons obtenir un outil qui permet d'analyser automatiquement l'utilisation du plus grand nombre possible d'applications au laboratoire DOMUS.

Ensuite, si l'utilisateur se sait filmé, cela risque de changer son comportement lors de l'utilisation. En effet, se savoir filmer risque non seulement de le mettre mal à l'aise quant à l'image qu'il renvoie, mais de plus, l'utilisateur risque de se sentir testé et cela risque d'accentuer la pression pour réussir alors que nous ne testons que l'interface.

Enfin, chaque patient a droit au respect de sa vie privée et même en utilisant des capteurs pour activer la caméra uniquement lors d'interactions et en filmant très serré, il est impossible de réduire à zéro le risque de filmer quelque chose totalement sans rapport avec l'expérience. Lorsqu'une expérimentation a lieu en laboratoire, cela ne pose aucun problème, mais, au domicile du participant, cela pose des problèmes éthiques. Une autre solution envisagée a été de capturer en permanence l'affichage de l'écran. Cela permet de reconstituer toutes les actions utilisateur, mais pose deux problèmes. Le premier problème est la taille du fichier ainsi créé. Cette taille peut être réduite en enregistrant uniquement les moments où il y a des interactions, cependant, pour peu qu'il y ait beaucoup d'interactions, la taille du fichier pourrait tout de même être conséquente. Le deuxième problème concerne de nouveau le respect de la vie privée de l'utilisateur. Si toutes les données entrées par l'utilisateur sont enregistrées, de nombreuses données à caractère privé le seront. De plus, le fichier à analyser serait toujours une vidéo, qui serait donc très longue à analyser et donc l'analyse ne serait pas facile à automatiser.

C'est pourquoi nous devons trouver une manière de récolter les données qui soit transparente pour l'utilisateur tout en nous permettant de reconstituer ses actions. Nous avons donc décidé d'utiliser les traces d'utilisation.

Même en se limitant aux traces d'utilisation, le problème de la vie privée se pose encore. Une application dont le but est de garder en mémoire des informations médicales ou financières ne doit pas récolter le même type d'informations qu'un assistant culinaire. Les entrées clavier d'un assistant culinaire ne contiennent que des entrées de recette ou des commentaires de l'utilisateur. Avec AP@LZ en revanche, nous risquons

de trouver dans les entrées clavier des informations personnelles sur l'entourage de l'utilisateur. Ces données sont étrangères à l'étude et l'entourage de la personne n'a jamais signé de formulaire de consentement.

1.1 Différentes sources

Le DOMUS développe des applications pour assister des personnes avec troubles cognitifs dans leur vie de tous les jours. La population cible du DOMUS est donc très large et requiert des applications qui assistent un grand nombre d'AVQ (Activités de la Vie Quotidienne). Les AVQ ont lieu dans différentes circonstances et les supports matériels et logiciels doivent donc tenir compte des contraintes d'environnement et d'utilisation de chacune de ces AVQ. Certaines applications du DOMUS sont destinées à la mobilité tandis que d'autres n'ont d'utilité que dans le cadre du domicile : un assistant pour faire la cuisine n'est pas utilisé dans les mêmes circonstances qu'un assistant pour se laver les mains ou pour se déplacer dans la rue. Les supports utilisés sont donc variés : téléphones intelligents, tablettes tactiles, ordinateurs... Ces interfaces ont des utilisations différentes et ne fonctionnent pas avec le même langage de programmation, c'est pourquoi il est difficile de choisir une seule manière de faire.

De plus, lors de la récolte des informations, il y a encore un problème de confidentialité, même en se limitant aux traces d'utilisations. Tout d'abord, le niveau de respect de la vie privée dépend de la manière dont sont analysées les données et de la nature plus ou moins sensible des informations produites. Selon les expérimentations, il est possible que les chercheurs cliniques aient accès à certaines données auxquelles les chercheurs en informatique n'ont pas accès.

Dans tous les cas, les données récoltées ne sont pas utilisées par tout le monde de la même manière : dans le cas d'étude en aveugle, par exemple, certains chercheurs ne doivent pas connaître certaines des données.

1.2 Différents usages

L'analyse des données des études d'utilisabilité peut avoir plusieurs buts. Pour le DOMUS, il s'agit principalement de trouver des erreurs dans l'utilisation de l'interface pour déterminer s'il est possible d'en améliorer l'utilisabilité. Pour les chercheurs cli-

niques, l'analyse est différente, il s'agit souvent d'obtenir des fréquences d'utilisation des fonctionnalités pour déterminer comment le système influe sur la vie de l'utilisateur. Ces différents usages requièrent différentes informations qui doivent toutes être enregistrées.

1.3 Description uniforme des traces d'utilisation du DOMUS

Toutes ces observations nous ont décidés à mettre en place une description uniforme des traces d'utilisation récoltées par le DOMUS. Nous avons décidé de nous limiter aux applications ne comportant qu'un seul écran (laissant donc de côté les interfaces distribuées)

Voici la description que nous avons créée :

Timestamp :	date et heure auxquelles a eu lieu l'interaction
Écran :	écran sur lequel a eu lieu l'interaction
Type d'interaction :	clic, double-clic, glissement (de l'anglais swipe)...
CIG :	composant d'interface sur lequel a eu lieu l'interface
X absolu :	abscisse absolue du point de l'écran où a eu lieu l'interaction
Y absolu :	ordonnée absolue du point de l'écran où a eu lieu l'interaction
X relatif :	abscisse relative du point de l'écran où a eu lieu l'interaction
Y relatif :	ordonnée relative du point de l'écran où a eu lieu l'interaction

Le timestamp peut par exemple être le nombre de millisecondes depuis l'Epoch (1er janvier 1970).
La nature de l'écran dépend du support utilisé, il peut s'agir d'une activité Android par exemple ou un objet Panel en Java.
Le CIG peut être un bouton ou une liste, mais aussi le clavier ou une fenêtre.
Les coordonnées absolues indiquent le point de départ de l'interaction si l'interaction en comporte plusieurs (un glisser-déposer par exemple).
Les coordonnées relatives servent à comparer les séquences d'utilisation qui ont eu lieu sur la même application, mais sur des écrans de taille différente. Pour les obtenir, on divise les coordonnées absolues par la taille de l'écran.

Dans le cas, où plusieurs interactions ont lieu simultanément, par exemple plusieurs doigts en même temps sur l'écran, chacune des interactions pourrait être stockée indépendamment, mais en comportant toutes le même timestamp.

Si, pour chaque application, il faudra tout d'abord réfléchir à la précision avec laquelle les informations seront récoltées (entrée au clavier ou non par exemple), cette uniformisation devrait permettre de stocker les traces d'utilisation produites par les différentes applications du DOMUS dans les mêmes modèles de données. Cela signifie que les différents outils d'analyse développés pourront être utilisés assez facilement, quelle que soit l'application qui a créé les données. À cause de la grande disparité des supports et des applications, il est évident que le standard ne pourra pas être respecté à la lettre, certaines autres informations devant être stockées. Malgré tout, ce standard prend en compte un certain nombre de possibilités. L'un des outils dans lequel ce standard sera d'une grande utilité est GEDOPAL.

2 GEDOPAL

GEDOPAL (pour GEstion des DOnnées de PAtients et des Logs) est l'outil que nous avons développé pour répondre à notre objectif O1 : créer un outil pour faciliter le stockage, l'accès et la visualisation des données produites durant les études d'utilisabilité. Il existe certaines tâches que les chercheurs doivent accomplir à chaque étude d'utilisabilité et qui prennent beaucoup de temps.

Durant une étude d'utilisabilité, beaucoup de données sont produites. Par exemple, pour chaque participant, un dossier contenant ses informations personnelles est créé. Une des préoccupations majeures des chercheurs est de stocker ces données de manière à ce qu'elles soient accessibles lorsque l'on en a besoin tout en restant confidentielle pour protéger la vie privée des participants. Il arrive aussi souvent que des études aient lieu dans différents emplacements. Le partage des données devient alors complexe, surtout lorsqu'une partie doit être anonymisée. Le but de GEDOPAL est de rendre ces opérations, souvent complexes et chronophages, transparentes pour les utilisateurs.

Pour cela, GEDOPAL utilise une copie locale d'une base de données centrale qui

est synchronisée en Master-to-Master. Cela permet aux chercheurs d'avoir en local les données dont ils ont besoin pour travailler. Une gestion des droits d'accès par rôles couplée à un cryptage des données de base de données permet de protéger les données. L'article suivant présente en détail ce logiciel.

A tool to help store, access and visualize the data created during usability studies

Perrine Cribier-Delande, Université of Sherbrooke
Maud Loarer, Université of Sherbrooke
Paul Cornec, Université of Sherbrooke
Hélène Pigot, Université of Sherbrooke

During usability studies, numerous data are produced for analyzing applications. The researchers need to access, extract and analyze these large amount of data regardless of the place they work from (in a laboratory or at home). We created an application, GEDOPAL, that provides researchers a secure access to these data and various analyzing tools. We used a centralized storage with a local replication on each computer with master-master synchronization. The software allows for secure storage of different type of data related to usability studies. Researchers can register all information regarding a usability test. They can precise data about the context of the experimentation: date and place, application tested, application's configuration. The characteristics of the participants are registered: name, age, level of education and the group of experimentation they belong to. Usability data gathered through the experimentation are saved and accessible according to the role of the researcher in the experimentation. Tests were performed on GEDOPAL by different researchers from different fields of research and from different locations (both in and outside laboratory)

Categories and Subject Descriptors: D.2.2 [**Design Tools and Techniques**]: User interfaces

General Terms: Experimentation, Security, Standardization

Additional Key Words and Phrases: automatic usability testing, user experience, graphical user interfaces, database management system

ACM Reference Format:
Perrine Cribier-Delande, Hélène Pigot 2014. A tool to help store, access and visualize the data created during usability studies. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 17 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The DOMUS laboratory, located at Sherbrooke University, QC, Canada, develops applications for people with cognitive deficits to assist them in daily life [Imbeault et al. 2011; Imbeault et al. 2014]. To improve these applications, many usability studies are performed [Imbeault et al. 2010]. The goal of usability studies is to evaluate application use in order to improve usability as defined in the ISO standard 9241-11 [ISO 9241 1998]. They are designed to evaluate the application interface according to usability criteria or to detect design issues in new interfaces [Nielsen 1993; Nielsen and Molich 1990]. During usability tests, subjects give personal information that need to be protected. These data come from experimental studies that are approved by ethics

This work is supported by the INTER, under grant FQRNT.
Author's addresses: P. Cribier-Delande and H.Pigot, Computer Science Department, University of Sherbrooke, 2500 Boulevard de l'Université, Quebec J1K 2R1.

DOI:http://dx.doi.org/10.1145/0000000.0000000

committee, binding researchers to protect private data. This protection is reinforced in case of experimentation involving subjects with medical conditions. Due to the great amount data, issues arise related to:

— safe storage of large amount of data
— Protected access to the data from various working places

To tackle these two issues, the DOMUS laboratory developed a software called "GEstion des DOnnées des PAtients et des Logs" a.k.a. GEDOPAL (meaning management of patient's data and logs). This article first presents the background that led to develop this application and second the specifications and constraints. The third part presents the conception of GEDOPAL and the fourth the implementation and results.

2. BACKGROUND

Usability studies are designed to improve applications. These studies require experimentation with various subjects during which data are gathered including personal information, application use and context of use [Dumas and Redish 1999; Pitkänen et al. 2012]. In this section, we describe the data generated during usability studies and how researchers use them for analyzing usability.

2.1. Usability studies

Usability studies are designed to evaluate the usability of a system according to three criteria : efficiency, effectiveness and satisfaction [ISO 9241 1998]. There can be many goals to a usability study (non-exhaustive list). A usability study may pertain on:

— **Comparison**: Compare two designs to find out which one is better according to predefined criteria.
— **Evaluation**: Evaluate a design for one specific target group) to see if any improvement can be made.
— **Impact**: Evaluate the impact of an application on the every day life of the participants.

The organization of usability tests is divided in four steps that are detailed below:

(1) **Definition of the study** During this first step, most of the choices must be made. The version of the application, the scope (what are the goals of the study) and the target groups must be decided. Because the experimentation involves people, a protocol must be written and approved by an ethic committee.
(2) **Recruitment** The participants must be recruited and correspond to the criteria of the target population. They give their personal information and are assigned to a group. They are often assigned an identification number to ensure privacy.
(3) **Testing** During the test, testing information such as logs on the actions performed by the user, success or errors while doing the test are created. Additional information on the context of the experimentation are also created : where, when, how long the experiment takes as well as the name of the experimenters.
(4) **Analysis** The data produced during the test are analyzed to achieve the defined goal of the usability test. Depending of the kind of usability test, the goals could be:.
    — **Comparison goal**: Determine which design is the most suitable according to the criteria.
    — **Evaluation goal**: Determine the reasons of errors or discomfort and find a solution that improves interfaces to avoid them.

— **Impact goal**: find out how much the system influence the lives of the participants (for example by comparing with the results of a control group or by questioning the participants, their relative and/or their caregiver).

2.2. Experimentation output

Usability studies are very useful to improve an application [Heim 2007], but they produce a lot of data [Hersh 2010], especially when the experimentation takes place in smart environments or natural settings such as participants' homes.

The data are gathered during each of the four steps described above. We split them into two categories: those created before the test (Definition and Recruitment steps) and those created during or after the test (Testing and Analysis steps).

2.2.1. Pre-experimentation data. The first type of data is produced during the experimentation's definition and recruitment. We categorized them into two sub-types: Experimentation definition data and Participants data.

— **Experimentation definition data**: data produced during the definition of the experimentation: application and version tested, scope of the experimentation (criteria and task used to evaluate), location of the test (laboratory, homes, smart environment,...), number and nature of the target groups,... They also contain information such as protocol approved by the committee, informed consent form,...

— **Participants data**: personal data and data related to participation in a specific experimentation. The personal data include names, age, address, level of education, occupation, results of cognitive tests... Sometimes, results of previous cognitive tests are prerequisite.

  Data related to an experimentation are similar for each experimentation: identification number of the participants, the group the participant belongs to, where and when will the test take place for this specific participant...

2.2.2. Experimental data. The second type of data produced during the test is often specific to the application tested. These data will then analyzed in order to improve the application. We categorized them into four sub-types: Usage data, Configuration data, Logs data, Test data.

— *Usage data*: data produced during the use of the application and stored by the application. They are application dependent. When testing an agenda for example, usage data are the appointments scheduled by the participants.

— *Configuration data*: data configuring the application specifically for the participant. Some data are application dependent such as enabled functionalities and some are independent of the application such as language used during test, version of the application tested, hardware information.

— *Logs data*: data containing the recording of all the interactions occurring with the application. They include mouse and keyboard strokes or graphical user interface interactions. In some experimentation, eye movements, video or audio recordings and written observations can be added. If the experimentation took place in a smart environment, the sensors data are also added.

— *Test data*: response in the different tests that are performed for the experimentation. For example, if the goal is G1(Comparison), the results (time taken, difficulties

encountered, help given...) for each task performed for each design during the test. Also, at the end of the test, a questionnaire is often given to the participants to evaluate their satisfaction.

2.3. Data storage

The data produced during experimentations need to be saved in order to provide researchers easy access for future analysis. Depending on the medium the data is stored on (paper, spreadsheet or automatic acquisition on databases) analyzing the data can be complicated by three main aspects: data location, security and researcher skills in using computers.

2.3.1. Location. Applications can be developed by many researchers and laboratories. Also experimentations may take place in various locations, some located several hundred kilometers apart. Researchers need to gather experimental data together and to obtain access to them from any working place. Questionnaires and observation notes are most often registered on paper, increasing the risks of loss and damages during transfer and storage. Although less likely, a numerical version can be lost. Numerical transfer rises issue of compatibility that can lead to unreadable documents. In any case, transferring the data is very time-consuming and must be done with caution.

2.3.2. Security. Participant data include private information such as name, address or phone number that need to be protected according to ethics rules. Moreover, some usability experimentations are conducted inside the participant's homes or with subjects with particular health conditions. Medical information such as diagnosis or medical history must be protected to respect the privacy of the participants as well as medical confidentiality. Also some usage data include private information that could potentially permit identification of the participants. Researchers do not need access to all of the data. Distinctions have been made between researchers that need direct access to the participants and require private information and those who don't.

In this paper, we will use the word "experimenter" to designate researchers that have direct access to the participants and their information and to usability researcher to designate researchers that do not. This does not make any assumption on the actual role of the researchers (medical specialist, clinical researcher or computer scientist).

Two main reasons lead to data protection: protect the private data of the subjects and protect the research.

— In the first case, application may involve personal data. For instance testing an agenda implies that the participants register who they met, where and when. This information has to remain confidential. Experimenters obtains such information about their patients ; however, they are bound by professional secrecy, therefore those information as well as the medical ones should not be available to the usability researchers.
— For a study with a comparison goal, blind experimentations necessitate that the experimenter does not know the group each participant belongs while other researchers need this information.

In both cases, some of the data have to be de-identified before being transferred to some researchers. Researchers use an anonymous identification number that allows them to separate nominal data from anonymous ones.

Access to data then depends on the type of data and on the role of the researchers in the experimentation. In any case, the necessary task of de-identifying the data is very time consuming.

2.3.3. Computer skill. As explained above, all the researchers need to pay attention at task such as de-identification and transfer of the data. Researchers in usability testing come from various domains, computer science, ergonomics or clinical fields. They have various skills in using computers, and the tasks of de-identifying the data can require some complex skills to be performed efficiently.

3. SPECIFICATIONS AND CONSTRAINTS

Derived from these requirements, (issues with experimentations, data location, security and computer skills) the DOMUS laboratory decided to design an application that serves the various goal of researchers. In this section, we present the specifications of the software: first the needs to meet and the functions to fulfill, and second, the constraints to abide by.

3.1. Specifications

The specifications are divided in three sections: access, security and analysis.

3.1.1. Data access. The first goal of our project is to manage the data produced during usability studies, before analyzing them. The researchers need to access the data, from various locations (laboratory, office, clinical buildings or home). Several researchers may want to work on the same data at the same time, the software must thus allow concurrent access to the same data from different places.

The transfer from one researcher to the other must be transparent for the user and the de-identification must be performed automatically. Access rights will be described in section 4.3.

This access must allow researchers to meet their needs. They must be able to create, access, modify and delete data about participants, applications and experimentations.

3.1.2. Security and confidentiality. Accessing the data must be secure to protect confidentiality.

Firstly, access to specific data (such as private information) must be restricted to researchers who are allowed to look at them. A system of access control must be designed to ensure this confidentiality. As explained above, this access control will also be used to protect the integrity of the research, for example during a blind study.

Second, if the access is regulated according to users' right, some security must be present to protect from intrusions from the outside. Because no security is unbreakable, data must also be stored in a secure way to avoid infringement in confidentiality in the event of an intrusion.

3.1.3. Data analysis. The two main actions researchers want to perform are selection of relevant data and use of analysis tools.

Selection. Depending on the goal of the usability study, different types of analysis of the data can be needed. Therefore, the software must allow researchers to select specific and meaningful data. Few data are needed to compare frequencies of use of an application. However there are also cases where a large amount of data is needed to perform analysis.

Analysis tools. Because the amount of data produced is large, many statistics tools exist to analyze data from usability studies [Ivory and Hearst 2001; Hilbert and Redmiles 2000]. It would be helpful to integrate these tools in the software to facilitate their use.

3.2. Constraints

In addition to meeting these specifications, the software must abide by some constraints due to the environment and the users. Those five constraints are presented in this section.

3.2.1. Portability. As the researchers use different operating systems, the software must function on the most commonly used ones in our researcher community: MAC OS, Windows and Linux Ubuntu. The most common version have been selected: on MAC OS, it will have to function on OS X 10.9, (a.k.a. Mavericks) or higher, on Windows, it will have to work on Windows 7 or higher and on Linux Ubuntu on version 12.04 TLS or higher.

3.2.2. Maintenance. The software is designed for a laboratory and is not intended to become a commercial product. Because of the high turn-over of research assistants and students in research laboratories, its maintenance needs must be small. It can not be a full time job to maintain it.

3.2.3. Ergonomic. Pre-existent analysis tools already have interfaces and the researchers are used to them. Because of the above-mentioned turnover, the software will often be used by new users. Researchers from many different scientific domains will use it for usability studies so the interface must respect some usability criteria [Dix et al. 1998] defined in the ISO standard 9241-11 [ISO 9241 1998] such as easy to learn and to remember. A very short tutorial must be created to complete the formation. The installation of the software can be more complex as there are some technical specialists in every research center that are not affected by this high turn-over.

