HAL open science

Fuzzy4U : un système en logique floue pour l'adaptation des interfaces utilisateur

Tanguy Giuffrida, Sophie Dupuy-Chessa, Jean-Philippe Poli, Eric Ceret

To cite this version:

Tanguy Giuffrida, Sophie Dupuy-Chessa, Jean-Philippe Poli, Eric Ceret. Fuzzy4U : un système en logique floue pour l'adaptation des interfaces utilisateur. 30e conférence francophone sur l'interaction homme-machine, Oct 2018, Brest, France. pp.52-62, 10.1145/3286689.3286704 . hal-01898998

HAL Id: hal-01898998

https://hal.science/hal-01898998v1

Submitted on 19 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fuzzy4U : un système en logique floue pour l'adaptation des interfaces utilisateur

Fuzzy4U: a fuzzy logic system for user interfaces adaptation

Tanguy Giuffrida
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
38000 Grenoble, France.
tanguy.giuffrida@univ-grenoble-alpes.fr

Sophie Dupuy-Chessa
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
38000 Grenoble, France.
sophie.dupuy@univ-grenoble-alpes.fr

Jean-Philippe Poli
CEA LIST
Laboratoire Analyse de Données et Intelligence Artificielle
91191 Gif-sur-Yvette, France.
jean-philippe.poli@cea.fr

Éric Céret
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
38000 Grenoble, France.
eric.ceret@univ-grenoble-alpes.fr

* Institute of Engineering Univ. Grenoble Alpes

ABSTRACT

User Interfaces adaptation is a well-known requirement in human-computer interaction. However, even if many works address this topic, some challenges such as context uncertainty and combination of adaptation rules, are still remaining. This article tackles these challenges by using fuzzy logic to manage adaptation. It proposes an architecture where an adaptation engine is supported by fuzzy logic. It shows its benefits and compares it with an approach using crisp logic.

CCS CONCEPTS

• **Human-centered computing ~ Human computer interaction (HCI)**

KEYWORDS

HCI adaptation, context of use, fuzzy logic, uncertainty.

RÉSUMÉ

L'adaptation des interfaces est un besoin reconnu en interaction homme-machine. Cependant, même si de nombreux travaux ont abordé cette thématique, il demeure des verrous tels que l'incertitude du contexte d'usage et de la combinaison des règles d'adaptation. Cet article aborde ces verrous en proposant d'utiliser la logique floue pour gérer l'adaptation. Il propose une architecture dans laquelle le moteur d'adaptation s'appuie sur la logique floue. Il en montre les bénéfices et compare l'approche avec une approche en logique booléenne.

MOTS-CLEFS

Adaptation d'IHM, contexte d'usage, logique floue, incertitude.

1 INTRODUCTION

L'adaptation des Interfaces Homme-Machine (IHM) à leur contexte d'usage, caractérisé par le triplet <utilisateur, plateforme, environnement> [1], est un besoin connu. Elle propose de modifier dynamiquement une interface pour qu'elle réponde plus précisément aux besoins et aux attentes de chacun, tout en prenant en compte les contraintes posées par le dispositif d'interaction et l'environnement dans lequel se déroule cette interaction.

De nombreux travaux ont déjà traité des IHM adaptatives [2, 3, 4, 5, 6]. Cependant, des verrous scientifiques demeurent, en particulier si l'adaptation prend en compte l'utilisateur et pas seulement la plateforme et l'environnement [7, 8, 9]. Cela requiert d'abord de pouvoir identifier des caractéristiques nombreuses et variées puis de définir comment les intégrer dans le calcul de la solution. A ces deux niveaux, des difficultés existent.

Tout d'abord, il est nécessaire d'avoir un modèle de contexte fin et donc comprenant un grand nombre de caractéristiques. Rien que pour l'utilisateur, le cluster Virtual User Modelling and Simulation (VUMS) a identifié plus de 250 caractéristiques utiles pour simuler un utilisateur humain [10] cependant cette taxonomie ne couvre pas l'ensemble des besoins, puisqu'elle focalise principalement sur les capacités physiques de l'utilisateur [1, 11].