3.2.4. Nomadic use. Researchers perform studies in the laboratory or in various settings such as participant's home. The software must be able to function even if there is no Internet access. Access to updated data will not be possible, so it needs to enable an offline utilization. Researchers may also want to analyze the data from home, therefore a secure access will be needed for home connections. These connections are less secure than the one provided by universities and research centers. In that case, when a secure access is not possible or if the connection is cut, the software must switch to offline mode.

3.2.5. Security at the Université de Sherbrooke. Security to access data is not only constrained by the connections but also by the environment. The Université of Sherbrooke requires that any connection from the outside to one of its servers goes through a VPN (Virtual Private Network) connection via a Cisco's software (Cisco AnyConnect Secure Mobility Client [http://www.cisco.com/ 2015]). Université of Sherbrooke has developed its own authentication method. It assigns roles to the user. Because it is not extensible, the VPN can only be used as an identification and authentication tool. It cannot be used as an authorization tool. Even though we assume that each connection coming through the VPN is who it claims to be, the implementation of the authorization (the rights assigned to this connection) is to be implemented in the software.

Despite the relative security of the VPN, it must be remembered that the data stored by the application must be protected, so the second requirement of the section 3.1.2 (confidentiality and security) still need to be met.

4. CONCEPTION

Taking into account the specifications and constraints, we designed GEDOPAL. An overall view of the application and its context is shown in figure 1. The figure summarizes the implemented software that is described in details in the following sec-

tion.Three main parts of the architecture are presented: the storage and access to the data, the data model and the problem of confidentiality.

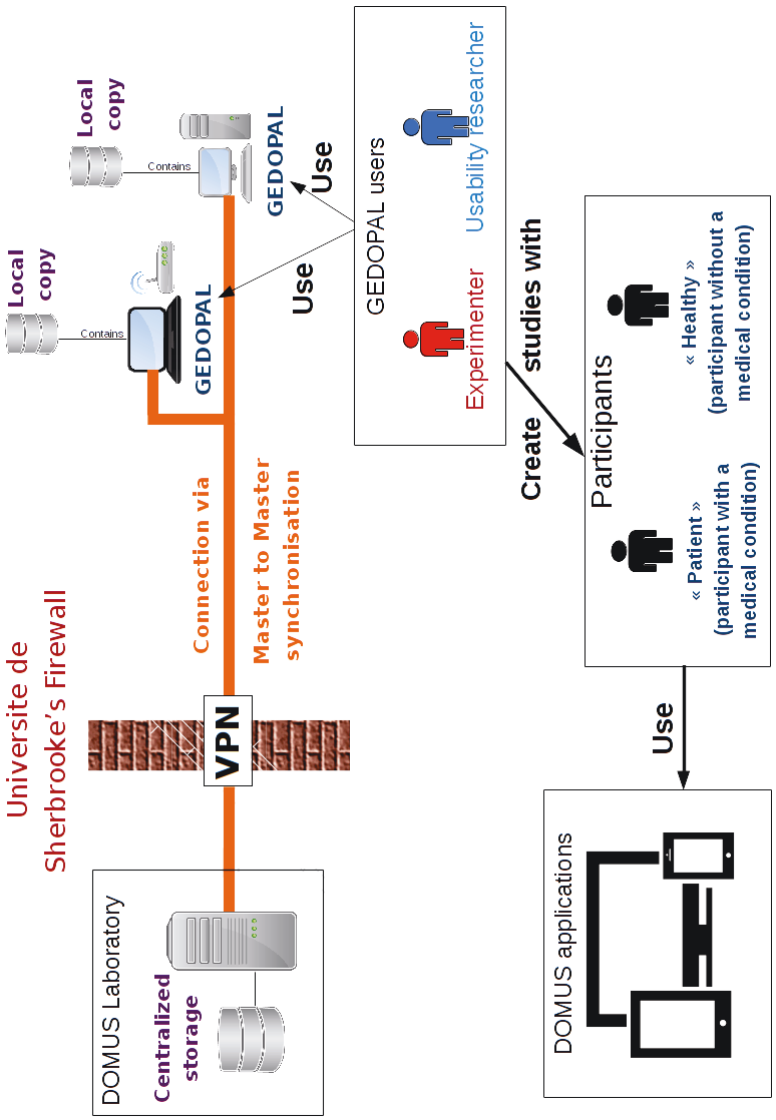


Fig. 1. An overview of GEDOPAL.

4.1. Storage and access

4.1.1. Storage. Because most of the data is very structured, we decided to use a DBMS (Data Base Management System). A database management system (DBMS) using SQL (Structured Query Language, [Groff and Weinberg 1999]) has many advantages [Ramakrishnan and Gehrke 2003]:

— Sharing data is very easy, so GEDOPAL can easily access data with minimum maintenance. Such access makes it possible to be aware of changes very quickly.
— The more users access the data, the greater the risks of data security breaches. A DBMS ensures some data privacy and security policies meaning that GEDOPAL can use them without implementing new ones.
— If the database is properly designed, the risk of data inconsistency is greatly reduced. This minimizes the amount of verification GEDOPAL has to perform on the data.
— It improves the access to specific data as well as the retrieval of specific information. The use of query to select the information asked by a researcher renders it very simple.
— It provides a solution for backup and recovery. In case of problems, GEDOPAL only has to ensure that the DBMS holds the data to be saved.

Moreover, DBMS is quite simple to develop and maintain. The SQL standard (Structured Query Language) we decided to use is very easy to integrate to GEDOPAL.

The only actual disadvantages encountered is that if the DBMS fails for some reason, all of the architecture fails with it.

4.1.2. Access. To ensure that the data would be accessible even in case of failure of the centralized storage or another component (such as an Internet connection), we could only think of one solution for the storage facility: use a centralized storage with local copies. All the clients would connect to the centralized storeage to update the information of their local copies. The peer-to-peer decentralized storage solution was considered but discarded because of the small number of connected peers. Indeed, in the peer-to-peer architecture, there can be quite a delay between the modification of an information and the moment the changes are passed to all. During this delay, another peer may have changed the same information before its update. This recurrent problem in peer-to-peer architecture becomes more serious as the number of peer diminishes. The small number of users, and the little time of connection of each decided us against peer-to-peer.

To maintain the information up to date on the local copy, we chose a master to master communication. Data modified by an user are also modified on the centralized storage. If another user accesses the data, the centralized storage triggers an alarm that warns that the data has been modified. However, this solution causes a critical problem. If a user makes a mistake, every other user will receive it and it will impossible to recover the correct data. This constraints must be kept in mind and some safeguards have to be implemented to prevent such mistakes. The users must be made aware of the problems. In case of data deletion, no data will be eliminated from the database but just de-activated to be able to recover the data.

4.2. Data model

As shown in figure 2, the database holds several relations. The three main relations concern the users, the participants and the experimentations (respectively called User, Participants and Experimentation). They are only used to store the data that can be easily stored into databases. To connect these three relations other relations were designed such as the Access_experimentation and Participant_experimentation.

The connections look odd at first because there are two representation of participants in the database. The relation Participants holds the general information of a participant (such as name or age). The relation Participant_experimentation holds the information on a participant with respect to an experimentation. This representa-

tion of participants is different from the first one for two reasons: information specific to this experimentation are added (group or test results for example) and other information are removed (such as irrelevant diagnosis). It allows for experimenters to keep irrelevant information for a specific experimentation secret. This feature is often used for example if a participant has already been in a experimentation. When using GEDOPAL, an experimenter first has to enter participants in the first representation. Then for each experimentation, the user add the participants to the experimentation (which allows users to add or remove information as they wish). The relations Participants and Participant_experimentation are connected using external and primary keys.

The relation Access_experimentation is connected through the same system of keys to the relations Users and Experimentations. This allow to get for each user, the access right they posses on a specific experimentation. The access right are based on the group described in section 4.3.1.

There are a few other relations to store information such as log or list of existing application. They are easily connected to the rest of the relations through the same system of primary keys based on external keys.

4.3. Confidentiality and anonymity

Specifications (section 3.1.2) lead us to create several categories of researchers. According to the category a user belongs to, they have access to different information. We first present the various user groups designed for access control and then the mechanisms used to ensure anonymity.

4.3.1. Groups. We decided to implement a Role Based Access Control. Each researcher is granted a role for each application and experimentation. Depending on the role, specific data can or cannot be accessed. The difference between experimenter and others is refined.

We differentiate five roles in a experimentation:

(1) ***External*** : This role can only **access** the **de-identified** experimental data on the experimentations the user participate in.
(2) ***Usability researcher*** : This role can only **access** and **modify** the **de-identified** experimental data on the experimentations the user participate in.
(3) ***Laboratory directors*** : This role does not have access to any personal data. However it can **access** and **modify every experimental information** on every experimentation for every application. Since roles are granted for each experimentation, this role corresponds to giving a ”Usability researcher” role to the user for all the experimentations. This role exists mainly for maintenance purposes for researchers who cannot access the confidential information.
(4) ***Experimenter*** : This role has access to the **personal information** on their patients but they can only **access** and **modify** all of the information for the experimentations they participate in. For example, if a patient of the user is in another study (called experimentation B) the user does not have access to, the user has no rights to access the confidential information pertaining to this patient on experimentation B.
(5) ***Administrator*** : This role can **access** and **modify everything** (experimental data and others). This role, that exists mainly for maintenance purposes, has to be used very carefully. To avoid ethical problem, the principle of least privilege must apply. This is why this role is not embodied by a person but can only be accessed after obtaining its password through due process.

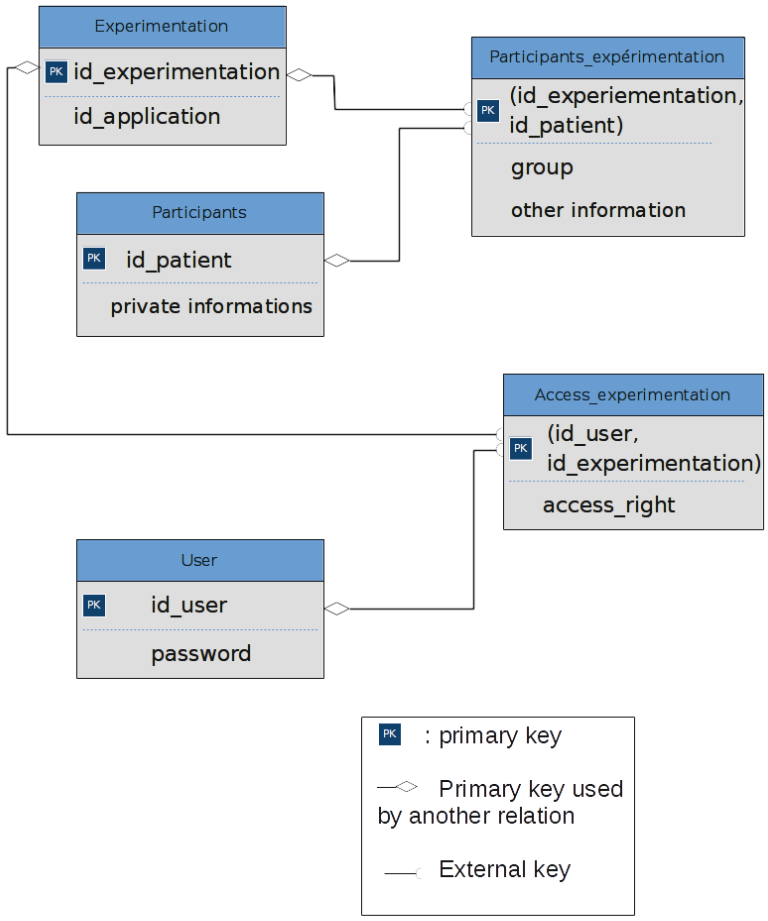


Fig. 2. A simplified representation of how relations are held and connected to each other in the database.

Two figures summarize the roles. The first one (figure 3) represents the rights for one experimentation. The administrator can do anything exactly like the experimenters when there is only one experimentation. For only one experimentation, the laboratory directors and usability researchers roles have the same rights (access and modify only de-identified information). Finally, external researchers can only access de-identified information.

When we consider several experimentations, the figure 4 is more accurate as to user's access right. The difference is that the administrator has the right of clinical researcher for every experimentations and laboratory directors have the right of usability researchers for every experimentation.

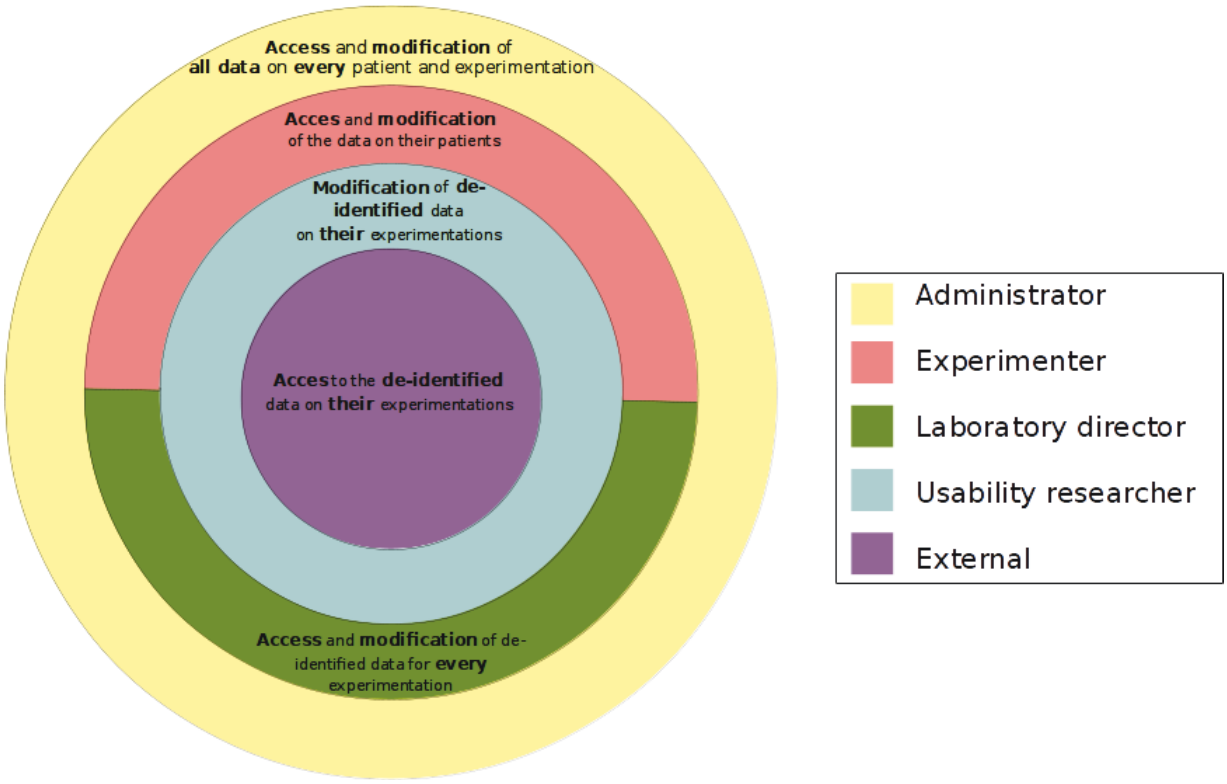


Fig. 3. A representation of the role's rights within one experimentation.

The de-identification of the data is made by the software during access. The software will only display the information one has access according to one's role. Meaning that the request to the database only asks for some of the columns, selected according to the rights registered in the database. This way, when experimenters save some data, they do not have to worry about de-identification because the software will do it for them.

4.3.2. Anonymity. Anonymity must be ensured on the servers and on the local copy of every user. This is why we decided to encrypt the sensitive data in the database such as the patient data (see section 2.2) and researchers information. This way, if an unauthorized person successfully accesses the local copy of the database, this person will only see encrypted data. A key is required to decrypt data as well as the technique used to encrypt the data. This way, ideally, the data will only be readable through the application. GEDOPAL.

5. GEDOPAL IMPLEMENTATION

From this conception, an implementation was made. During implementation, there were some technical problems to solve. Once the software was implemented, it was rapidly tested with some users. This section presents the technical and architectural considerations encountered and the results obtained during the pre-test.

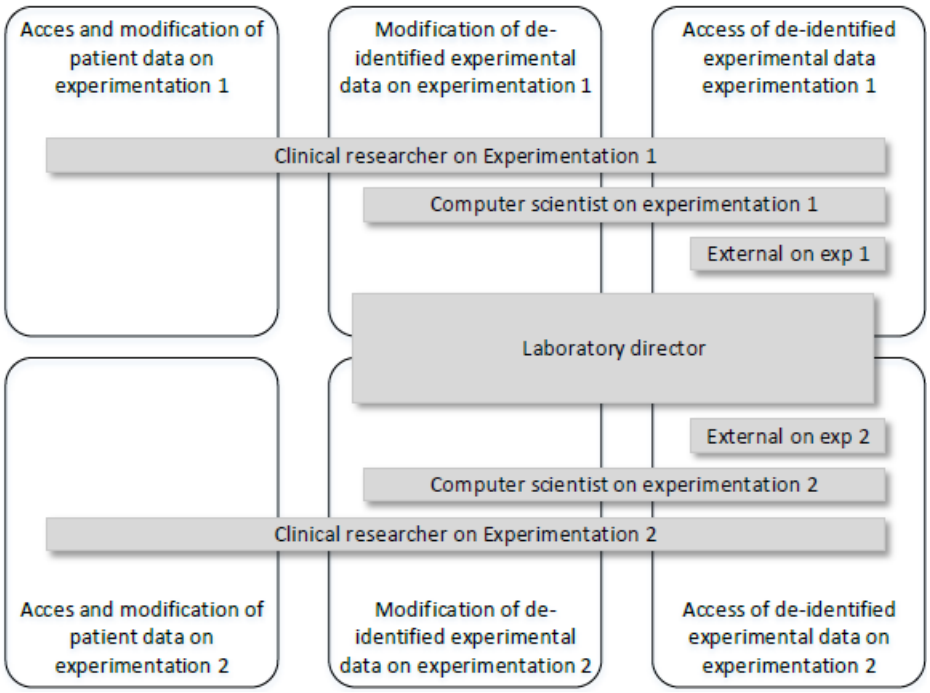


Fig. 4. A representation of the role's right with two experimentations.

5.1. Architectural choices

There were different solutions for a client-server application like the one described above. The first choice was whether or not to use an Internet rich client. Because of the portability problem (different navigators to support) and the maintainability of JavaScript code, this solution was abandoned. The turn-over of students and research assistants, mentioned above would have made this solution very unstable and not as robust as needed. It was then decided that Rich Client Platform (or RCP) would be a good trade off between all of the constraints.

The constraints detailed in section 3.1.3 specify that tools for analysis should be integrated if possible. The choice of RCP allows these constraints to be partly fulfilled, because of the easy way plugins can be added to the software (with restrictions as to the technology used to develop those tools and whether or not the code is available).

The main advantage of the RCP is the possibility to add plugins in the future. A plugin could be developed specifically for an application or a type of research, and it could easily be installed on some of the users' version with no impact on the others.

5.2. Technical choices

Technical problems have lead to technical solutions.

5.2.1. SGBD choice. We decided to use PostgreSQL [http://www.postgresql.org/ 2015] for two main reasons: it is the most complete free SGBD today and it is already in use at the DOMUS laboratory. Moreover, this open source SGBD is widely used and possesses a complete graphical interface. Other solutions such as MSSQL or MySQL

were considered, but the first one is expensive and the second one is less developed and less efficient than PostgreSQL.

5.2.2. RCP choice. There are a few Rich Client Platforms available but development in JAVA restrains the choice between Eclipse RCP and Netbeans RCP. We chose Eclipse RCP as it was more documented and tutorials were available on the Web.

5.2.3. MiddleWare choice. The Master to Master connection is made through SymmetricDS by JumpMind, Inc. [1] We have chosen an open source software that implements Master to Master and works on several platform including Windows, Ubuntu and iOS.

5.2.4. Access control and privacy.

Role based access control. As explained in section 4.3.1, a role-based identification has been implemented. Because the connection passes through the VPN of the Université de Sherbrooke (*cf* section 3.2.5), the authentication part is not central. The identification only requires to assign rights to each user. Each user possesses a user name and a password for GEDOPAL. Once entered, if the two match, access rights for every experimentation are retrieved from the database and sent to the software.

Privacy. As explained in section 4.3.2, sensitive information is encrypted. We decided to use the PostgreSQl extension pgcrypto. This extension is easy to install and to use because the functions are supplied in PostgreSQL. Many techniques of encryption and hashes are available through this extension.

5.2.5. User experience. After the implementation, GEDOPAL was installed to be tested by several researchers. We first present the procees used to install GEDOPAL (and the problems encoutered during the installation) followed by a simple description of the interface.

Installation. The installation of GEDOPAL is not easy as it requires the installation of several distinct pieces of software (PostgreSQl, SymmetricDS, and GEDOPAL itself). In addition, there is an amount of configuration to do on the DOMUS servers in order to add a new node to the Master-to-Master network. The installation on the researchers' computers must be performed by technical specialists of the research center. However, GEDOPAL could not be installed on a computer connected to the Health network of Quebec. The access permissions are so strict that we propose to the clinicians to use their own computers.

The installation is followed by a quick tutorial (to emphasize the "danger" of master-to-master synchronization among other things, cf 4.1.2). It takes preferably the form of a presentation but, if impossible, a written tutorial exists.

Use. The interface was designed to be easy to learn and to remember. As the technology used is Eclipse RCP, the interface is similar to the one used by Eclipse.

A print screen of the application is presented in figure 5.

Each interface is build the same way: on the left side, a tree allows the user to move through GEDOPAL and on the right the interface allows to interact with GEDOPAL.

On the tree, the first three items give access to the list of applications the user can interact with, the list of participants (if the user has patients meaning the user is either experimenter or administrator), and the list of participants that have accepted to be contacted again for other experimentations. Then comes a branch called "add elements" that contains the items to add participants or experimentations. Finally, a

[1] [http://www.symmetricds.org/ 2015]

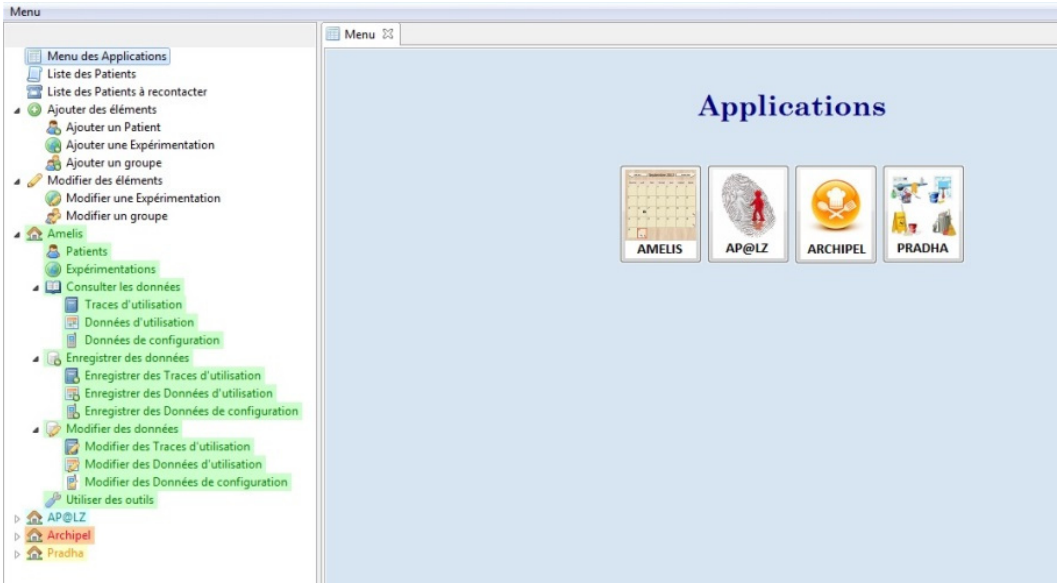


Fig. 5. A print screen of the home page of GEDOPAL.

branch is added for each application the user can access. Depending on the user rights, the application branch allows access to the experimentations and their participants. Three other branches allow to add, consult or modify information. Finally an item allows to use external tools on the data.

Throughout the interface, a color code was designed. One color for each application and a color for outside any application (meaning when no application is selected). This color code helps navigating in GEDOPAL and distinguishing between several applications as the interfaces look the same except colors and name of applications tested.

We pay attention to offer a rapid access to functions thanks to simple icons that are consistent through GEDOPAL, such as a green plus represent the action to add, a red minus the action to remove.

6. TESTS

GEDOPAL has been installed on several researchers' computers. Tests have been conducted, one with a clinical researcher, one with a usability researcher. A simple scenario has been created. The testers had to create an experimentation and some patients and assigned them some arbitrary results into GEDOPAL. The goal of this simulated experimentation was to evaluate the impact of a technology on the life on some elderly patients by comparing with a control group. The tasks were:

— Create an experimentation: define all of the parameters described (scope, groups, location...) section 2.2.
— Create the two groups: control group and actual group.
— Create patients: define all of the parameters described (name, age, group,...) in section 2.2.
— Add patients to the experimentation.
— Enter some imaginary results for the patients.
— Request some of the results of the patients to perform analysis.

In both cases, the researcher holds the role they would have had in an actual experimentation. The goal of the tests were to verify that GEDOPAL is indeed easy to learn and remember and also that it fullfil the needs of the researchers without giving them much extra work.

6.1. Clinical researcher

Installation. The installation was too complex to be made by the researcher whose field was occupational therapy. The 5 minutes tutorial was followed by few questions before the test itself began.

Use. The clinical researcher found the interface quite pleasant and easy to use. They completed the scenario quite easily after attending the tutorial. The first thing the scenario ask was to create an experimentation and add users to it. To do that, the user first has to add a participant then to create an experimentation and finally to add a user to this experimentation. They found this process quite complex and burdensome. They also found that not having all the information on a participant is disturbing. The color code was not noticed. This is probably because our scenario included only one application and the colors were not helpful. Even if the clinical researcher agreed to the advantages of GEDOPAL for sharing information with colleagues, they decided to keep using their usual paper sheets to store and analyze the data and to use GEDOPAL to share data. The main reason for the decision was that GEDOPAL does not allow for easy annotation of participant data. There are many information on participants that do not feet the standard we use to descrie participants in the database. Those information have to be entered into a free text case and as such are not easy to search through. The lack of flexibility of the format of the data was too repellent.

6.2. Usability researcher

Installation. GEDOPAL was installed by the researcher whose field is computer science. Due to minor problems, we change the installation tutorial to be more clear and precise on the process. This time, the written tutorial on how to use GEDOPAL was used instead of an oral presentation because the computer skills were not an issue.

Use. The computer scientist understood very well the way GEDOPAL works. They appreciated how easy and fast the de-identified information are available. They completed the scenario very quickly and found the interface quite simple (even too simple at times). They regretted the few features to search through data, expressing the will to have more criteria available. The type of data were also judged too limited, meaning not enough flexibility in the format of the data was present, exactly like the clinical researcher. Though they find the utilization quite simple, the color code was again not noticed, so not used. However, because the interface was so simple, they probably don't need this color code.

6.3. Discussion

Positive elements. Both researchers found the interface very easy to use and almost no problems were encountered to complete the scenario. They found GEDOPAL helps saving a lot of time and resolving many difficulties while protecting the participant's privacy.

Negative elements. Both researchers found that GEDOPAL is not flexible enough. For the clinical researcher, the test results were not flexible enough, the number of tests should be variable. For an experimentation all the participants must have the same number of cognitive tests. The researcher would have like to be able to change this number with every participants.

For the usability researcher, there was not enough flexibility in the type of data to be entered and to search through. To create and add a participant, several field must be filled out. Those fields have a defined number and type (name is a String, age is a Integer), except for a free text field used to add any other relevant information on this participant. The usability researcher would like to be able to add fields and to assigned them a type to be able to search on those fields instead of having to use the free text field every time.

Appropriation. In both cases, the researchers started to seize GEDOPAL to achieve their goal. Though the utilization is not exactly what we thought it would be (the clinical researcher only used it to share data and not during analysis for example), this is a very good sign. For the clinical researcher, we fear that the extra work of maintaining up to date the information on both a paper sheet and GEDOPAL could be repelling.

7. CONCLUSION

GEDOPAL fulfills an important need in a laboratory where the usability studies are necessary to improve prototypes. These studies imply complex functionalities to store and to share the data produced. GEDOPAL was designed to be easy to learn, to remember and to maintain because of the turn-over in research laboratories. We particularly focused on privacy and anonymity issues because laws and ethic committee require it. Tests have shown that GEDOPAL is easy to use and fulfills some of the needs of researchers, even though some limitations were encountered.

The first limitation was with the type of data taken into consideration. Not considered during conception, the data from smart environments (such as sensors response), were asked by researchers to be added to GEDOPAL. Because of the architecture used by Eclipse RCP, this problem can easily be fixed by developing a plug-in for researchers who need this functionality. Some of the data produced during the studies are not de-identifiable by GEDOPAL. For example, if the application tested is an agenda, some of the data will hold information such as names and addresses of the people the participants met. This information must be kept secret (to respect the privacy). However, such data are very application dependent and GEDOPAL cannot implement a general mechanism to de-identify them. Therefore GEDOPAL can only store such data and only experimenters can access it. A plug-in for each experimentation would have to be developed to de-identify these data so they can be used by usability researchers. Finally, existing tools to perform usability studies should be integrated into GEDOPAL. These tools do not necessarily come in the same programming language, so it may not be possible to fully integrate them. Once again, some plug-ins would have to be developed to fulfill that need. GEDOPAL is then a first attempt to satisfy various researchers involved in usability testing. The challenge that interdisciplinary introduces in this particular prototype testing has been partially resolved by GEDOPAL. The skeleton is well designed and needed further developments to integrate various data and to allow access to the data of a specific participant easily.

REFERENCES

Alan Dix, Janet Finley, Gregory Abowd, and Russell Beale. 1998. *Human-computer Interaction (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Joseph S. Dumas and Janice C. Redish. 1999. *A Practical Guide to Usability Testing*. Intellect Ltd, Exeter, England.

James R. Groff and P.N. Weinberg. 1999. *SQL, the Complete Reference*. Osborne/McGraw-Hill.

Steven Heim. 2007. *The Resonant Interface: HCI Foundations for Interaction Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Marion A. Hersh. 2010. The design and evaluation of assistive technology products and devices Part 2: Evaluation of assistive products. In *International Encyclopedia of Rehabilitation [online]*, M. Blouin and S. Stone (Eds.). CIRRIE. http://cirrie.buffalo.edu/encyclopedia/en/article/311

David M. Hilbert and David F. Redmiles. 2000. Extracting Usability Information from User Interface Events. *ACM Comput. Surv.* 32, 4 (Dec. 2000), 384–421. DOI:http://dx.doi.org/10.1145/371578.371593

http://www.cisco.com/. 2015. AnyConnect, Cisco AnyConnect Secure Mobility Client. (2015). Retrieved April 17, 2016 from http://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html

http://www.postgresql.org/. 2015. PostgreSQL, The world's most advanced open source database. (2015).

http://www.symmetricds.org/. 2015. SymmetricDS, An open source database replication software by JumpMind Inc. (2015).

Hlne Imbeault, Nathalie Bier, Hlne Pigot, Lise Gagnon, Nicolas Marcotte, Tamas Fulop, and Sylvain Giroux. 2010. Development of a personalized electronic organizer for persons with Alzheimers disease: the AP@lz. *Gerontechnology* 9, 2 (2010). http://gerontechnology.info/index.php/journal/article/view/gt.2010.09.02.168.00

Hlne Imbeault, Nathalie Bier, Hlne Pigot, Lise Gagnon, Nicolas Marcotte, Tamas Fulop, and Sylvain Giroux. 2014. Electronic organiser and Alzheimer's disease: Fact or fiction? *Neuropsychological Rehabilitation* 24, 1 (2014), 71–100. DOI:http://dx.doi.org/10.1080/09602011.2013.858641 PMID: 24359438.

Hlne Imbeault, Hlne Pigot, Nathalie Bier, Lise Gagnon, Nicolas Marcotte, Sylvain Giroux, and Tamas Flp. 2011. Interdisciplinary Design of an Electronic Organizer for Persons with Alzheimers Disease. In *Toward Useful Services for Elderly and People with Disabilities*, Bessam Abdulrazak, Sylvain Giroux, Bruno Bouchard, Hlne Pigot, and Mounir Mokhtari (Eds.). Lecture Notes in Computer Science, Vol. 6719. Springer Berlin Heidelberg, 137–144. DOI:http://dx.doi.org/10.1007/978-3-642-21535-3_18

ISO 9241 1998. ISO 9241 : Exigences ergonomiques pour travail de bureau avec terminaux à écrans de visualisation . (1998). http://www.iso.org/iso/fr/catalogue_detail.htm?csnumber=16883

Melody Y. Ivory and Marti A Hearst. 2001. The State of the Art in Automating Usability Evaluation of User Interfaces. *ACM Comput. Surv.* 33, 4 (Dec. 2001), 470–516. DOI:http://dx.doi.org/10.1145/503112.503114

J. Nielsen. 1993. *Usability Engineering*. Academic Press, Boston, MA.

Jacob Nielsen and Rolf Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM New York, NY, USA, Seattle, Washington, United States, 249–256.

Janne Pitkänen, Matti Pitkäranta, and Marko Nieminen. 2012. Usability Testing in Real Context of Use: The User-triggered Usability Testing. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordiCHI '12)*. ACM, Copenhagen, Denmark, 797–798. DOI:http://dx.doi.org/10.1145/2399016.2399153

Raghu Ramakrishnan and Johannes Gehrke. 2003. *Database Management Systems* (3 ed.). McGraw-Hill, Inc., New York, NY, USA.

Chapitre 4

Analyse formelle

Ce chapitre a pour but de répondre à l'objectif 2 (Chapitre 2) : utiliser les méthodes d'analyse formelle pour analyser une application. L'application AP@LZ du laboratoire DOMUS nous servira de cas d'étude, mais la méthode proposée peut s'appliquer à n'importe quelle application. Le but est d'analyser plus en détail l'interface de AP@LZ pour améliorer son utilisabilité. Cela nous aide aussi à affiner notre compréhension de l'interface pour aider à une meilleure mise en place de notre objectif 3 (analyse automatique des traces d'utilisation, voir Chapitre 3). Le plan du chapitre 4 suit la méthodologie utilisée :

- Identifier la liste des tâches réalisables, comprenant toutes les fonctionnalités que notre application offre à l'utilisateur). Cette liste des tâches est indispensable pour la mise en place de l'objectif 3, car elle fournit la liste des classes possibles de notre classification.
- Examiner chaque tâche pour détailler les différents moyens offerts par l'application de l'effectuer. Cette étude facilite l'analyse des opérations effectuées par les utilisateurs à partir des traces d'utilisation uniquement.
- Modéliser l'application AP@LZ à l'aide des méthodes d'analyse formelle de GOMS. Cela nous permettra à la fois de vérifier la validité de l'analyse en la comparant avec des résultats d'utilisation réelle et de mieux comprendre et analyser les erreurs détectées durant l'analyse automatique des traces d'utili-

sation.

1 L'application AP@LZ

L'application AP@LZ est disponible sur téléphone intelligent Android et a pour but de fournir un assistant personnel pour les personnes atteintes de la maladie d'Alzheimer afin de réaliser les tâches d'organisation de leur emploi du temps et pour se remémorer les informations les plus usuelles [24, 25]. Sur la page d'accueil de l'application AP@LZ apparaissent les activités de la journée, voir Figure 4.1. Les six boutons en bas de l'écran permettent d'accéder aux fonctionnalités de l'application. Ces dernières sont : consulter les informations personnelles et les photos, appeler des contacts, accéder à un bloc-note simplifié, gérer des informations médicales (liste des intervenants médicaux, des médicaments et historique médical) et enfin ajouter, supprimer ou modifier des activités à l'agenda. Les flèches à gauche et à droite en dessous du cadre bleu de la date permettent de naviguer dans les jours passés ou futurs pour consulter les activités effectuées ou prévues. L'écran d'accueil (à la date du jour) est le point principal de l'application. Celui sur lequel l'application s'ouvre, mais aussi celui sur lequel l'application revient si plus de 10 minutes passent sans interaction avec le téléphone. Si l'on navigue dans les jours passés ou futurs, il suffit de cliquer sur le bandeau de date bleu foncé pour revenir à la date d'aujourd'hui. Dans les écrans des autres fonctionnalités, un bouton de retour se trouve toujours au même endroit, en bas à droite. Ce bouton est de la même couleur que le bandeau de date et représente une flèche. Quel que soit l'écran sur lequel on se trouve, ce bouton annule l'action en cours et retourne à la page d'accueil (voir Figure 4.2).

Une liste d'activités prédéfinies facilite l'entrée de données. En effet, il suffit d'entrer une fois dans l'application le nom et le lieu des activités que l'on souhaite effectuer et elles seront ajoutées à cette liste. Il est ensuite très facile d'ajouter une activité de cette liste à l'agenda (c'est-à-dire lui donner une date et un horaire précis pour que l'agenda puisse le rappeler). Il est tout aussi facile d'ajouter, modifier ou enlever des activités de cette liste.

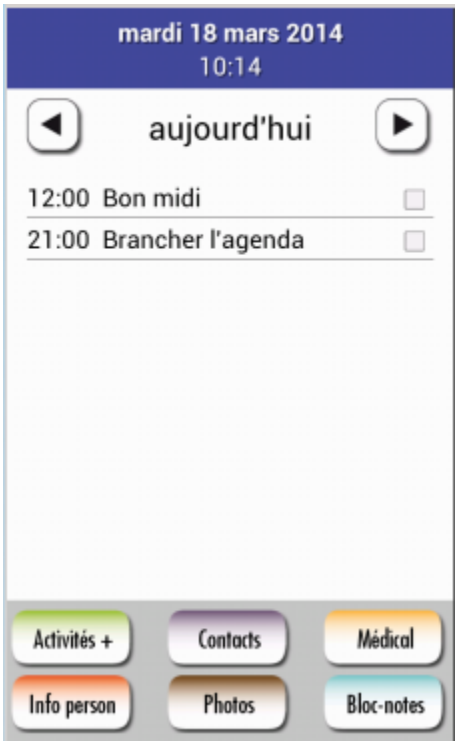


Figure 4.1 – Page d'accueil de l'application AP@LZ

Cette application offre donc différentes fonctionnalités pour effectuer des tâches que nous avons d'abord listées de manière exhaustive avant de les analyser plus en détail grâce à des méthodes d'analyse formelle.

Nous différencierons ici entre tâches et activités même si dans le langage commun, ces mots sont très proches. Tâche désignera ici les actions effectuées avec l'application tandis qu' activité désignera seulement les activités de l'agenda (c'est-à-dire les rendez-vous et les rappels qui sont entrés dans l'application).

1.1 Visualisation des interfaces

Avant d'effectuer une modélisation plus formelle, nous avons préparé une liste de toutes les tâches qui peuvent être effectuées par AP@LZ. Cette liste est triée selon les sept catégories, six pour les fonctionnalités principales qui correspondent aux six

Figure 4.2 – Bouton retour de l'application AP@LZ

boutons de l'écran d'accueil et une pour la navigation dans les jours passés ou futurs à l'aide des flèches. Ces tâches représentent toutes les possibilités qu'offre l'application à l'utilisateur. La liste de ces tâches se trouve en Annexe A.

Une liste des CIG (Composants d'Interface Graphique) présents dans AP@LZ a été compilée pour décider d'un vocabulaire commun et aussi pour commencer l'analyse GOMS (Goals, Operators, Methods, and Selection rules) que nous verrons en Section 2. Pour chaque écran, une liste de tous les CIG accessibles est faite (cf Annexe A). Le vocabulaire utilisé est celui du code de l'application, car c'est celui qui figurera dans les traces d'utilisation. Le type de chaque CIG est indiqué. Les boutons sont faciles à interpréter, mais les *ListView* et *GridView* sont plus compliquées, car sous ce nom se trouvent plusieurs types d'action. Une *ListView* correspond tout simplement à une liste telle que la liste des activités prévues que l'on voit sur l'écran d'accueil (Figure 4.1). Lorsqu'une liste contient des articles de différentes natures (une ligne de texte simple et une ligne de texte cliquable par exemple), cela peut devenir plus compliqué. Par exemple, sous la *ListView knownActivitiesView* de l'écran *KnownActivitiesMainList* (la liste des activités prédéfinies, cf figure 4.3b) se trouve non seulement la liste d'activités prédéfinies, mais aussi les CIG qui permettent d'ajouter et de modifier cette liste d'activités prédéfinies. De la même manière, les *GridView* peuvent être difficiles à interpréter. Les *GridView* sont semblables au ListView, sauf qu'ils présentent les articles disposés dans une grille plutôt que dans une liste.