Un deuxième verrou est l'incertitude entourant les valeurs des caractéristiques. Toutes les valeurs, qu'elles soient captées (ex : imprécision de la capture ou de l'inférence) ou déclarées (ex : différences de perceptions entre utilisateurs, estimations personnelles des valeurs, etc.) ne sont pas certaines. Cette incertitude est habituellement négligée [1, 11, 12, 13, 14].

Une fois le contexte modélisé, il reste à calculer une adaptation qui prenne en compte les valeurs collectées ou déclarées. Classiquement, ce calcul se fait à l'aide de règles d'adaptation. On trouve différentes approches pour exprimer ces règles, comme les transformations de modèles dans les approches dirigées par les modèles [15], les patrons de solutions [11] ou le développement de solutions multiples précalculées dont l'une est choisie selon des règles logiques [16]. Au regard du nombre de caractéristiques potentielles, l'adaptation des IHM repose sur des règles parfois complexes mais surtout nombreuses, ce qui pose un problème de combinaison des règles. Un système plus intéressant que le classique choix d'une règle parmi un ensemble de règles possibles, par exemple à l'aide de poids ou de priorités, serait de parvenir à un

compromis entre toutes les règles applicables. Le grand nombre de solutions potentielles ouvre alors un nouveau problème qui est celui de l'explosion combinatoire [17] ou autrement dit, la difficulté à gérer la multiplicité des compromis possibles et à faire un choix parmi ceux-ci. Cette combinatoire pouvant conduire à des problématiques de performance [18].

Dans cet article, nous nous intéressons à la logique floue, qui a les capacités d'aborder ces verrous. Nous proposons d'exprimer les règles d'adaptation dans ce formalisme. L'utilisation d'un moteur d'inférence floue permet de gérer automatiquement la combinaison des règles et de prendre en compte l'incertitude du contexte. Cet article détaille notre approche basée sur la logique floue dans le calcul des adaptations et présente une analyse comparative des résultats d'adaptation obtenus et ceux issus d'une approche plus classique en logique booléenne, accompagnés des avis d'experts.

Cet article est structuré de la façon suivante : dans la section 2, nous présentons l'état actuel des recherches en adaptation des IHM. La section 3 introduit les concepts fondamentaux de la logique floue. La section 4 présente l'architecture du démonstrateur et son extension à la logique floue avec les bénéfices attendus. La section 5 décrit la base de règles floues, leur comparaison avec les règles booléennes préexistantes et présente l'avis d'experts à ce sujet. L'article se termine par un résumé du travail et des perspectives.

2 ETAT DE L'ART

Afin de mieux situer notre contribution, nous présentons dans cette section une analyse de travaux similaires.

2.1 Travaux existants

L'état de l'art se base sur l'étude de deux types de travaux : les travaux relatifs à l'adaptation des IHM et ceux utilisant la logique floue. Il existe de nombreux travaux traitant de l'adaptation au contexte d'usage. Nous ne présentons ici que les travaux les plus représentatifs.

CAMELEON [1] est une approche de référence, basée sur les modèles et permettant la génération dynamique d'IHM adaptée à leur contexte d'usage. Il se base sur un processus de réification autour de 4 niveaux d'abstractions : modèle de tâches, IHM abstraite, IHM concrète et enfin IHM finale. Théoriquement, toutes les caractéristiques du contexte peuvent être prises en compte. Rien ne mentionne l'incertitude de leurs valeurs. Les règles d'adaptation sont des transformations de modèles appliquées conditionnellement selon le contexte d'usage, qui doivent être deux à deux exclusives, ce qui est particulièrement complexe à obtenir, par exemple lorsque deux caractéristiques devraient induire des adaptations antagonistes.

[7] propose quant à lui un système nommé « Adaptation Integration System », basé sur Cameleon. Ce système vise à intégrer des exigences d'accessibilité pour des personnes en situation de handicap en incluant les règles d'adaptation dans le processus de développement. Ce travail utilise des caractéristiques peu nombreuses et prédéfinies.

Maria [12] est un UIDL basé sur les modèles et permettant la génération dynamique d'IHM adaptée aux différentes plateformes utilisées par l'utilisateur au cours d'un processus de migration. L'article se focalise donc sur l'aspect Plateforme du contexte d'usage.