Le clavier est indiqué sur chacun des écrans où il peut être utilisé. Les lettres tapées ne pouvant pas figurer dans les traces d'utilisation (pour des raisons de respect de la vie privée, comme expliqué au Chapitre 3), il est seulement mentionné clavier.

Il faut maintenant définir les « chemins » à utiliser pour effectuer chacune des tâches définies. Le chemin est la suite de CIG utilisé pour effectuer une tâche considérée.

Pour chaque tâche, le chemin est décrit comme suit :

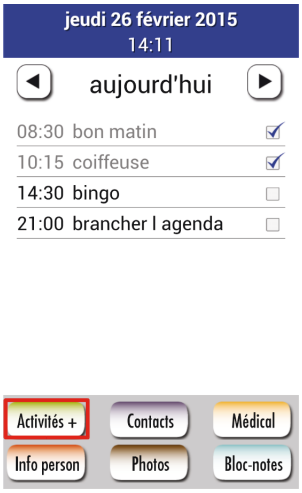
écran où l'on est → CIG utilisé

Lorsqu'une ligne est entre parenthèses, cela signifie que l'interaction peut être répétée plusieurs fois de suite, et donc que l'on ne change pas d'écran.

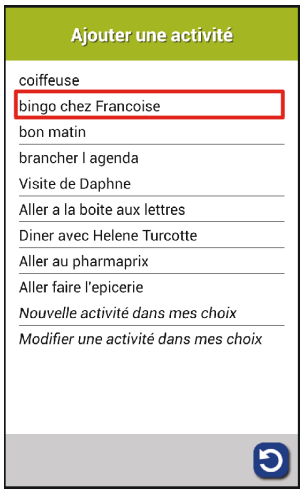
La tâche que nous allons utiliser en exemple dans ce chapitre est : ajouter une activité prédéfinie à l'agenda. La suite de captures d'écran correspondant à la réalisation cette tâche est présentée en Figure 4.3.

Dans ce cas, le chemin est :
ApalzActivity → appointmentButton
KnownActivitiesMainList → knownActivitiesView
(AppointmentActivity → Arrows)
AppointmentActivity → addButton
AppointmentConfirmation → backButton

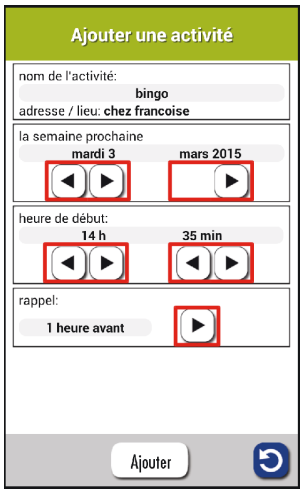
La première ligne correspond à l'écran d'accueil (qui s'appelle ApalzActivity). Le premier CIG à utiliser est *appointmentButton* qui correspond au bouton Activité+. La seconde ligne correspond au clic sur la ligne de l'activité à ajouter. Sur l'écran de la liste des activités prédéfinies (qui s'appelle *KnownActivitiesMainList*), il faut cliquer sur le bon article de la liste (une *ListView* qui s'appelle *konwnActivitiesView*). La troisième ligne correspond au choix de la date, de l'heure et du rappel à l'aide des flèches. Sur l'écran de choix (qui s'appelle *AppointmentActivity*), on peut cliquer un nombre indéterminé de fois sur chacune des flèches pour choisir le bon jour, la bonne date et le bon rappel. Aucune de ces flèches ne fait changer d'écran (d'où les parenthèses). La dernière ligne correspond à l'ajout effectif de l'activité à l'agenda. L'écran est toujours le même et lorsque l'on clique sur le bouton ajouter (qui s'appelle *addButton*), l'activité est ajoutée à l'agenda. Enfin le dernier écran (*AppointmentConfirmation*) ne permet que de cliquer sur le bouton retour pour retourner à l'écran d'accueil.



(a) Tout d'abord, il faut accéder à la liste des activités, donc cliquer sur le bouton Activités+.



(b) Il faut ensuite sélectionner l'activité que l'on souhaite ajouter à l'agenda



(c) Il faut ensuite saisir à l'aide des flèches la date, l'heure et le rappel

(d) Enfin, il faut cliquer sur le bouton ajouter en bas de l'écran

(e) Un dernier écran permet de confirmer que l'activité a bien été ajoutée à l'agenda. Cliquer sur le bouton retour permet de retourner à l'accueil

Figure 4.3 – Suite de captures d'écran représentant le chemin pour ajouter une activité prédéfinie à l'agenda

Cette analyse est effectuée pour chaque tâche. Elle montre qu'un problème récurrent est de faire la différence entre la consultation (des informations personnelles par exemple) et une erreur de l'utilisateur. En effet dans les deux cas, l'utilisateur va utiliser un bouton pour accéder à l'écran, puis cliquer sur le bouton retour pour retourner à l'écran d'accueil. La seule chose qui différencie les deux séquences est le temps passé sur l'écran d'information. Dans certains cas, si le patient ne détecte pas son erreur assez rapidement, il n'est pas possible de faire la différence entre une consultation et une erreur (en utilisant seulement les traces d'utilisation).

Comme il y a beaucoup d'informations contenues dans les différents documents produits, et que le texte n'est pas forcément le format le plus efficient pour l'analyse de ces données, des schémas résumant la plupart des informations ont été compilés.

Les schémas ont été séparés en fonction des fonctionnalités fournies par AP@LZ. Le premier, le plus complexe, correspond à la fonctionnalité principale, celle de gestion des activités et de l'agenda, cf Figure 4.4.

Dans ces diagrammes, les cases représentent les différents écrans et les flèches sont les CIG. Les rectangles gris arrondis sont les tâches définies plus haut. Le diagramme ne contient pas toutes les informations contenues dans les textes, car cela n'était pas forcément possible. Par exemple, pour ajouter une activité à l'agenda et pour modifier une activité déjà inscrite à l'agenda, les chemins sont très proches même si les tâches sont différentes. Il n'est pas possible de les différencier dans le schéma même si la différenciation est faite dans la version texte. Les couleurs utilisées reprennent celles de l'application AP@LZ (vert pour les activités, violet pour les contacts, jaune pour le médical...).

Le diagramme suivant (cf Figure 4.7) correspond aux fonctionnalités permettant de gérer ce qui se rapporte au médical (les médecins, les médicaments et l'historique médical).

Le troisième diagramme (cf Figure 4.6) réunit trois fonctionnalités de AP@LZ, les deux plus simples, celle permettant de consulter ses informations personnelles et celle permettant de consulter ses photos combinées avec celle plus complexe permettant de consulter et d'appeler ses contacts.

Enfin le dernier diagramme (cf Figure 4.5) concerne les fonctionnalités offertes par le bloc-note de AP@LZ.

Ces quatre diagrammes contiennent le rectangle correspondant à l'écran d'accueil et aux flèches permettant de naviguer dans les jours passés ou futurs. Lorsqu'ils sont réunis en une seule figure, nous obtenons la Figure 4.8. Bien qu'assez peu lisible, elle met bien en valeur deux choses :

— L'application AP@LZ est assez simple d'utilisation et offre petit nombre de possibilité sur chaque écran.

— L'écran d'accueil est le point central de l'application, le point de départ de chaque tâche.

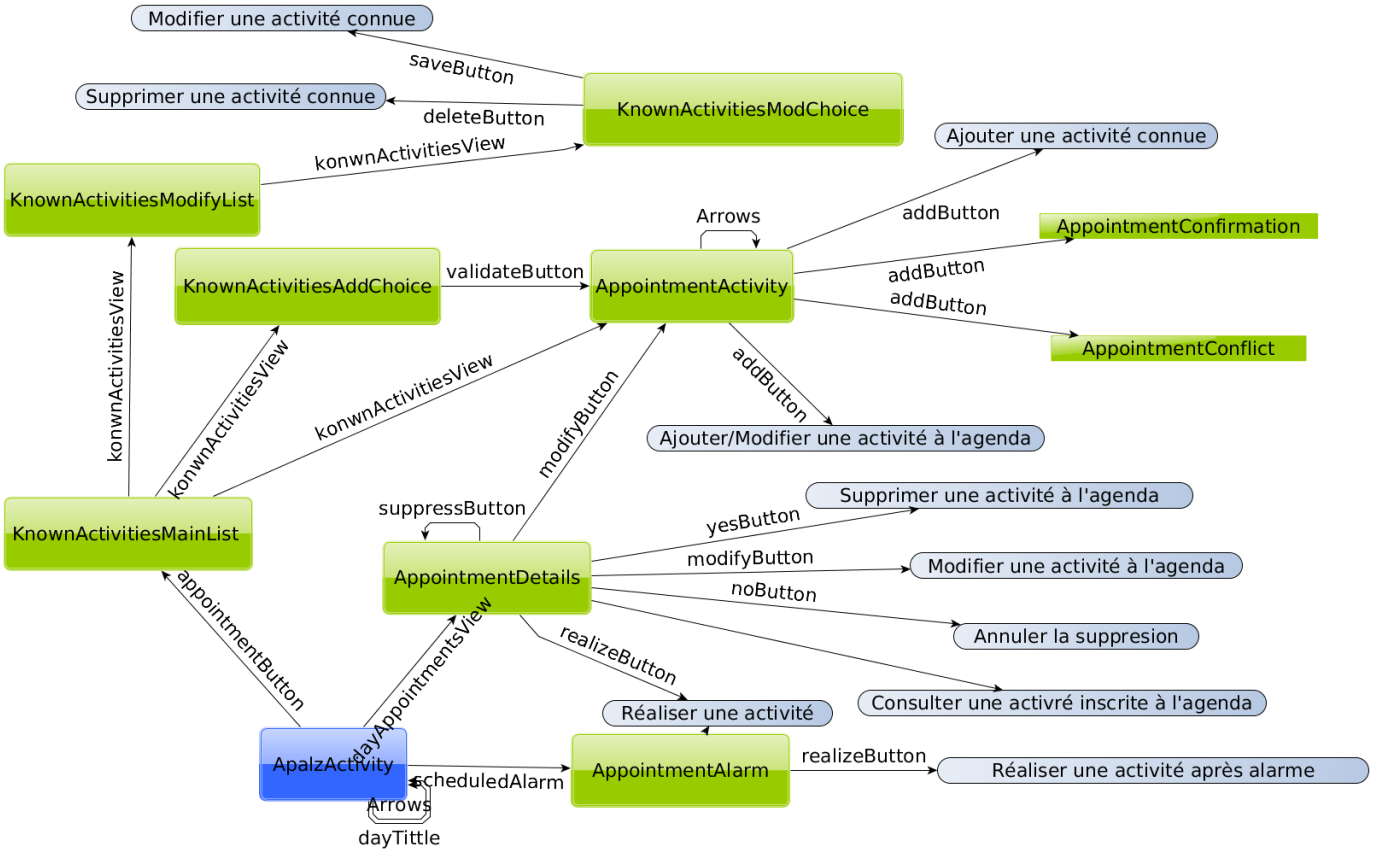


Figure 4.4 – Résumé des chemins permettant d'accéder aux fonctionnalités concernant la gestion des activités et de l'agenda.

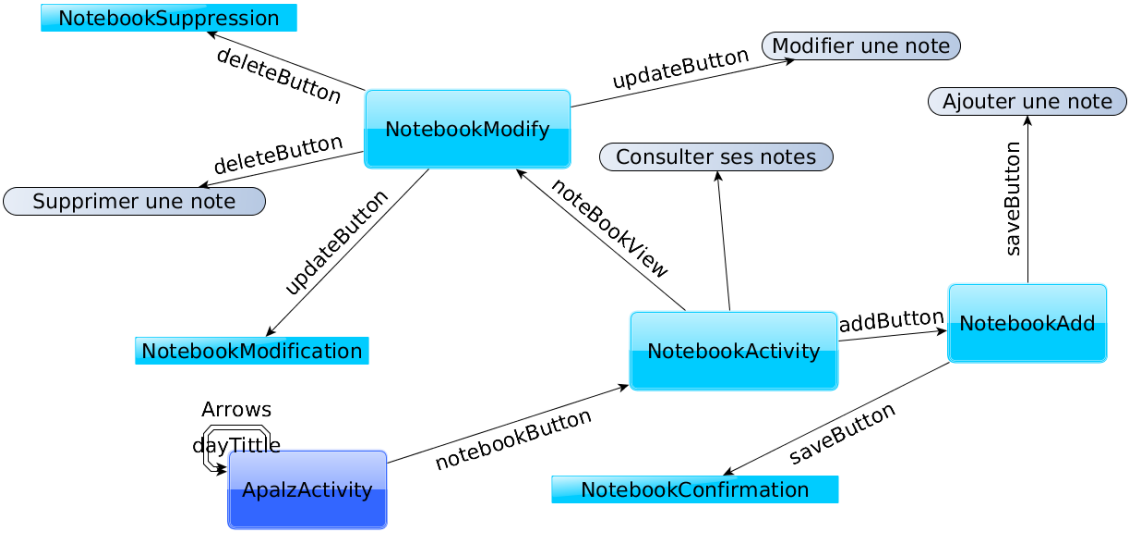


Figure 4.5 – Résumé des chemins permettant d'accéder aux fonctionnalités fournies par le bloc-note.

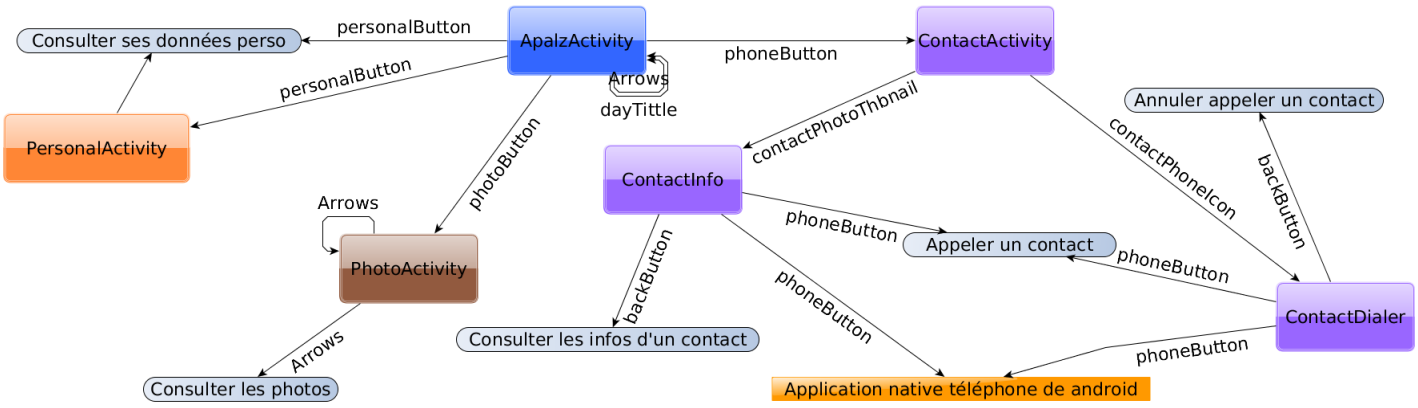


Figure 4.6 – Résumé des chemins permettant d'accéder aux fonctionnalités concernant les informations personnelles, les photos et les contacts.

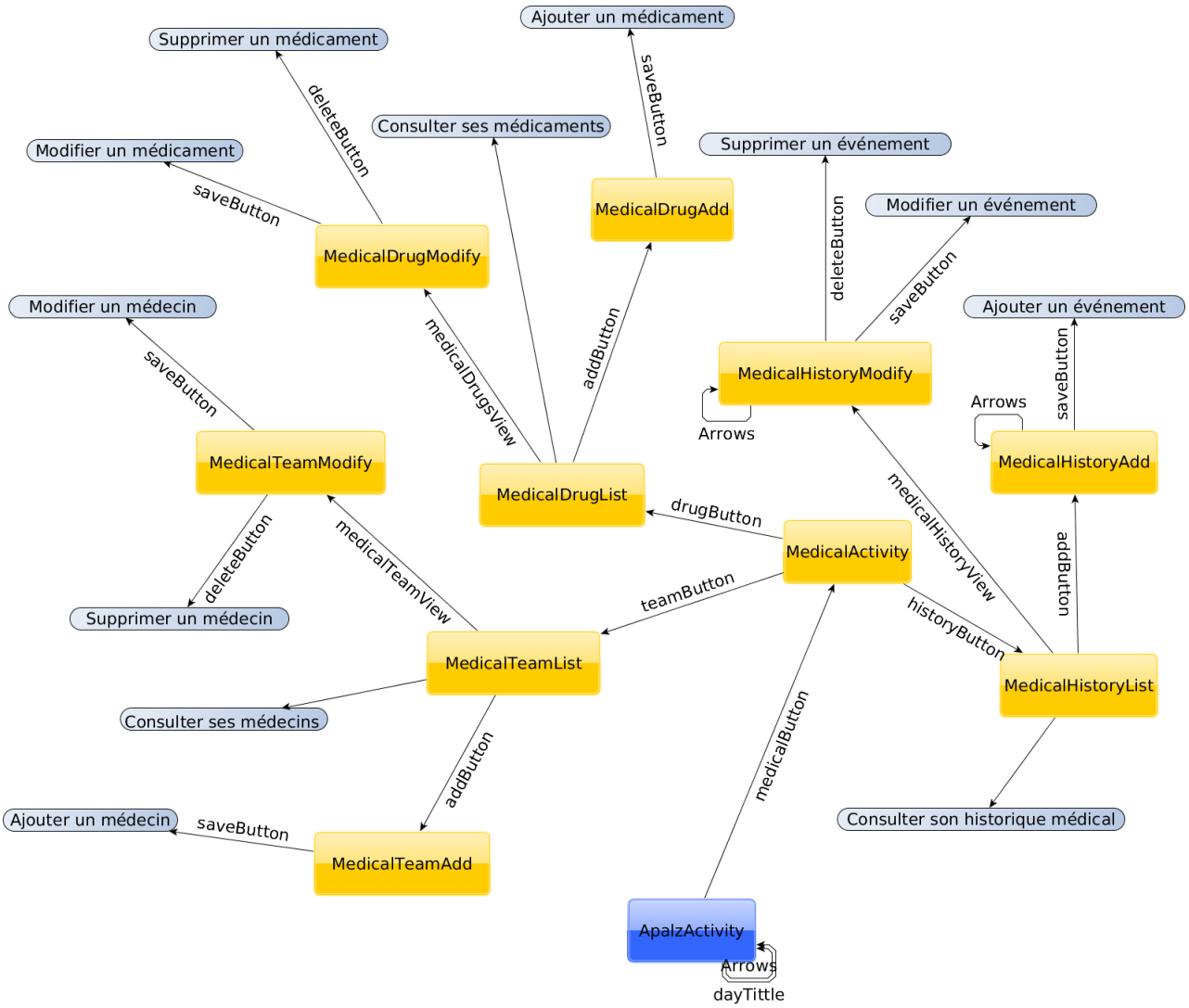


Figure 4.7 – Résumé des chemins permettant d'accéder aux fonctionnalités concernant la gestion des informations médicales.

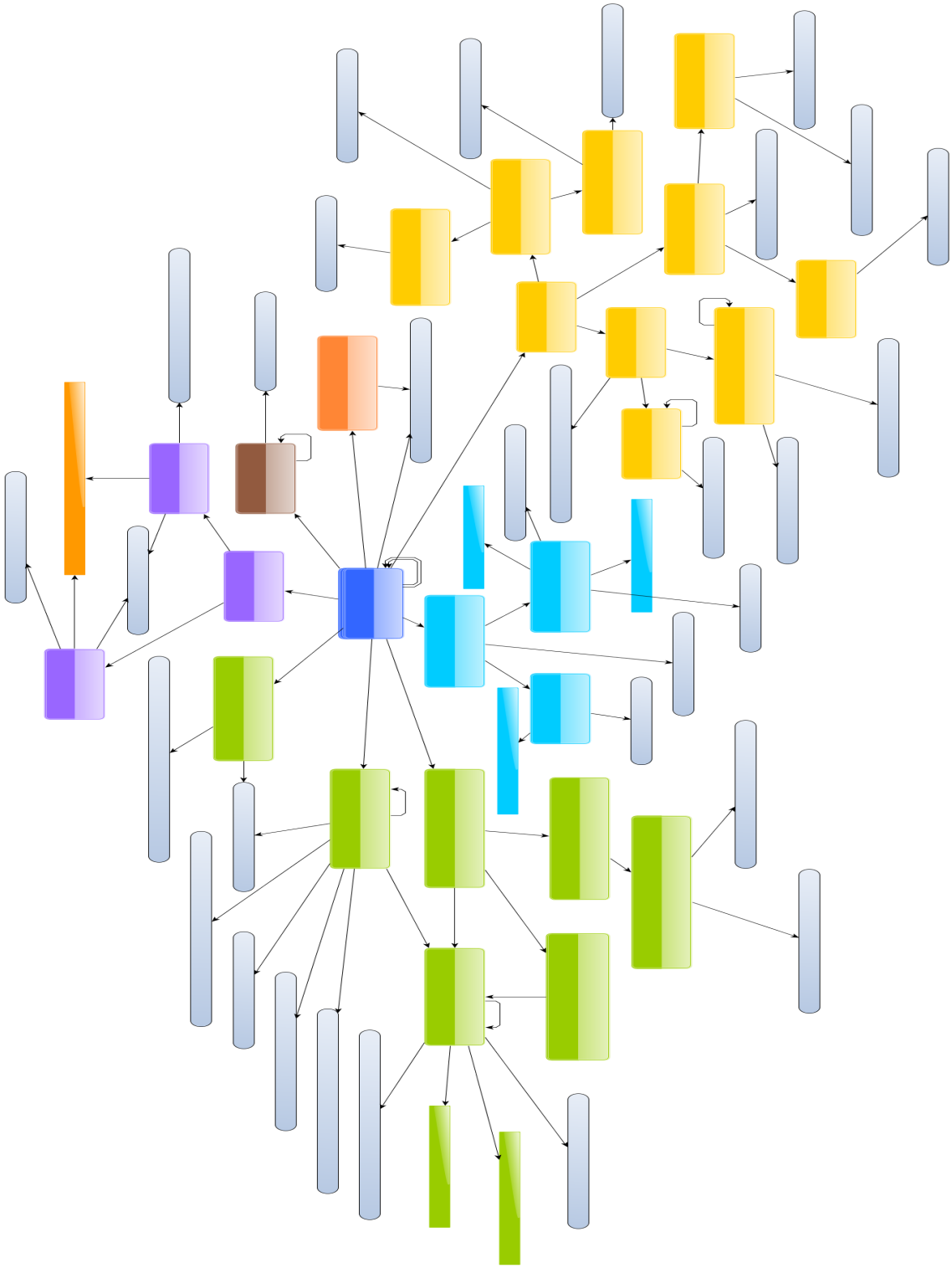


Figure 4.8 – Résumé des chemins possibles dans AP@LZ. Pour plus de lisibilité, les étiquettes ont été enlevées. Tout comme dans les figures précédentes, les cases grises en forme de stade correspondent aux différentes tâches que l'on peut effectuer, les flèches aux CIG utilisables et les cases rectangulaires aux différents écrans de AP@LZ. La couleur de ces dernières dépend de leur utilité. Pour voir plus en détail, se référer aux figures précédentes (cf Figure 4.4 à Figure 4.7).

2 Modélisation avec les méthodes GOMS

Deux méthodes d'analyse formelle présentées par Heim ([18]) ont été appliquées . Toutes deux dérivent des méthodes de GOMS (Goal, Operator, Method, Selection). Nous avons sélectionné ces méthodes pour deux raisons : la première est de valider les travaux présentés précédemment, la seconde est de pouvoir comparer à des résultats expérimentaux (en estimant la différence de temps entre une consultation d'un écran et une erreur. Cette différence pourra être une indication utile lors de l'analyse automatique des traces d'utilisation).

La méthode CMN-GOMS modélise les buts et les actions effectuées par l'utilisateur. La méthode GOMS-Keystroke prédit le temps qu'un utilisateur mettra à effectuer une action. La première méthode explique les erreurs des utilisateurs et prédit le comportement des utilisateurs dans l'interface. La deuxième méthode permet d'obtenir des indicateurs quantitatifs pour comparer le modèle à des résultats d'expérimentation. Si les temps prédits par la méthode Keystroke sont vérifiés par l'observation d'utilisateurs réels, alors le premier modèle sera considéré comme valide.

2.1 CMN-GOMS

Reprenant l'analyse précédente, une modélisation CMN-GOMS est effectuée pour chacune des tâches. Sur chacune des tâches, la suite de buts de l'utilisateur va être modélisée. Dans ce cas, le but signifie l'objectif et la suite de buts correspond aux sous-buts que l'utilisateur va se fixer pour atteindre son objectif final. Par exemple, lorsque quelqu'un a soif, son objectif final sera de boire. Pour atteindre cet objectif, cette personne va se fixer une suite de sous-buts : prendre un verre, le remplir, le boire. Cette suite de sous-buts sera plus simplement appelée suite de buts dans la suite.

Je vais ici expliquer cette méthode sur notre tâche exemple. Comme celle-ci n'a pas changé, la suite d'écran qui lui correspond est la même et est présentée en Figure 4.3.

Pour la tâche : "Ajouter une activité prédéfinie à l'agenda", le chemin expliqué

précédemment est :
ApalzActivity → appointmentButton (correspond au bouton Activité+, figure 4.3a)
KnownActivitiesMainList → konwnActivitiesView (correspond au clic sur la ligne de l'activité à ajouter, figure 4.3b)
(AppointmentActivity → Arrows) (correspond au choix de la date, de l'heure et du rappel à l'aide des flèches, figure 4.3c)
AppointmentActivity → addButton (correspond au bouton ajouter, figure 4.3d)
AppointmentConfirmation → backButton (bouton retour de la fenêtre de confirmation figure 4.3e)

L'analyse GOMS qui lui correspond est :
GOAL : Add-KnownActivtyToAgenda
GOAL : Edit-Unit-Task
GOAL : Goto-ApalzActivity
GOAL : Execute-Unit-Task (check if already on ApalzActivity or touch the GUI backButton)
GOAL : Edit-Unit-Task
GOAL : Goto-KnownActivitiesMainList
GOAL : Execute-Unit-Task (touch the GUI appointmentButton)
GOAL : Edit-Unit-Task
GOAL : Find right activity
GOAL : Execute-Unit-Task (find the right line to touch on the GUI knownActivitiesView)
GOAL : Edit-Unit-Task
GOAL : Goto-AppointmentActivity
GOAL : Execute-Unit-Task (touch the GUI knownActivitiesView)
GOAL : Edit-Unit-Task
GOAL :Choose the right date
GOAL : Execute-Unit-Task (touch the GUI arrows)
GOAL : Edit-Unit-Task
GOAL :Choose the right hour
GOAL : Execute-Unit-Task (touch the GUI arrows)

GOAL : Edit-Unit-Task

GOAL :Choose the right reminder

GOAL : Execute-Unit-Task (touch the GUI arrows)

GOAL : Edit-Unit-Task

GOAL : Add the activity to the agenda

GOAL : Execute-Unit-Task (touch the GUI addButton)

GOAL : Edit-Unit-Task

GOAL : Check the activity is added to the agenda

GOAL : Execute-Unit-Task (touch the GUI backButton)

Verify-Unit-Task

Cette modélisation représente les différents changements de buts que l'utilisateur effectue.

Dans un premier temps, lorsque l'utilisateur décide d'ajouter une activité à l'agenda, il va d'abord chercher à atteindre le bouton Activité+ (qui s'appelle appointmentButton). Pour cela, il lui faudra être sur l'écran d'accueil. Son premier but sera donc d'atteindre l'écran d'accueil. Si l'application est déjà sur l'écran d'accueil, il n'y a rien à faire, l'utilisateur change de but, sinon il utilise le CIG de retour (la flèche bleue foncée qui s'appelle) backButton pour s'y rendre.

Une fois sur l'écran d'accueil, le but change, il faut maintenant aller dans la liste des activités prédéfinies pour pouvoir choisir celle que l'on souhaite ajouter à l'agenda (utiliser le bouton Activité+).

Ensuite, il faut chercher la ligne de l'activité qui correspond à celle que l'on souhaite ajouter à l'agenda. Une fois cette ligne localisée, il faudra cliquer dessus (sur la bonne ligne de la ListView knownActivitiesView).

Enfin, il faut choisir la date, l'heure et le rappel à l'aide des flèches. Pour confirmer, l'utilisateur n'aura plus qu'à cliquer sur le bouton d'ajout (qui s'appelle addButton).

L'utilisateur termine cette tâche en vérifiant que la bonne activité a bien été ajoutée et en cliquant sur le bouton retour pour retourner à l'accueil.

Cette méthode d'analyse formelle modélise les « pensées » de l'utilisateur lorsqu'il

effectue une tâche.

À partir de l'application AP@LZ, nous avons effectué une analyse CMN-GOMS sur toutes les tâches possibles dans cette application. Cette méthode est la première partie du processus qui permet d'évaluer une application à partir des traces d'utilisation. Une fois ce modèle validé, nous serons en effet en mesure de l'utiliser pour la dernière partie de notre processus, l'utilisation de l'apprentissage automatique pour détecter les erreurs d'utilisation et les problèmes d'utilisabilité grâce aux traces d'utilisation.

2.2 GOMS-Keystroke

La méthode GOMS-Keystroke estime le temps pris par l'utilisateur pour effectuer une des tâches. Chaque action de l'utilisateur est découpée en opérateurs de bas niveau. Le temps d'une action est égal à la somme des opérateurs qui la composent. Il existe différents opérateurs dans GOMS-Keystroke tels que le temps de frappe au clavier, le temps de réflexion, le temps de mouvement de la main du clavier à la souris, le temps de mouvement de la souris pour pointer sur le bon élément... Dans notre cas, les seuls opérateurs qui peuvent être utilisés durant l'utilisation d'AP@LZ sont le temps de réflexion (M) et le temps de clic à l'écran (K).

En reprenant l'exemple du paragraphe précédent, pour ajouter une activité à l'agenda, il faut tout d'abord cliquer sur le bouton intitulé Activité+. Cela correspond à un temps de réflexion suivi d'un clic, donc à un M puis un K, écrit de manière formelle : MK.

Le résultat de l'application de la méthode GOMS-Keystroke à l'exemple en entier est présenté dans le tableau 4.1.

Ajouter une activité prédéfinie à l'agenda :

Ce résultat est obtenu en décomposant la tâche en étapes qui correspondent aux lignes du tableau.

Une première étape pour appuyer sur le bouton Activité+.
La deuxième pour choisir la bonne activité dans la liste.
La troisième étape est plus complexe, car il faut ensuite choisir la date, l'heure et le

Tableau 4.1 – Décomposition GOMS-Keystroke de la tâche : "Ajouter une activité prédéfinie à l'agenda". La colonne temps correspond au temps que le modèle prévoie pour effectuer ces tâches.

Chemin	GOMS-Keystroke	Tps
ApalzActivity →appointmentButton	MK	2,65s
KnownActivitiesMainList→konwnActivitiesView	MK	2,65s
(AppointmentActivity → Arrows)	MK[3]K[1] M K[12]K[6] MK	33,95s
AppointmentActivity → addButton	MK	2,65s
AppointementConfirmation →backButton	MK	2,65s
Total	MKMKMK[5]MK[12]K[6]MKMKMK	44,55s

type de rappel.

Pour saisir la date, l'utilisateur utilise en moyenne 4 clics. En effet, 5 clics sont nécessaires pour le jour (les rendez-vous sont rarement pris plus de 10 jours à l'avance) et aucun pour les mois, car il est très rare qu'un rendez-vous soit pris un mois à l'avance.

La saisie de l'heure nécessite en moyenne 6 clics, en effet 12 clics est le maximum, 0 le minimum. De la même façon, pour les minutes 3 clics sont nécessaires, 6 clics maximum et 0 minimum.

Enfin le rappel nécessite 1 clic, le rappel plus courant est de "1 heure avant", et il n'y a que 3 choix.

La quatrième étape correspond à l'ajout effectif de l'activité à l'agenda c'est-à-dire à un clic sur le bouton "Ajouter".

La cinquième et dernière étape consiste en un clic sur le bouton retour de l'écran de confirmation.

Le résultat est donc MK MK MK[5] MK[12]K[6] MK[1] MK MK (cf Tableau 4.1)

La valeur de M que nous avons choisi pour la modélisation est 1,35 seconde, le temps le plus couramment utilisé dans la littérature (cf. réf. [18]). La valeur fixée pour K est la valeur la plus élevée de la littérature (1,3 seconde) pour deux raisons,

la première est que l'application AP@LZ ne permet pas de faire plus d'un clic par seconde pour éviter les doubles-clics involontaires et la seconde est que le clavier de AP@LZ est alphanumérique donc assez peu d'utilisateurs l'utilisent très efficacement.

Pour cet exemple, on trouve 45,85 secondes ($7 \times 1,35 + 28 \times 1,3$).

Ces valeurs doivent maintenant être confirmées par des observations d'utilisateurs.

2.3 Expérimentation

Les observations ont été effectuées avec six utilisateurs sains, qui ont utilisé AP@LZ pendant deux jours en suivant un scénario prédéfini qui leur était communiqué par courriel au fur et à mesure. Le scénario est consultable en Annexe B. Les résultats sont très proches des valeurs attendues avec des personnes non atteintes de la maladie d'Alzheimer à l'exception d'une tâche ("supprimer une médecin") pour laquelle la différence est importante. Voici une présentation des résultats pour certaines des tâches possibles avec AP@LZ.

Tâche (nombre d'exécutions/scénario)	tps observé	tps modélisé	erreur relative
ajouter une activité à la liste (13)	50,3s	45,9s	0,10
supprimer une activité de la liste (13)	10,6s	10,6s	0
modifier une activité (9)	15,7s	17,15s	0,08
ajouter une activité à l'agenda (29)	44,7s	44,55s	0,01
ajouter un médecin (4)	60,3s	51,2s	0,18
supprimer un médecin (4)	18,9s	13,25s	0,42
modifier un médecin (8)	28,4s	27,55s	0,03

Les raisons pour lesquelles l'activité "supprimer une médecin" est plus longue dans la réalité que dans le modèle ne sont pas claires. Nous avons cependant observé un fait qui pourrait expliquer cette différence. En observant plus en détail la manière dont les utilisateurs effectuent cette tâche, nous avons remarqué qu'une partie du temps est utilisé par les utilisateurs pour trouver le bon médecin dans la liste. Lorsque l'on compare avec d'autres suppressions (comme supprimer une activité de la liste), les utilisateurs utilisent beaucoup plus de temps pour trouver un médecin que pour trouver une activité.

Malgré cette différence sur une activité, nous pouvons conclure que notre modèle est validé. Ainsi, nous avons effectué la première partie de notre processus. Ce modèle va nous être utile pour effectuer la deuxième partie : l'analyse automatique des traces d'utilisations à l'aide de l'apprentissage automatique.

3 Résumé

Nous avons utilisé différentes méthodes pour analyser l'interface de l'application AP@LZ. Tout d'abord, il a fallu définir les différentes tâches que l'on peut effectuer avec l'application. Ensuite, il a fallu déterminer comment effectuer ses tâches à l'aide de son interface (les différents chemins possibles). Puis, nous avons utilisé deux méthodes GOMS pour formellement connaître l'interface et prédire le temps nécessaire à un utilisateur pour effectuer chaque tâche. Cette analyse nous aide à mieux comprendre l'application et les tâches qu'elle permet d'effectuer, et constitue la première partie de notre processus pour déceler les erreurs d'utilisation et les problèmes d'utilisabilité dans les applications du laboratoire DOMUS. Nous pouvons donc maintenant passer à notre objectif 3 : créer un outil permettant l'analyse automatique des traces d'utilisation pour déceler les erreurs des utilisateurs et améliorer l'utilisabilité.

Chapitre 5

Utilisation de l'apprentissage automatique pour l'analyse des traces d'utilisation dans le but de détecter les erreurs

Il est assez facile de collecter les évènements utilisateurs. Malheureusement, ces données sont très riches en informations, mais prennent souvent beaucoup de temps à analyser. Il existe différentes manières de les analyser et nous avons décidé d'utiliser des algorithmes d'apprentissage automatique pour classifier ces données en tâches que l'utilisateur a accomplies pour essayer d'en détecter les erreurs. Tout d'abord, nous avons utilisé les connaissances tirées de l'analyse formelle de AP@LZ (la première partie de notre processus, présentée au Chapitre 4) pour traiter les traces d'utilisation et obtenir les classes nécessaires à notre classification. Nous avons choisi de classifier en tâches plutôt qu'une classification binaire erreur/sans erreur, car nous souhaitons que notre algorithme donne plus d'information. En effet, notre algorithme nous permet non seulement de détecter les erreurs, mais aussi d'étudier plus en détail la manière dont une tâche est réalisée.

1 Prétraitement des données

Les données brutes sont récupérées directement sur le téléphone intelligent où AP@LZ est installé. Elles se présentent sous la forme d'une suite d'entrées dans une base de données qui suit la description uniforme présentée au Chapitre 3. Cette forme n'est pas directement exploitable par les algorithmes (car la base de données peut faire plusieurs dizaines de milliers d'entrées), c'est pourquoi nous devons effectuer un prétraitement en nous appuyant sur les informations tirées de l'analyse formelle. Nous avons tout d'abord découpé les données en séquences qui pourraient être utilisées pour la classification. Les algorithmes que nous avons utilisés nécessitent un apprentissage supervisé, nous avons donc annoté les données pour créer un ensemble d'entraînement, un ensemble de test et un ensemble de validation. Les scénarios utilisés pour créer ces données demandent souvent la réalisation d'une même tâche plusieurs fois de suite. Cela signifie que dans les données récoltées, les exemples sont souvent groupés par classe. Pour éviter que cela crée un biais (par exemple, l'algorithme devrait une classe au début de l'apprentissage sans jamais revoir d'autres représentants de cette classe après), nous avons ajouté à la base de données une fonction qui associe à chaque séquence un nombre aléatoire. Ainsi, les séquences de nos ensembles de test, d'entraînement et de validation sont dans un ordre aléatoire, les exemples ne sont plus groupés par classe. Cet ordre est donc le même, quel que soit l'algorithme utilisé. Enfin, ces données ont été transformées pour les rendre utilisables par nos algorithmes.

1.1 Découpage

Les traces d'utilisation sont composées de milliers de lignes dans la base de données. La première solution à laquelle nous avons pensé était de fixer un temps maximal entre deux entrées. Si le temps écoulé dépassait un certain seuil, les deux entrées étaient considérées comme appartenant à une séquence différente. Toutefois, si l'utilisateur effectue trop d'opérations à la suite, le nombre de classes possibles, c'est-à-dire le nombre de tâches possibles, devient trop grand. Si une séquence de traces peut contenir plusieurs tâches (telles que définies dans le Chapitre 4), par exemple si

«Ajouter un médecin»

«Ajouter un rendez-vous»

«Ajouter un médecin puis ajouter un rendez-vous»
sont trois séquences différentes devant être classifiées dans trois classes différentes, le nombre de combinaisons possibles de tâches effectuées dans une séquence devient trop grand. Nous souhaitons que ces séquences soient classifiées en

«Ajouter un médecin»

«Ajouter un rendez-vous»

«Ajouter un médecin» puis «Ajouter un rendez-vous»

Donc deux classes en tout.

C'est pourquoi nous avons préféré découper les traces en utilisant la structure de AP@LZ (étudiée au Chapitre 4). En effet, dans AP@LZ, il est nécessaire de retourner à l'accueil entre chaque tâche. Chaque tâche commence et finit sur l'écran d'accueil. À partir de là, nous avons décidé de découper la base de données en séquences qui commencent et finissent à chaque retour à l'accueil. Ainsi il est impossible que plusieurs tâches soient effectuées dans une seule séquence. Cette hypothèse, qui concerne seulement AP@LZ, sera abordée comme une limitation de l'étude dans la discussion (cf Section 4).

1.2 Annotation

Les séquences ayant été découpées, il faut leur assigner une classe pour créer l'ensemble d'entraînement. C'est pour cela que nous avons développé une application permettant de revoir sous forme de film les traces d'utilisations. Pour chaque séquence, apparaît à l'écran une suite d'images qui correspondent aux écrans sur lesquels ont eu lieu les interactions et l'endroit où a eu lieu cette interaction. L'interface permet d'annoter la séquence et de lui donner une classe. Cela nous a permis non seulement de classifier les séquences, mais aussi, de regarder plus en détail les séquences où des erreurs d'utilisation ont été détectées pour savoir plus précisément ce qui s'est passé. Un petit nombre de séquence n'ont pas pu être classifiées de manière sure, car l'objectif de l'utilisateur n'était pas clair. Dans ce cas, les séquences ont été classifiées en erreur.