Cedar [13] fournit une architecture facilitant le développement d'IHM adaptatives au sein de progiciels, basé sur des simplifications en fonction de rôle. Cedar permet ainsi d'avoir une adaptation efficace, mais dans un cadre contrôlé.

[14] définit une architecture en trois couches (3-layer architecture) pour la conception d'IHM adaptative. L'adaptation se base sur le contexte d'usage et vise les environnements intelligents. Les informations utilisées sont directement lues à partir des capteurs, sans gestion de l'incertitude. Le processus d'adaptation se limitant aussi à un cadre défini, avec des « situations types ».

MyUI [11] est une infrastructure de développement d'IHM visant à améliorer l'accessibilité par le biais d'une adaptation à l'utilisateur. Les règles d'adaptation sont ici gérées grâce à des patrons. Ces travaux listent un grand nombre de caractéristiques utilisateurs. Cependant, ces caractéristiques sont principalement déclarées, et MyUI utilise des seuils fixes pour définir les différents états d'une caractéristique. En termes de règles, MyUI se focalise sur l'amélioration de l'accessibilité en fonction de certains handicaps.

Supple [19] prend en charge la génération automatique d'IHM adaptées aux capacités, aux appareils, aux tâches, et aux préférences de chaque utilisateur. Les capacités visuelles et motrices sont les principales caractéristiques utilisateurs prises en charge lors de l'adaptation et 40 facteurs d'interface utilisateur (par exemple, la taille de police) sont supportés afin d'optimiser la disposition des éléments de l'IHM. Il a été noté dans [11] que les performances de Supple sont problématiques car elles excèdent des temps raisonnables ; cela serait donc d'autant plus vrai en cas de combinaison complexe des règles.

[18] se penche sur l'utilisation du Machine Learning pour adapter des IHM au contexte d'usage, principalement dans le but d'optimiser les interactions des utilisateurs et de limiter leurs erreurs. Le machine learning permet de gérer des données complexes et floues, et calculer des inférences complexes à partir de celles-ci. Cependant, le machine learning nécessite de recueillir une base de connaissances conséquente avant de pouvoir proposer des adaptations améliorant effectivement l'expérience utilisateur. De plus, cet article présente des cas d'applications potentiels, mais aucun n'a réellement été mis en pratique.

Il existe aussi des travaux intégrant la logique floue pour supporter des systèmes adaptatifs.

FSAM [20] est un outil basé sur la logique floue permettant de choisir le service le plus adapté à un utilisateur en fonction de son contexte d'usage. Ces travaux se restreignent à un choix entre des services prédéfinis et n'abordent pas les questions de l'interaction homme-machine.

[21] propose une approche en logique floue pour le développement d'applications mobiles adaptatives, supportant la diversité des plateformes mobiles (hardware, OS, API, etc.) et les capacités de chacune de ces plateformes en termes de performance. L'utilisateur n'est pas pris en compte, limitant de fait le nombre de caractéristiques et leur incertitude.

[22] présente un outil utilisé pour développer des applications par des conducteurs ou passagers de voiture au travers d'une interface vocale. La logique floue est utilisée afin de calculer le niveau d'expertise et de concentration de l'utilisateur : débutant, intermédiaire ou avancé. Chaque niveau entraînant un niveau plus ou moins élevé d'options et d'informations fournies. Les règles floues utilisées sont au nombre de 7, elles se basent sur 10 caractéristiques (nombre d'applications créées ; nombre de demande d'aide ; nombre de jurons…) afin de modifier une sortie : le niveau d'expertise. Cette utilisation de la logique floue ne

correspond donc pas à ce que nous souhaitons faire en termes de règle d'adaptation.

[23] présente une autre approche en logique floue, pour l'adaptation de page web afin d'améliorer l'expérience utilisateur. Elle se base cependant sur des cas d'utilisation dont elle en présente 7 (ex : Passer une commande, Effectuer une recherche, …). Les données sont inférées à partir des comportements utilisateurs, limitant de fait la taille du modèle utilisateur et donc les règles d'adaptation possibles. Il n'aborde donc pas la question de la combinaison des règles qui nous semble centrale.