1.3 Représentation en Python

Après avoir classifié les données et créé les ensembles d'entraînement, il nous fallait les transférer depuis la base de données où elles étaient stockées jusqu'à notre environnement de développement Python. Nous avons décidé de représenter les séquences en tant qu'objet Python avec trois attributs :

duration : la durée en millisecondes de la séquence.

id : le numéro d'identification aléatoire de la séquence dans la base de données.

list_log : une liste de traces d'utilisation.

L'objet log (dont est composé l'attribut list_log) correspond à une entrée dans la base de données des traces d'utilisation, donc à une interaction (tel que définie au Chapitre 3 dans le standard) est représenté en Python comme ceci :

id : le numéro d'identification de la trace d'utilisation dans la base de données

timestamp : le moment auquel a eu lieu l'interaction (en millisecondes depuis l'Epoch, le 1er janvier 1970).

activity_name : le nom de l'activité Android sur laquelle l'interaction a eu lieu.

rclass_name : le nom du CIG sur lequel a eu lieu l'interaction.

Ces objets ayant été représentés en Python, il faut les transformer en vecteurs ou en matrices qui pourront être utilisés par nos algorithmes. Selon l'algorithme utilisé, le type d'entrée n'est pas le même, il s'agit soit d'une matrice qui est composée de vecteurs colonnes représentant chacun une interaction, soit d'un vecteur qui représente un histogramme de la séquence.

Une matrice (Figure 5.1) Dans ce cas, on transforme chaque séquence en matrice. Dans cette matrice, chaque colonne représente une interaction. Ces vecteurs d'interaction ont chacun trois composantes :

Temps : la première partie est un entier qui correspond au temps écoulé depuis la dernière interaction (en millisecondes) et vaut 0 pour la première interaction d'une séquence.

Écran : la seconde partie est un vecteur contenant un unique "1" à la position correspondant à l'écran sur lequel l'interaction a eu lieu et des "0" partout ailleurs (il y a 31 écrans possibles dans AP@LZ donc à chaque fois il y a trente

"0" et un unique "1").

CIG : la dernière partie est aussi composée d'un vecteur avec un unique "1" à la position du CIG sur lequel a eu lieu l'interaction et des "0" partout ailleurs.

Chaque interaction ainsi transformée en vecteur, les vecteurs sont mis côte à côte pour former une matrice, dans l'ordre où les interactions leur correspondant ont eu lieu.

Temps	0	1983	1359	706	1434
Écran	0	1	0	1	0
	0	0	0	0	0
	...	...	...	...	...
	0	0	0	0	1
	1	0	0	0	0
	0	0	1	0	0
	...	...	...	...	...
	0	0	0	0	0
GUI	0	0	1	1	0
	...	...	...	...	...
	0	1	0	0	0
	1	0	0	0	0
	0	0	0	0	0
	...	...	...	...	...
	0	0	0	0	1

Figure 5.1 – Matrice représentant une séquence de 5 interactions. Chaque colonne correspond à une interaction, donc une ligne dans la base de données. Pour chaque colonne, la première ligne correspond entre le temps écoulé entre cette interaction et la précédente. Les autres lignes permettent d'encoder l'écran et le CIG concerné par cette interaction.

Temps	5482	
Écran	2	
	0	
	...	
	1	
	1	
	1	
	...	
	0	
GUI	2	
	...	
	1	
	1	
	0	
	...	
	1	

Figure 5.2 – Vecteur représentant la même séquence de 5 interactions. La première ligne correspond au temps qu'a duré cette interaction, tandis que les suivantes représentent un histogramme des écrans et des CIG utilisés durant cette séquence d'interaction. Ce vecteur est donc la somme sur les colones de la matrice représentant la même séquence.

Un vecteur (Figure 5.2 Dans ce cas, on transforme chaque séquence en un vecteur, qui correspond à un histogramme, c'est-à-dire que le vecteur a les trois mêmes composantes que les vecteurs représentant une interaction, mais qu'au lieu de "1" et de

"0" se trouve le nombre de fois que ce CIG où cet écran a été utilisé dans la séquence.

Un vecteur représentant une séquence correspond en fait à la somme sur les colonnes de la matrice représentant cette même séquence.

1.4 Ensemble d'entraînement

Notre ensemble d'entraînement est assez petit, pour 9 classes, il contient entre 75 et 200 exemples pour chaque classe. Les classes sont les suivantes :

— ajouter une activité à la liste prédéfinie
— supprimer une activité de la liste prédéfinie
— modifier une activité de la liste prédéfinie
— ajouter une activité à l'agenda
— erreur activité
— ajouter un médecin
— supprimer un médecin
— modifier un médecin
— erreur médecin

Cet ensemble d'entraînement a été obtenu auprès de personnes saines. Pour simuler les atteintes cognitives que pourrait avoir un patient atteint de la maladie d'Alzheimer, nous n'avons pas donné d'entraînement aux utilisateurs. Ils ne connaissaient pas, ou peu, AP@LZ avant et ont effectué sur plusieurs jours un scénario qui leur était communiqué par courriel. Cela permettait d'avoir une charge cognitive assez élevée, car les utilisateurs devaient continuer leurs activités quotidiennes.

2 Présentation des algorithmes

Lorsque Python peut comprendre ces données, nous avons utilisé des algorithmes pour effectuer une classification : des réseaux de neurones. Comme expliqué au Chapitre 1, les réseaux de neurones sont des algorithmes d'apprentissage automatique. Dans notre cas, l'apprentissage est supervisé (nous disposons d'un jeu de données étiqueté), discriminatif (il prédit la probabilité qu'une séquence de traces d'utilisation donnée en entrée appartienne à une classe étant donnée cette séquence), et optimise

la fonction de log-probabilité.

Nous avons utilisé deux types de réseaux de neurones différents. Un réseau de neurones multicouche classique et un réseau de neurones à convolution.

Comme expliqué dans le Chapitre 1, ces algorithmes optimisent la fonction de log-probabilité en utilisant la descente du gradient. Soit $\mathbf{x}$ une séquence en entrée et y la classe à laquelle cette séquence appartient, le réseau de neurones estime alors $f(\mathbf{x})_c = p(y = c|\mathbf{x})$, la probabilité que $\mathbf{x}$ appartiennent à la classe c connaissant la séquence $\mathbf{x}$.

2.1 Réseau de neurones multicouche

Le premier type de réseau utilisé est celui décrit dans le Chapitre 1. C'est un réseau de neurones multicouche. Les équations le décrivant ayant été présentées dans l'état de l'art (Section 2.3, équations 1.15 à 1.18, 1.28 et 1.29), nous ne les représenterons pas.

Entrées

Ce réseau ne peut prendre en entrée que des vecteurs de taille fixe. Les traces d'utilisation n'ayant pas de taille maximale (car le nombre d'interactions que l'utilisateur peut avoir avant de retourner à l'accueil n'est pas limité), nous avons décidé d'utiliser comme entrée de ce réseau, le vecteur histogramme décrit plus haut.

Hyperparamètres

Il y a 8 hyper paramètres pour ce réseau de neurones :

— lr est le taux d'apprentissage.

— dc est la constante de diminution, par laquelle le taux d'apprentissage est divisé à chaque epoch : $\alpha = \frac{lr}{1+dc\times nombre}$

— $L1$ est le poids associé à la régularisation L1

— $L2$ est le poids associé à la régularisation L2

— $sizes$ est la taille et le nombre de couches cachées utilisées

— $tanh$ est un booléen qui indique si la fonction d'activation utilisée est tanh ou sigmoïde

— *seed* est l'initialisation du générateur de nombres aléatoires pour initialiser les poids

L'utilisation de l'histogramme en entrée à la place de la matrice implique une perte d'information sur la composante temporelle. Il est impossible de savoir combien de temps s'est écoulé entre deux interactions et l'ordre dans lequel elles se sont produites. De plus, la manière dont est fait le vecteur histogramme implique une imprécision dans la nature exacte des interactions. Si un CIG est présent dans deux activités Android différentes, alors il y a un risque que l'on ne puisse pas faire la différence entre les deux instances du CIG.

Pour ces deux raisons, nous avons décidé d'utiliser un autre type de réseau de neurones qui permet de prendre en entrée une matrice de taille variable et donc de garder toutes les informations sur les traces d'utilisation.

Choix des hyperparamètres Nous avons cherché quelle configuration des hyperparamètres avait la meilleure erreur de classification sur l'ensemble de validation grâce à une recherche selon une grille.

Nous avons effectué une recherche par grille grâce à l'algorithme suivant :

```
for  L2 in 0, 0.01, 0.03, 0.001, 0.003 do
  for  L1 in 0, 0.01, 0.03, 0.001, 0.003 do
    for  sizes in [20,20],[10,10],[30,30],[40,40] do
      for  dc in 0, 0.0000001, 0.000001, 0.00001 do
        for  lr in 0.1, 0.03, 0.003, 0.001  do
          for  seed in 1234, 5764, 9999  do
            for  tanh in True, False do  python run_nnet.py
$lr $dc $sizes $L2 $L1 $seed $tanh
            end for
          end for
        end for
      end for
    end for
```

```
    end for
  end for
```

Résultats

Nous n'avons trouvé aucune configuration pour laquelle l'erreur de classification sur l'ensemble de validation serait égale à 0. La meilleure configuration, celle ayant la plus petite erreur sur l'ensemble de validation, est la suivante : $lr = 0.03$, $dc =$ 0,00001 , $sizes = [40, 40]$, $L1 = 0.001$, $L2 = 0$, $seed = 9999$, $tanh =$ True.

Ce réseau de neurones obtient une erreur de classification de 0.016, c'est-à-dire un peu plus de 1% d'erreur sur l'ensemble de validation. En revanche, sur l'ensemble de test, l'erreur est de 0.13, ce qui fait 13% d'erreur.

La différence de performance entre les deux ensembles nous a poussés à chercher si la seconde meilleure configuration sur l'ensemble de validation ne ferait pas un meilleur résultat sur l'ensemble de test. Cette configuration obtient sur l'ensemble de validation, une erreur de 0,048, c'est-à-dire presque 5%, donc plus de 4 fois plus que la configuration précédente. Cette configuration est la suivante : $lr = 0.1$, $dc =$ 0,00001 , $sizes = [20, 20]$, $L1 = 0$, $L2 = 0.001$, $seed = 9999$, $tanh =$ False.

Elle obtient sur l'ensemble de test un taux d'erreur de 0.095, ce qui est un peu meilleur que la précédente, mais reste tout de même relativement élevé.

Ces variations de performances entre l'ensemble de validation et l'ensemble de test sont dues à la taille trop restreinte de notre ensemble d'entraînement.

Ce taux d'erreur étant assez élevé, nous avons décidé d'essayer un autre algorithme, peut-être plus adapté à notre problématique : le réseau de neurones à convolution. Celui-ci présente des avantages dont le plus important est la prise en compte de la dimension temporelle.

2.2 Réseau de neurones à convolution

Le réseau de neurones que nous avons utilisé s'inspire de celui décrit par Mikael Henaff [19] ainsi que celui décrit par Nal Kalchbrenner [28]. C'est un réseau de neurones à convolution optimisant une fonction de log-probabilité régularisée (avec les normes L1 et L2).

Entrées

Il prend en entrée la matrice représentant une séquence de traces d'utilisation décrite dans la Sous-section 1.3. Les différences avec les réseaux de convolution classique présentées au Chapitre 1 sont les suivantes.

Principes

Convolution La première couche du réseau utilise la convolution large. La seule différence avec la convolution étroite est le domaine de définition. Au Chapitre 1 (équation 1.32), nous avions défini la convolution étroite comme :

$$g(x) = (f \star h)(x) = \sum_{i=-\infty}^{\infty} f(x-i)h(i) \tag{5.1}$$

où g est définie pour toute valeur de x telle que $f(x-i)$ est définie pour tout i appartenant au domaine de définition de h. La convolution large permet d'agrandir ce domaine, g étant maintenant définie pour toute valeur de x pour laquelle il existe une valeur de i appartenant au domaine de définition de h telle que $f(x-i)$ est définie. Lorsque $f(x-i)$ n'est pas défini, alors on le considère égal à 0.

Par exemple, la convolution large du vecteur [**5 4 2 3 5 4 2**] par le filtre [**3 1 2**] donne le vecteur [**15 17 20 19 22 23 20 10 4**]. Cet exemple est illustré en Figure 5.3. En règle général, pour un vecteur de taille n et un filtre de taille m, le résultat de la convolution large sera de taille $n+m-1$. C'est parce que le résultat est toujours de taille supérieure ou égale à l'entrée que ce type de convolution est dit large.

Cette convolution a deux avantages principaux :

— il n'y a pas de supposition sur la taille de la séquence d'entrée, qui dans notre cas peut faire de 2 à plusieurs dizaines d'éléments de longs

— et les éléments sur les bords de la séquence ont le même poids que les autres lors du calcul, ils sont utilisés le même nombre de fois.

Ces deux avantages apparaissent sur la Figure 5.4. Ils sont aussi importants dans notre cas, car l'analyse effectuée au Chapitre 4 a montré que les interactions les plus importantes d'une trace d'utilisation sont les premières et les dernières. On voit aussi

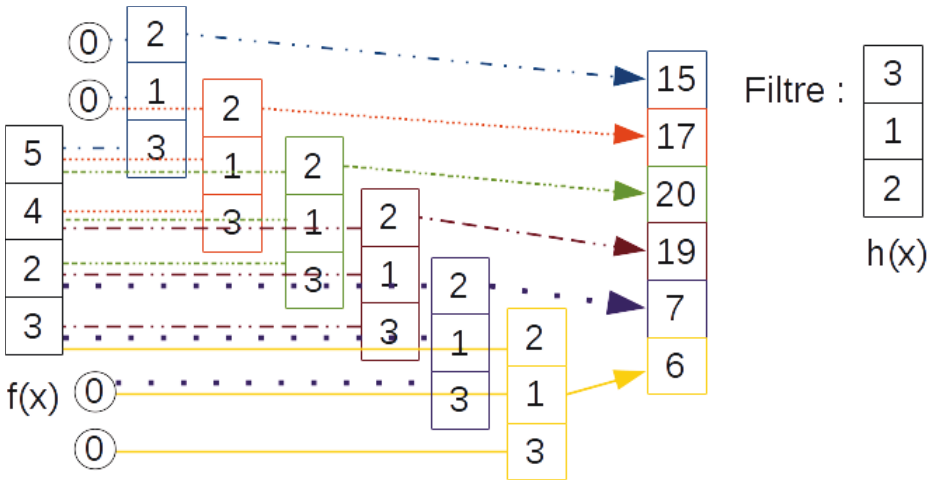


Figure 5.3 – Exemple de convolution large sur une entrée vectorielle.

sur la figure que la convolution large n'est qu'une généralisation de la convolution classique (qui est en gris sur la deuxième image). L'inconvénient de la convolution large est qu'elle nécessite plus de mémoire et de calcul. La convolution large conserve les propriétés de la convolution classique et permet au réseau de neurones de ne pas être sensible aux différences locales et invariants par translation.

Échantillonnage Après la couche de convolution, le réseau a une couche d'échantillonnage par le maximum (ou sous-échantillonnage) qui est suivie de l'ajout d'un biais et d'une fonction non linéaire. Cet échantillonnage est important. Premièrement, il permet de ne sélectionner que les neurones les plus activés, donc les plus représentatifs comme dans les réseaux de neurones à convolution habituels. Deuxièmement, parce qu'il est fait ligne par ligne (chaque voisinage est une ligne), le vecteur résultat est de taille constante (sa taille ne dépend pas de la taille de l'entrée).

Il ne reste plus qu'à ajouter une couche complètement connectée pour effectuer la classification.

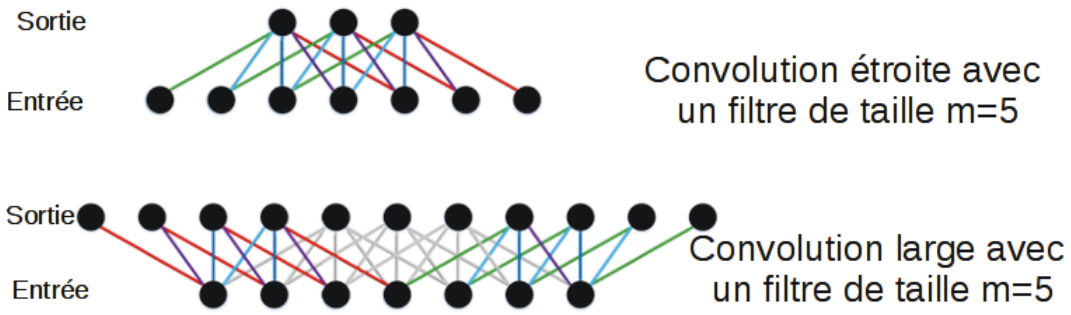


Figure 5.4 – Différences entre convolution classique (étroite) et convolution large. Sur chaque schéma, les points du bas correspondent à l'entrée (ici, de taille 7). Les points du haut correspondent à la sortie. Le filtre de taille 5 est représenté par les traits de couleur. Chacune des 5 couleurs correspond à un poids différent.

Propagations avant et arrière :

Un résumé de l'architecture du réseau de neurones est présenté en Figure 5.5.

Propagation avant La première étape est de calculer la convolution large avec les k filtres de taille n par m (sur la Figure 5.5, k=3, n=2 et m=3). L'échantillonnage est ensuite effectué sur les k cartes des patrons ainsi obtenues, seul le maximum de chaque ligne est conservé. Les k vecteurs qui en résultent sont mis ensemble pour former un grand vecteur auquel est ajouté un biais, avant d'y appliquer une fonction d'activation. Notre implémentation permet l'utilisation de deux fonctions d'activation différentes, la tangente hyperbolique (tanh) et la fonction sigmoïde. Le vecteur obtenu est la sortie de la couche d'entrée et l'entrée de la couche de sortie, donc la couche cachée. Cette couche cachée est ensuite connectée complètement de manière classique (une matrice de poids, suivie de l'ajout d'un biais) à une couche de sortie. Pour effectuer la classification, la fonction softmax est appliquée.

Notre réseau possède les composants suivants (cf Figure 5.5) :

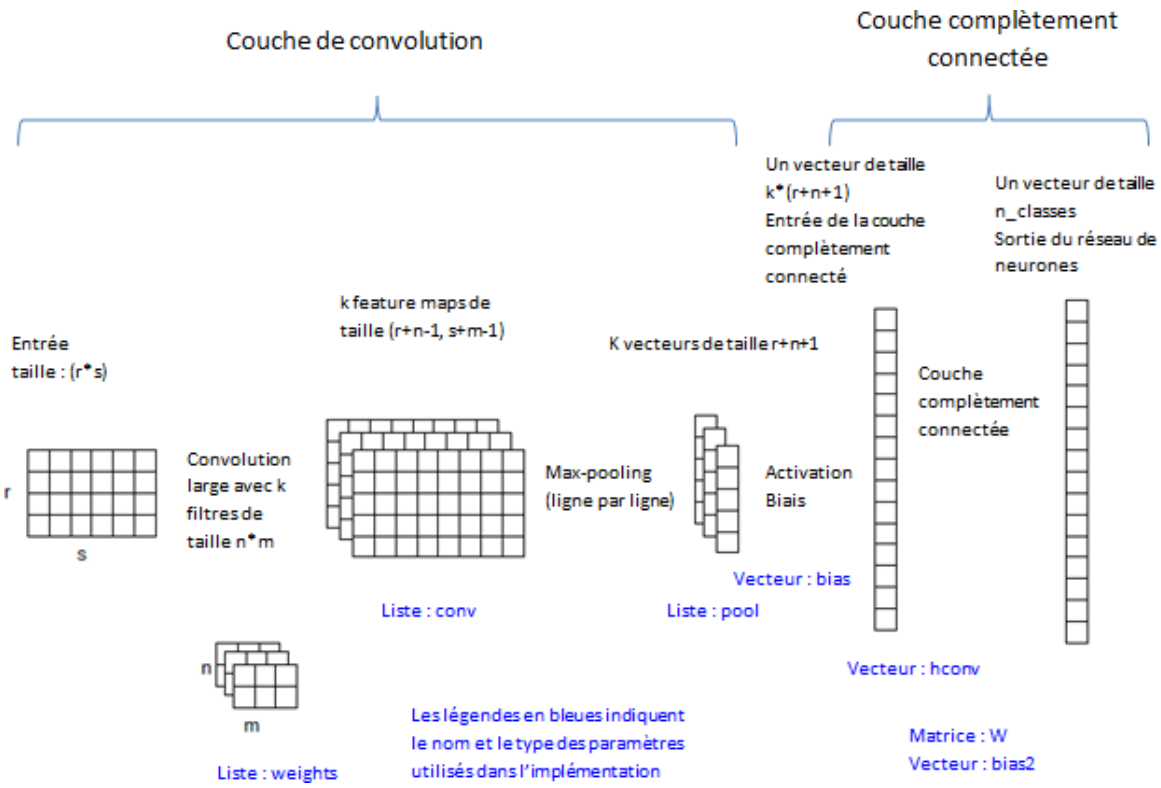


Figure 5.5 – Résumé de l'architecture du réseau de neurones. Ce réseau de neurones comprend une couche de convolution (convolution large puis échantillonnage puis biais et l'application d'une fonction d'activation) suivie par une couche complètement connectée.

— La première couche est l'entrée.
— Les filtres de convolution (dans notre implémentation «filters» est une liste de filtre).
— Le résultat de la convolution est stocké dans une liste de matrice *conv*. ($conv = input \star filter$)
— Le résultat de l'opération d'échantillonnage est une liste de vecteurs stockée dans la variable *pool*. ($pool = max(conv)$, pour chaque ligne).
— Le résultat de l'agrégation de ces vecteurs en un grand vecteur puis de l'ajout des biais est appelé *a*. ($a = pool + bias$)

— Le résultat de la fonction d'activation appliquée à ce vecteur est $hconv$. ($hconv = activation\_function(a)$)
— La matrice de poids s'appelle W, et le biais de la couche complètement connectée s'appelle $bias2$.
— Enfin, la sortie s'appelle $output$. ($output = \text{softmax}(W \cdot hconv + bias2)$)

Pendant la phase d'entraînement du réseau de neurones, à chaque propagation avant, correspond une propagation arrière.

Propagation arrière Nous utiliserons les mêmes notations qu'au Chapitre 1, c'est-à-dire que $\nabla_\theta f$ représente le gradient par rapport au paramètre θ de la fonction f et ∇_θ représente $\nabla_\theta(-\log(f(x)_y)$ (la fonction $(-\log(f(x)_y)$ est omise).

Tout d'abord, la rétropropagation dans la couche complètement cachée est exactement la même que dans un réseau multicouche classique.

Il faut obtenir le gradient avant la softmax :

$$\nabla_{out} = -(e(y) - f(x))$$

où $\mathbf{e}(y)$ est un vecteur avec un unique "1" à la y^{e} position et des "0"' partout ailleurs.

Pour obtenir le gradient de la matrice des poids de la couche complètement connectée (W) :

$$\nabla_W = \nabla_{out} hconv^T$$

Pour obtenir le gradient du biais de la couche complètement connectée :

$$\nabla_{bias2} = \nabla_{out}$$

Pour obtenir le gradient de la couche activée juste après la convolution :

$$\nabla_{hconv} = W^T \nabla_{out}$$

Pour obtenir le gradient de la couche pré-activation juste après la convolution :

$$\nabla_a = \nabla_{hconv} \odot \begin{pmatrix} \dots \\ g'((\mathbf{a})_j) \\ \dots \end{pmatrix}$$

où $\odot$ est la multiplication membre à membre entre vecteurs et g' la dérivée de la fonction d'activation qui est appliquée à chaque membre du vecteur $\mathbf{a}$.

Il faut ensuite continuer la propagation arrière dans la couche de convolution.

Tout d'abord, le calcul du gradient du biais est le même que dans une couche complètement connectée, donc :

$$\nabla_{bias} = \nabla_a$$

La rétropropagation du gradient à travers le sous-échantillonnage se fait de manière à ce que le gradient ne se propage que sur les neurones sélectionnés dans la couche de convolution (c'est-à-dire les neurones qui avaient la plus haute sortie sur chaque voisinage) :

$$(\nabla_{conv})_{i,j} = \begin{cases} (\nabla_a)_{i,j} & \text{si } (i,j) \text{ a été selctionné} \\ 0 & \text{sinon} \end{cases}$$

Ensuite, la rétropropagation à travers l'opération de convolution se fait grâce à l'opérateur de corrélation, donc pour chaque filtre de convolution :

$$\nabla_{filter} = \nabla_{conv} \heartsuit \ input$$

où $\heartsuit$ est l'opérateur de corrélation étroite défini au 1 (équation 1.33)

Les gradients ayant été calculés, pour chaque exemple les paramètres sont mis à jour. L1 et L2, les poids associés aux régularisations du même nom ne sont pas présentés pour ne pas compliquer les formules, mais ils sont cependant utilisés dans les calculs.

Hyperparamètres

Il y a 8 hyperparamètres pour ce réseau de neurones :

— lr est le taux d'apprentissage.

— dc est la constante de diminution, par laquelle le taux d'apprentissage est divisé à chaque epoch : $\alpha = \frac{lr}{1+dc \times nombre}$

— $L1$ est le poids associé à la régularisation L1

— $L2$ est le poids associé à la régularisation L2

— $sizes$ est la taille (largeur et hauteur) des filtres de convolution utilisés

— k est le nombre de filtres de convolution utilisés

— $tanh$ est un booléen qui indique si la fonction d'activation utilisée est tanh ou sigmoide

— $seed$ est l'initialisation du générateur de nombres aléatoires pour initialiser les poids

Optimisation des hyperparamètres Nous avons cherché quelle configuration des hyperparamètres avait la meilleure erreur de classification sur l'ensemble de validation grâce à une recherche selon une grille.

Dans ce cas, les hyperparamètres étant différents, nous avons utilisé le code suivant pour effectuer la recherche en grille :

```
for  L2 in 0, 0.0001, 0.0001, 0.01  do
   for  L1 in 0, 0.0001, 0.0001, 0.01  do
      for  sizes in [2,1], [2, 2], [3, 2], [3, 1] do
         for  dc in 0, 0.0000001, 0.000001, 0.00001, 0.0001  do
            for  lr in 0.1, 0.03, 0.01, 0.003  do
               for  k in 2, 3, 4, 5  do
                  for  seed in 1234, 9999  do
                     for  tanh in True, False do
python run_covnet.py $lr $ dac $sizes $L2 $k $L1 $seed $tanh
                     end for
                  end for
```

```
            end for
          end for
        end for
      end for
    end for
  end for
```

Résultats

La première chose que nous avons constatée est que pour beaucoup des configurations possibles, l'algorithme avait une erreur de classification de 0. Il ne s'agit malheureusement pas d'une preuve de l'efficacité de l'algorithme, mais plutôt du fait que notre ensemble de validation est assez petit (172 séquences). Pour différencier parmi les différentes configurations d'hyperparamètres ayant une erreur de classification nulle, nous avons choisi de tester ces configurations sur l'ensemble de test. De nouveau,comme l'ensemble de test est aussi assez petit (173 séquences), plusieurs configurations obtenaient une erreur de classification nulle. Nous avons donc choisi de sélectionner comme étant la meilleure, celle qui obtenaient la plus petite log-probabilité sur ces deux ensembles. Les valeurs des hyperparamètres de cette configuration sont : $lr = 0.1$, $dc = 0$, $sizes = [2, 2]$, $L1 = 0$, $L2 = 0$, $k = 4$, $seed = 1234$, $tanh = \text{True}$.

Ce que l'on constate est que les configurations des hyperparamètres qui fonctionnent le mieux ont 4 filtres qui sont des carrés de côté 2. Il est difficile de tirer d'autres conclusions étant donné la diversité des configurations obtenant une erreur de classification nulle. Il faudrait, pour une analyse approfondie, avoir un ensemble d'entraînement plus grand.

3 Erreurs détectées

Ces algorithmes ont été utilisés sur les traces d'utilisation produites par le patient atteint de DTA (Démence de Type Alzheimer). Ils ont classifié un certain nombre de séquences en "erreur". Après un examen de chacune de ces séquences, nous avons détecté 3 erreurs d'utilisation que nous présentons ici :

3.1 Erreur 1

La première est une erreur de complétion, qui se produit lorsque l'utilisateur ajoute une activité à son agenda. À ce moment-là, l'utilisateur est sur un écran (voir Figure 5.6) qui lui demande de renseigner la date et l'heure à laquelle il souhaite ajouter l'activité ainsi que le rappel s'il en souhaite un.

Figure 5.6 – Écran de choix de la date, de l'heure et du rappel.

Lorsque le patient a complété les 5 champs, il a l'impression que sa tâche est terminée. Comme le bouton de retour (la flèche bleue) est plus attirant que le bouton «Ajouter», il arrive que le patient choisisse d'utiliser ce dernier. Malheureusement, le bouton de retour à l'accueil annule tout et le patient doit reprendre du début.

3.2 Erreur 2

La seconde erreur se produit lorsque l'utilisateur souhaite modifier ou supprimer une activité. Pour cela, il doit tout d'abord accéder à la liste des activités. Sur cet écran, pour supprimer ou modifier une activité, il faut cliquer sur les boutons dédiés en bas de cette liste. cf Figure 5.7.

Malheureusement, il arrive souvent que l'utilisateur choisisse plutôt de cliquer sur le nom de l'activité qu'il souhaite supprimer. Il doit alors retourner à l'accueil et re-

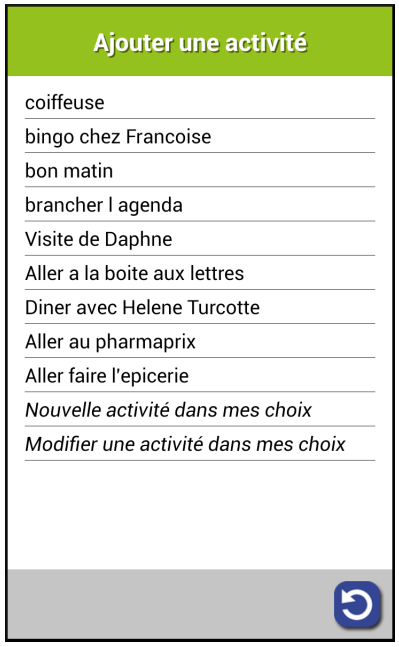


Figure 5.7 – Liste des activités prédéfinies.

commencer.

3.3 Erreur 3

La dernière erreur se produit lorsque l'utilisateur souhaite modifier une information dans l'application. Lorsque l'on ajoute une information à l'interface, le bouton «Enregister» se trouve en bas, au milieu de l'écran voir Figure 5.8 pour un exemple lorsque l'on ajoute un médecin.

Lorsque l'on souhaite modifier une information entrée, le bouton «Enregistrer» se trouve en bas à gauche de l'écran et c'est le bouton «Effacer» qui se trouve au milieu (cf Figure 5.9) . Ces deux mots ayant de plus des graphies similaires, il arrive que l'utilisateur utilise le bouton «Effacer» par habitude sans se rendre compte que toutes les informations vont être effacées. Nos utilisateurs étant atteints de la maladie d'Alzheimer, il y a de plus un risque qu'ils ne se souviennent pas des données qui ont été effacées ni même du fait qu'ils ont accidentellement effacé quelque chose.

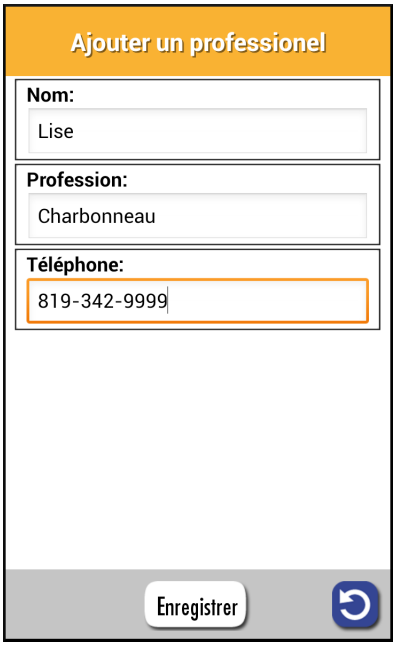


Figure 5.8 – Ajout d'un médecin dans l'application

4 Discussion

L'analyse effectuée au Chapitre 4 a permis de définir les classes que nous avons utilisées. L'utilisation d'algorithmes d'apprentissage automatique permet effectivement de classifier les traces d'utilisation selon les tâches effectuées. Si la taille de notre ensemble d'entraînement nous a limités durant l'apprentissage, nous avons malgré tout pu détecter trois erreurs. L'outil d'apprentissage automatique est peut-être un peu trop complexe par rapport la tâche à effectuer étant donné la simplicité de l'application étudiée, mais il a l'avantage que n'importe quelle application utilisant la description des traces d'utilisation données au Chapitre 3 pourra réutiliser cet outil avec un minimum de changement dans le code. Un certain nombre d'hypothèses ont été faites pour que ces algorithmes fonctionnent, la plus importante étant le retour à l'accueil systématique entre chaque séquence. Cette contrainte très forte est vérifiée grâce à la structure d'AP@LZ, mais ne pourra pas être implémentée sur d'autres applications dont l'architecture est moins rigide.

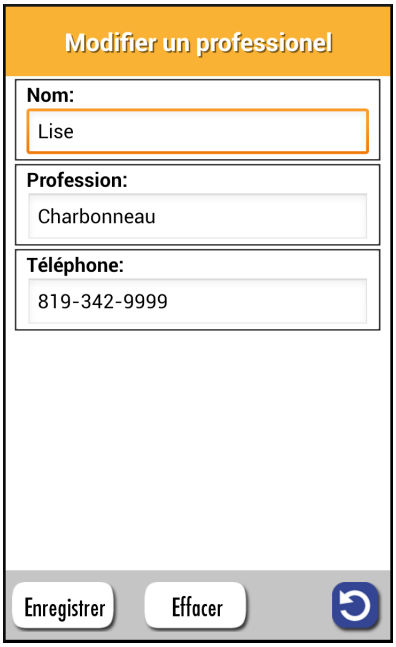


Figure 5.9 – Modification d'un médecin de l'application

Conclusion

1 Discussion

Différents outils ont été créés pour aider à la mise en place des études d'utilisabilité. Le premier s'appelle GEDOPAL et sert à stocker les données produites durant les études d'utilisabilité puis d'en permettre l'accès et la visualisation par les chercheurs de manière sécurisée. Cet outil utilise une base de données centralisée au laboratoire DOMUS qui est synchronisée en Master-to-Master avec des copies locales se trouvant sur les ordinateurs des utilisateurs. L'accès se fait à travers le RPV de l'Université de Sherbrooke pour une meilleure authentification et utilise une gestion des accès par groupe. Il a été développé en Eclipse RCP ce qui permet une grande modularité, mais aussi une grande robustesse, même si le temps d'apprentissage est assez long et l'installation assez complexe. Cet outil est utilisé actuellement par des chercheurs et, même s'il a quelques limites, il fonctionne et permet une meilleure organisation dans les études d'utilisabilité. Parmi les limites de GEDOPAL, certaines ont été prévues dès la conception. GEDOPAL ne gère ni les entrées vidéo ni les entrées audio. D'autres limitations sont apparues à l'usage : le modèle de données que nous avons créé pour GEDOPAL est sans doute trop rigide et ne donne pas aux utilisateurs autant de liberté qu'ils le souhaiteraient. De plus, GEDOPAL a été pensé trop centré sur le laboratoire DOMUS, d'autres laboratoires effectuant souvent des études d'utilisabilité trouveraient GEDOPAL très utile, mais ne peuvent aujourd'hui pas l'utiliser à cause du manque de flexibilité. Les types de données prises en compte par exemple sont spécifiques au laboratoire DOMUS.

GEDOPAL permet le stockage et la visualisation des données qu'il faut ensuite

analyser. Pour cela, nous avons utilisé un processus : d'abord analyser l'application à l'aide de méthodes formelles pour mieux la décrire puis utiliser ces résultats pour paramétrer un deuxième outil qui permet l'analyse automatique des traces d'utilisation. Ce processus permet non seulement de détecter les erreurs d'utilisation commises, mais aussi de classifier les traces d'utilisation en tâches qui peuvent ensuite être analysées plus en détail. L'utilisation de l'apprentissage automatique est peut-être complexe, en particulier dans ce but qui est relativement simple, mais elle a l'avantage d'être très généralisable. N'importe quelle application générant des traces d'utilisation conformes à celles décrites dans ce mémoire (cf chapitre 4) pourra, avec un minimum de changement dans le code, utiliser ces outils. Ce processus montre des résultats extrêmement encourageants, il a permis de détecter des erreurs d'utilisation dans AP@LZ. Ces résultats doivent maintenant être validés avec d'autres applications.

2 Travaux futurs

Nous avons identifié deux limites à ce processus, qui devront être surmontées lors de son utilisation avec d'autres applications. Tout d'abord, une application comme AP@LZ est limitée, dans le sens où elle ne permet d'effectuer que peu de tâches. Une application offrant plus de possibilités aura donc un ensemble de classes beaucoup plus grand. La classification risque alors de devenir difficile. Une solution à laquelle nous avons pensé est d'utiliser deux niveaux, par exemple, un premier niveau classifie les traces d'utilisation en tâches de bas niveau et un second niveau classifie ces tâches de bas niveau en classes de plus haut niveau (dans les deux cas, la granularité des classes dépend de la taille de l'application et peut varier si l'on souhaite ajouter plus de deux niveaux).

Une autre limite que l'on constate ici est le prétraitement des données : dans AP@LZ, il est trivial de découper les traces d'utilisation à chaque retour à l'accueil, mais une application moins contraignante risque de ne pas avoir de découpage aussi simple. Une solution à ce problème est de chercher durant l'analyse formelle de l'application, quelles sont les fins de tâches, par exemple, pourrait-on définir certain CIG comme représentant la fin d'une tâche. La découpe des traces deviendrait alors plus simple.

De plus, un inconvénient de notre découpage en tâche est que si deux séquences successives sont reliées, ce lien risque d'être perdu (la suppression d'une activité suivie de la création de cette activité indique probablement une erreur). Il faudrait alors développer un deuxième outil qui utiliserait les résultats de notre classification et reviendrait à l'ordre chronologique des séquences pour découvrir ce genre de cas.