[24] propose un mécanisme d'adaptation d'IHM basé sur la logique floue et prenant en considération trois facteurs cognitifs des utilisateurs définis par [25] : leur dimension « Verbal/Imager », leur dimension « Analyst/Wholist », et leur capacité de mémoire de travail. Chacun de ces aspects ayant des implications particulières en termes de design. Ces travaux ne gèrent donc qu'un nombre restreint de caractéristiques, et l'approche comprend une gestion manuelle de la composition des règles, ce qui limite les compétences du système et sa capacité à prendre en charge des situations inattendues.

Enfin [26] présente une approche pour la personnalisation d'IHM grâce à la logique floue se basant sur des relations sémantiques entre composants d'IHM et préférences utilisateurs afin de sélectionner les composants les plus adaptés. Le cas d'étude présenté ne donne qu'une seule règle, ayant deux entrées et une sortie, et la combinatoire entre les entrées a été gérée manuellement au sein de la règle. Ce travail ne prend donc pas en compte des aspects qui nous semblent importants tels que l'étendue du contexte d'usage et la combinaison des règles.

2.2 Analyse

Notre analyse des travaux existants s'appuie sur des critères relatifs aux difficultés identifiées pour réaliser un moteur d'adaptation. Ils concernent donc : 1) l'étendue du contexte d'usage pris en compte en termes de couverture des 3 aspects (utilisateur, plateforme, environnement) ainsi que du nombre de caractéristiques abordées, 2) la gestion d'une incertitude au sein des valeurs des caractéristiques du contexte d'usage, 3) la complexité et 4) la combinaison des règles d'adaptation.

Le tableau 1 regroupe les différents travaux cités et les évalue au regard de nos critères avec des boules de Harvey.

Les critères considérés permettent de regrouper les travaux allant de Caméléon à Supple, listés en début de tableau. En effet, ceux-ci traitent tous d'adaptation dynamique en fonction d'un contexte d'usage plus ou moins conséquent et comprenant un nombre plus ou moins élevé de caractéristiques. Cependant, ces travaux ne se penchent pas sur le reste de nos critères que sont l'incertitude, la complexité ou la combinaison des règles.

Par ailleurs, les travaux [20, 21, 22, 23, 24, 26] utilisent la logique floue et prennent bien en compte l'incertitude des valeurs. Dans les travaux [20, 21], la composition des règles est gérée par la logique floue, mais, de fait, l'absence de prise en compte d'un contexte d'usage conséquent limite le problème. Aucun de ces travaux n'aborde vraiment la question de la combinaison des règles.

Enfin, les travaux de Genaro Motti [18] ont les meilleures notes globales, avec 3 de nos 4 critères gérés. Cependant, l'utilisation du Machine Learning nécessite de disposer d'une base de connaissance importante – et donc difficile à acquérir – avant de pouvoir effectuer des adaptations correspondant effectivement aux besoins des utilisateurs.

Approches	Etendue du contexte d'usage	Incertitude	Complexité	Combinaison des règles
Cameleon [1]	●	○	○	◐
Minon [7]	●	○	○	◐
MyUI [11]	●	○	○	◐
Maria [12]	◐	○	○	○
Cedar [13]	◐	○	○	○
3-layer architect. [14]	●	○	○	○
Supple [19]	◐	○	○	○
FSAM [20]	◐	●	○	◐
Desruelle [21]	○	●	○	◐
Cueva-Fernandez [22]	○	●	○	○
Nyongesa [23]	◐	●	○	○
Papatheocharous [24]	○	●	○	○
Soui [26]	○	●	○	○
Genaro Motti [18]	●	●	○	●

● : traité, ◐ : partiellement traité, ○ : non traité

Tableau 1 : Comparaison des travaux de l'Etat de l'Art en fonction des critères ciblés

Afin de dépasser les limites des travaux actuels et de proposer une solution qui permette de gérer l'incertitude du contexte et de nombreuses caractéristiques et règles d'adaptation, nous nous intéressons à la réalisation d'un système d'adaptation des IHM basé sur la logique floue. Pour aller au-delà des propositions existantes en logique floue, nous montrerons son intérêt pour deux aspects peu abordés jusqu'alors qui sont la complexité des règles et la combinaison des règles d'adaptation.