Annexe A

Description selon GOMS

1 Liste des tâches possibles avec AP@LZ

Médical :

Médicaments :

- Ajouter un médicament
- Supprimer un médicament
- Modifier un médicament
- Consulter ses médicaments

Médecins :

- Ajouter un médecin
- Supprimer un médecin
- Modifier un médecin
- Consulter ses médecins

Historique :

- Ajouter un événement médical
- Supprimer un événement médical
- Modifier un événement médical
- Consulter son historique médical

Contacts :

— Appeler un contact
— Consulter les informations d'un contact
— Annuler l'appel d'un contact

Info person :
— Consulter ses informations personnelles

Bloc-notes :
— Ajouter une note
— Supprimer une note
— Modifier une note
— Consulter ses notes

Photos :
— Consulter ses photos

Activités+ :
— Réaliser une activité
— Supprimer une activité de l'agenda
— Modifier une activité à l'agenda
— Ajouter une activité prédéfinie à l'agenda
— Ajouter une activité prédéfinie puis l'ajouter à l'agenda
— Modifier une activité à l'agenda
— Consulter une activité à l'agenda
— Annuler une suppression
— Supprimer une activité prédéfinie
— Modifier une activité prédéfinie
— Ajouter une activité prédéfinie

navigation :
— Consulter les activités des jours passés

— Consulter les activités des jours futurs

2 Liste GUI par écran

ApalzActivity :

- ListView dayAppointmentsView
- Button leftArrow
- Button rightArrow
- Button appointmentButton
- Button phoneButton
- Button medicalButton
- Button personalButton
- Button photoButton
- Button notebookButton
- TextView dayTitle

AppointmentActivity :

- Button addButton
- Button backButton
- Button arrowButton
- Button dayLeftArrow
- Button dayRightArrow
- Button monthLeftArrow
- Button monthRightArrow
- Button hourLeftArrow
- Button hourRightArrow
- Button minuteLeftArrow
- Button minuteRightArrow
- Button reminderLeftArrow
- Button reminderRightArrow

PersonalActivity :

— Button backButton

PhotoActivity :

— Button backButton
— Button nextRightArrow
— Button previousLeftArrow

NotebookActivity :

— ListView noteBookView
— Button backButton
— Button addButton

MedicalActivity :

— Button backButton
— Button historyButton
— Button drugButton
— Button teamButton

ContactActivity :

— GridView contactGridView
— Button backButton
— ImageView contactPhotoThbnail
— ImageView contactPhoneIcon

MedicalTeamList :

— Button backButton
— Button addButton
— ListView medicalTeamView

MedicalTeamAdd :

— Button validateButton

— Button backButton
— clavier

MedicalTeamModify :
— Button saveButton
— Button deleteButton
— Button backButton
— clavier

MedicalHistoryList :
— ListView medicalHistoryView
— Button backButton
— Button addButton

MedicalHistoryAdd :
— Button saveButton
— Button backButton
— Button formatLeftArrow
— Button formatRightArrow
— Button yearLeftArrow
— Button yearRightArrow
— Button dayLeftArrow
— Button dayRightArrow
— Button monthLeftArrow
— Button monthRightArrow
— clavier

MedicalHistoryModify :
— Button saveButton
— Button backButton
— Button deleteButton
— Button formatLeftArrow

— Button formatRightArrow
— Button yearLeftArrow
— Button yearRightArrow
— Button dayLeftArrow
— Button dayRightArrow
— Button monthLeftArrow
— Button monthRightArrow clavier

MedicalDrugList :
— ListView medicalDrugsView
— Button backButton
— Button addButton

MedicalDrugAdd :
— Button saveButton
— Button backButton
— clavier

MedicalDrugModify :
— Button saveButton
— Button deleteButton
— Button backButton
— clavier

NotebookAdd :
— Button backButton
— Button saveButton
— clavier

NotebookModify :
— Button deleteButton

— Button updateButton
— Button backButton
— clavier

KnownActivitiesMainList :
— ListView knownActivitiesView
— Button backButton

KnownActivitiesModifyList :
— ListView knownActivitiesView
— Button backButton

KnownActivitiesModChoice :
— Button saveButton
— Button deleteButton
— Button backButton
— clavier

KnownActivitiesAddChoice :
— Button validateButton
— Button backButton
— clavier

AppointmentDetails :
— Button suppressButton
— Button modifyButton
— Button realizeButton
— Button backButton
— Button yesButton
— Button noButton

AppointmentAlarm :

— Button realizeButton

— Button backButton

ContactInfo :

— Button backButton

— Button phoneButton

ContactDialer :

— Button phoneButton

— Button backButton

AppointmentConfirmation :

— Button backButton

AppointmentConflict :

— Button backButton

NotebookConfirmation :

— Button backButton

NotebookModification :

— Button backButton

NotebookSuppression :

— Button backButton

3 Chemins pour chaque tâches

Supprimer une activité de l'agenda :

ApalzActivity → dayAppointmentsView

(AppointmentDetails → suppressButton)
AppointmentDetails → yesButton

Modifier une activité à l'agenda :
ApalzActivity → dayAppointmentsView
AppointmentDetails → modifyButton
(AppointmentActivity → arrows)
AppointmentActivity → addButton

Supprimer une activité connue :
ApalzActivity → appointmentButton
KnownActivitiesMainList → konwnActivitiesView
KnownActivitiesModifyList→ konwnActivitiesView
KnownActivitiesModChoice → deleteButton

Modifier une activité connue :
ApalzActivity → appointmentButton
KnownActivitiesMainList → konwnActivitiesView
KnownActivitiesModifyList → konwnActivitiesView
(KnownActivitiesModChoice → clavier)
KnownActivitiesModChoice → saveButton

Supprimer un événement médical :
ApalzActivity → medicalButton
MedicalActivity → historyButton
MedicalHistoryList → medicalHistoryView
(MedicalHistoryModify → Arrows)
MedicalHistoryModify → deleteButton

Modifier un médecin :

ApalzActivity → medicalButton

MedicalActivity → teamButton

MedicalTeamList → medicalTeamView

(MedicalTeamModify → clavier)

MedicalTeamModify → saveButton

Supprimer une note :

ApalzActivity → notebookButton

NotebookActivity → noteBookView

NotebookModify → deleteButton

Modifier une note :

ApalzActivity → notebookButton

NotebookActivity → noteBookView

(NotebookModify → clavier)

NotebookModify → updateButton

Ajouter une activité connue à l'agenda :

ApalzActivity → appointmentButton

KnownActivitiesMainList → konwnActivitiesView

(AppointmentActivity → Arrows)

AppointmentActivity → addButton

Ajouter une activité INconnue à l'agenda :

ApalzActivity → appointmentButton

KnownActivitiesMainList → konwnActivitiesView

(KnownActivitiesAddChoice → clavier)

KnownActivitiesAddChoice → validateButton

(AppointmentActivity → Arrows)

AppointmentActivity → addButton

Réaliser une activité à l'agenda (sans alarme) :

ApalzActivity → dayAppointmentsView

AppointmentDetails → realizeButton

Appeler un contact :

ApalzActivity → phoneButton

select : { ContactActivity → contactPhotoThbnail ; ContactInfo → phoneButton ; *ou* ; ContactActivity → contactPhoneIcon ; ContactDialer → phoneButton

Consulter ses médicaments :

ApalzActivity → medicalButton

MedicalActivity → drugButton

Consulter ses infos personnelles :

ApalzActivity → personalButton

Ajouter un médicament :

ApalzActivity → medicalButton

MedicalActivity → drugButton

MedicalDrugList → addButton

(MedicalDrugAdd → clavier)

MedicalDrugAdd → saveButton

Consulter ses photos :

ApalzActivity → photoButton
(PhotoActivity → Arrows)

Consulter les infos d'un contact :

ApalzActivity → phoneButton
ContactActivity → contactPhotoThbnail
ContactInfo → backButton

Ajouter une activité connue :

ApalzActivity → appointmentButton
KnownActivitiesMainList → konwnActivitiesView
(KnownActivitiesAddChoice → clavier)
KnownActivitiesAddChoice → validateButton
AppointmentActivity → backButton

Modifier une activité à l'agenda :

ApalzActivity → dayAppointmentsView
AppointmentDetails → modifyButton
(AppointmentActivity → Arrows)
AppointmentActivity → addButton

Annexe B

Scénario des expérimentations

1 Scénario

1er jour :

En donnant le téléphone :

Ajouter les activités à la liste prédéfinie :

Réunion de labo, lieu : salle de Conférence

5@8, lieu : atrium

Réveil, pas de lieu

Brancher l'agenda, lieu : chez nous

Dîner, lieu : laboratoire DOMUS

Réunion projet, lieu : salle du DOMUS

Épicerie, lieu : Provigo

Cinéma, lieu : la maison du cinéma

Bière, lieu : Siboire

Expérimentation, lieu : laboratoire DOMUS

Ajouter les médecins :

Sohpie Turcotte :

Medecin de famille

819-555-2222

Mathieu Faubert :
Dentiste
819-555-4132

Simon Gagnon :
Physiothérapeute
819-555-0868

David Sorel :
Osteopathe
819-555-0088

Prévoir l'activité 5@8 jeudi prochain à 16 h
Prévoir le réveil pour demain matin (heure habituelle de réveil)
Prévoir le dîner aujourd'hui à 12 h.
Prévoir de brancher l'agenda ce soir

Premier courriel :

Ajouter à l'agenda une réunion de labo mardi prochain à 11 h.
Ajouter à l'agenda une épicerie lundi prochain à 18 h.
Ajouter à l'agenda une réunion de projet vendredi à 13 h 30.
Ajouter l'activité RDV avec Suzanne Turcotte à la liste prédéfinie (pour le lieu, mettre la profession indiquée dans la liste de professionnels).
Modifier le numéro de téléphone de Pierre Sorel en 819-821-9999.
Dans le courriel de réponse, indiquer le numéro de téléphone de Simon.

Deuxième courriel :

La réunion de labo est annulée, à la place, Ajouter à l'agenda une réunion de projet 30 minutes plus tard.
Ajouter à l'agenda un cinéma samedi prochain à 19 h 30.
Ajouter à l'agenda un RDV avec Suzanne Turcotte mercredi prochain à 15h.

Ajouter l'activité RDV avec Faubert à la liste prédéfinie (pour le lieu mettre la profession indiquée dans la liste de professionnel).
Modifier le numéro de Simon en 819-562-2222.
Dans le courriel de réponse, indiquer le numéro de téléphone de Pierre Sorel.

Troisième courriel :

Ajouter à l'agenda une réunion de labo jeudi prochain à 10 h 30.
Ajouter à l'agenda l'activité bière vendredi prochain à 17 h 30.
Ajouter à l'agenda un RDV avec Fauber dimanche prochain à 14 h 30.
Ajouter l'activité RDV avec Simon à la liste prédéfinie (pour le lieu, mettre la profession indiquée dans la liste de professionnels).
Modifier le numéro de téléphone de Faubert en 819-555-1234.
Dans le courriel de réponse, indiquer le numéro de téléphone de Suzanne Turcotte.

Quatrième courriel :

Ajouter à l'agenda une expérimentation vendredi prochain à 9 h.
Le cinéma est annulé, à la place ajouter à l'agenda une activité Bière 30 minutes plus tard.
Ajouter à l'agenda une réunion de projet jeudi prochain à 14 h 30.
Ajouter à l'agenda un RDV avec Simon mardi prochain à 16 h.
Ajouter l'activité RDV avec Pierre Sorel à la liste prédéfinie (pour le lieu, mettre la profession indiquée dans la liste de professionnels).
Modifier le numéro de téléphone de Suzanne Turcotte en 819-780-3333.
Dans le courriel de réponse, indiquer le numéro de téléphone de Faubert.

Mettre le réveil (heure au choix) et le dîner (plutôt midi, mais au choix) pour le lendemain .
Prévoir l'activité 5@8 mercredi prochain à 17 h dans l'agenda.
Prévoir de brancher l'agenda ce soir.

2e jour :

Premier courriel :

Ajouter à l'agenda une réunion de labo mardi prochain à 14 h.
Ajouter à l'agenda une épicerie mercredi prochain à 18 h.
Ajouter à l'agenda une réunion de projet samedi à 13h30h.
Ajouter à l'agenda un RDV avec Pierre Sorel mercredi prochain à 11 h.
Modifier le numéro de téléphone de Pierre Sorel en 819-821-8800.
Dans le courriel de réponse, indiquer la profession de Simon.

Deuxième courriel :

La réunion de labo est annulée, à la place, ajouter à l'agenda une réunion de projet 30 minutes plus tôt.
Ajouter à l'agenda un cinéma dimanche prochain à 19 h 30.
Ajouter à l'agenda un RDV avec Suzanne Turcotte samedi prochain à 11 h.
Modifier le numéro de Simon en 819-562-4554.
Dans le courriel de réponse, indiquer la profession de Pierre Sorel.

Troisième courriel :

Ajouter à l'agenda une réunion de labo jeudi prochain à 11 h 30.
Ajouter à l'agenda l'activité bière dimanche prochain à 17 h.
Ajouter à l'agenda un RDV avec Fauber dimanche prochain à 9 h 30.
Supprimer l'activité RDV avec Pierre Sorel de la liste prédéfinie.
Supprimer le médecin Pierre Sorel.
Modifier le numéro de téléphone de Faubert en 819-555-5432.
Dans le courriel de réponse, indiquer la profession de Faubert.

Quatrième courriel :

Ajouter à l'agenda une expérimentation mercredi prochain à 9 h.
Ajouter à l'agenda une activité Bière lundi prochain à 11 h 30.
Ajouter à l'agenda une Réunion de projet vendredi prochain à 10 h 30.
Ajouter à l'agenda un RDV avec Simon lundi prochain à 16 h.

Supprimer l'activité RDV avec Fauber de la liste prédéfinie.

Supprimer le médecin Fauber.

Modifier le numéro de téléphone de Suzanne Turcotte en 819-780-8888.

Dans le courriel de réponse, indiquer la profession de Suzanne Turcotte.

En reprenant le téléphone :

Supprimer toutes les activités et tous les médecins restants.

Bibliographie

[1] David Akers, Robin Jeffries, Matthew Simpson et Terry Winograd.
« Backtracking Events as Indicators of Usability Problems in Creation-Oriented Applications ».
ACM Trans. Comput.-Hum. Interact., pages 16–16, 2012.

[2] Florence Balagtas-Fernandez et Heinrich Hussmann.
« A Methodology and Framework to Simplify Usability Analysis of Mobile Applications ».
Dans *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, pages 520–524. IEEE Computer Society, 2009.

[3] Xiaojun Bi, Yang Li et Shumin Zhai.
« Fitts Law : Modeling Finger Touch with Fitts' Law ».
Dans *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1363–1372. ACM, 2013.

[4] Christopher M. Bishop.
Neural Networks for Pattern Recognition.
Clarendon Press, 1995.

[5] Tom Brinck et Erik Hofer.
« Automatically Evaluating the Usability of Web Sites ».
Dans *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 906–907. ACM, 2002.

[6] Stuart K. Card, Thomas P. Moran et Allen Newell.
The Psychology of Human-Computer Interaction.

Taylor & Francis, 1983.

[7] Larry L. Constantine et Lucy A. D. Lockwood.
Software for Use : A Practical Guide to the Models and Methods of Usage-centered Design.
ACM Press/Addison-Wesley Publishing Co., 1999.

[8] Antoine Cornuéjols, Laurent Miclet et Yves Kodratoff.
Apprentissage artificiel : Concepts et algorithmes.
Eyrolles, 2002.

[9] Yann Le Cun et Yoshua Bengio.
« Word-level training of a handwritten word recognizer based on convolutional neural networks ».
Dans *International Conference on Pattern Recognition*, 1994.

[10] Dick de Ridder.
« Shared weights neural networks in image analysis ».
Delft University of Technology, Thèse de doctorat, 1996.

[11] Alan Dix, Janet Finley, Gregory Abowd et Russell Beale.
Human-computer Interaction (2Nd Ed.).
Prentice-Hall, Inc., 1998.

[12] Gérard Dreyfus, Manuel Samuelides, Jean-Marc Martinez, Mirta B. Gordon, Fouad Badran, Sylvie Thiria et Laurent Hérault.
Réseaux de neurones : Méthodologie et applications.
Eyrolles, 2004.

[13] Société Alzheimer du Canada.
« Une nouvelle façon de voir l'impact de la maladie d'Alzheimer et des maladies apparentées au Canada », 2012.

[14] Joseph S. Dumas, Rolf Molich et Robin Jeffries.
« Describing Usability Problems : Are We Sending the Right Message ? ».
interactions, 11(4):24–29, juillet 2004.

[15] Joseph S. Dumas et Janice Redish.
A practical guide to usability testing.
Intellect Ltd, 1999.

[16] Carolanne Fisher et Penelope Sanderson.
« Exploratory Sequential Data Analysis : Traditions, Techniques and Tools ».
SIGCHI Bull., 25(1):34–40, janvier 1993.

[17] Carolanne Fisher et Penelope Sanderson.
« Exploratory Sequential Data Analysis : Exploring Continuous Observational Data ».
interactions, 3(2):25–34, mars 1996.

[18] Steven Heim.
The Resonant Interface : HCI Foundations for Interaction Design.
Addison Wesley, March 2007.

[19] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu et Yann LeCun.
« Unsupervised Learning of Sparse Features for Scalable Audio Classification ».
Dans *Proceedings of International Symposium on Music Information Retrieval (ISMIR'11)*, 2011.

[20] David M. Hilbert et David F. Redmiles.
« Extracting Usability Information from User Interface Events ».
ACM Comput. Surv., 32(4):384–421, décembre 2000.

[21] David M. Hilbert, Jason E. Robbins et David F. Redmiles.
« EDEM : Intelligent Agents for Collecting Usage Data and Increasing User Involvement in Development ».
Dans *Proceedings of the 3rd International Conference on Intelligent User Interfaces*, IUI '98, pages 73–76. ACM, 1998.

[22] K. Hornik, M. Stinchcombe et H. White.
« Multilayer feedforward networks are universal approximators ».
Neural Networks, 2(5):359–366, 1989.

[23] Hélène Imbeault, Nathalie Bier, Hélène Pigot, Lise Gagnon, Nicolas Marcotte, Tamas Fulop et Sylvain Giroux.
« Development of a personalized electronic organizer for persons with Alzheimer's disease : the AP@lz ».
Gerontechnology, 9(2), 2010.

[24] Hélène Imbeault, Nathalie Bier, Hélène Pigot, Lise Gagnon, Nicolas Marcotte, Tamas Fulop et Sylvain Giroux.

« Electronic organiser and Alzheimer's disease : Fact or fiction ? ».
Neuropsychological Rehabilitation, 24(1):71–100, 2014.
PMID : 24359438.

[25] Hélène Imbeault, Hélène Pigot, Nathalie Bier, Lise Gagnon, Nicolas Marcotte, Sylvain Giroux et Tamas Fülüp.
« Interdisciplinary Design of an Electronic Organizer for Persons with Alzheimer's Disease ».
Dans Bessam Abdulrazak, Sylvain Giroux, Bruno Bouchard, Hélène Pigot et Mounir Mokhtari, éditeurs, *Toward Useful Services for Elderly and People with Disabilities*, volume 6719 de *Lecture Notes in Computer Science*, pages 137–144. Springer Berlin Heidelberg, 2011.

[26] « ISO 9241 : Exigences ergonomiques pour travail de bureau avec terminaux à écrans de visualisation », 1998.

[27] Melody Y. Ivory et Marti A Hearst.
« The State of the Art in Automating Usability Evaluation of User Interfaces ».
ACM Comput. Surv., 33(4):470–516, décembre 2001.

[28] Nal Kalchbrenner, Edward Grefenstette et Phil Blunsom.
« A Convolutional Neural Network for Modelling Sentences ».
Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, page 655–665, June 2014.

[29] Joke Kort et Henk de Poot.
« Usage Analysis : Combining Logging and Qualitative Methods ».
Dans *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 2121–2122. ACM, 2005.

[30] Hugo Larochelle.
« Réseau neuronaux », 2014.
University Lecture.

[31] Y. LeCun, L. Bottou, Y. Bengio et P. Haffner.
« Gradient-Based Learning Applied to Document Recognition ».
Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[32] Rolf Molich et Joseph S. Dumas.
« Comparative Usability Evaluation (CUE-4) ».
Behav. Inf. Technol., 27(3):263–281, mai 2008.

[33] Rolf Molich, Meghan R. Ede, Klaus Kaasgaard et Barbara Karyukin.
« Comparative usability evaluation ».
Behaviour & Information Technology, 23:65–74, 2004.

[34] Rolf Molich, Robin Jeffries et Joseph S Dumas.
« Making usability recommendations useful and usable ».
Journal of Usability Studies, 2(4):162–179, 2007.

[35] Jacob Nielsen.
Usability Engineering.
Academic Press, 1993.

[36] Jacob Nielsen et Rolf Molich.
« Heuristic evaluation of user interfaces ».
Dans *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM New York, NY, USA, 1990.

[37] Jakob Nielsen.
« Enhancing the Explanatory Power of Usability Heuristics ».
Dans *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 152–158. ACM, 1994.

[38] Robert H. Nielsen.
« Kolmogorov's Mapping Neural Network Existence Theorem ».
Dans *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA), volume III, pages 11–13. Piscataway, NJ : IEEE, 1987.

[39] Janne Pitkänen, Matti Pitkäranta et Marko Nieminen.
« Usability Testing in Real Context of Use : The User-triggered Usability Testing ».
Dans *Proceedings of the 7th Nordic Conference on Human-Computer Interaction : Making Sense Through Design*, NordiCHI '12, pages 797–798. ACM, 2012.

[40] Talya Porat, Alon Schclar et Bracha Shapira.
« MATE : A Mobile Analysis Tool for Usability Experts ».

Dans *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 265–270. ACM, 2013.

[41] Neil Ramsay, Stuart Marshall et Alex Potanin.
« Annotating UI Architecture with Actual Use ».
Dans *Proceedings of the Ninth Conference on Australasian User Interface - Volume 76*, AUIC '08, pages 75–78. Australian Computer Society, Inc., 2008.

[42] F. Rosenblatt.
« The Perceptron : A probabilistic model for information storage and organization in the brain ».
Psychological Review, 65:386–408, 1958.

[43] Stuart Russell et Peter Norvig.
Artificial intelligence : A modern approach.
Prentice Hall, 2002.

[44] Bernard Senach.
« Évaluation ergonomique des interfaces homme-machine : Une revue de la littérature ».
Recherche 1180, INRIA, 1990.

[45] P. Werbos.
« Beyond regression : new tools for prediction and analysis in the behavioral sciences ».
Harvard University, PhD thesis, 1974.