3 LOGIQUE FLOUE

Afin de comprendre l'utilisation de la logique floue pour l'adaptation des IHM, cette section présente les principes de cette logique et son processus d'inférence.

3.1 Systèmes experts flous

Les systèmes experts flous sont des algorithmes d'intelligence artificielle capables de reproduire des raisonnements à partir de règles basées sur la logique floue. La logique floue peut être vue comme une extension de la logique booléenne classique qui permet de prendre en compte l'incertitude et l'imprécision : alors que la logique classique ne considère que deux valeurs (« vrai » et « faux »), la logique floue considère un ensemble infini de valeurs réelles entre 0 et 1. Cette logique a été introduite par Zadeh [27] afin de se rapprocher du raisonnement humain et d'éviter les effets de seuil des opérateurs de comparaison. La logique floue permet également d'écrire des règles plus proches du langage naturel.

3.2 Concepts de sous-ensemble flou et de variable linguistique

La logique floue est basée sur le concept de sous-ensemble flou. Soit X l'univers du discours, c'est-à-dire un ensemble d'objets dénotés x. Un sous-ensemble flou A de X est totalement caractérisé par une fonction appelée « fonction d'appartenance ». Cette fonction indique à quel point un objet x appartient à X, alors que dans les ensembles « *crisp* » (le terme « *crisp* » se définit par

opposition à flou), les objets appartiennent ou non à l'ensemble. Les sous-ensembles flous sont donc utiles pour représenter l'imprécision des entrées ou l'incertitude liée aux connaissances mise en œuvre. La figure 1 représente un sous-ensemble flou défini sur [0;10] avec une fonction d'appartenance triangulaire qui représente « approximativement 5 ».

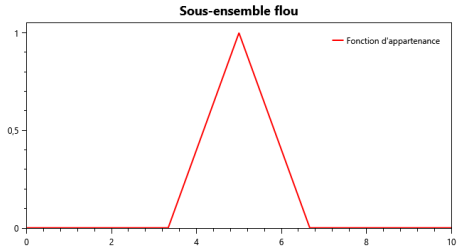

Figure 1 : Exemple de sous-ensemble flou

Les fonctions d'appartenance peuvent prendre différentes formes : triangulaire, trapézoïdale, Gaussienne, etc.

Le concept suivant dérive des sous-ensembles flous et s'appelle « variable linguistique » [28], qui est un triplet (V, X, T_V) défini par: 1) V, son nom ; 2) X, le domaine sur lequel elle est définie ; 3) T_V, une collection finie de sous-ensembles flous appelés « termes », qui qualifient V et qui sont nommés (ex. : « froide », « chaude »…).

La figure 2 représente une variable linguistique appelée « température » définie sur le domaine [-20;50] et composée de trois termes (« froide », « moyenne » et « chaude ») et leurs fonctions d'appartenance associées.

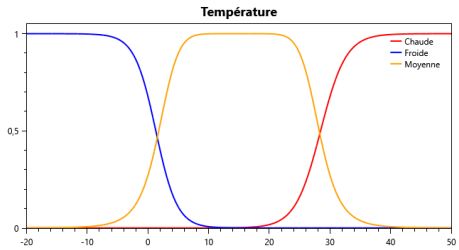

Figure 2 : Exemple de variable linguistique Température

3.3 Utilisation de règles floues pour modéliser les connaissances

Les définitions précédentes sont utilisées pour définir les connaissances d'experts humains dans un ensemble de règles. Ces règles sont basées sur une combinaison de propositions floues élémentaires. Une proposition floue élémentaire est la définition de « *V est A* » à partir d'une variable linguistique (V, X_V, T_V) où A est un terme de T_V. Ainsi, si l'on reprend la variable linguistique « température » de la figure 2, une proposition floue pourrait être « *température est froide* ». Cette proposition est évaluée à partir de la fonction d'appartenance du terme impliqué pour une température t donnée. Cette valeur est floue et donc comprise entre [0, 1] contrairement à la logique classique dans laquelle l'évaluation d'une proposition est soit vraie (1) soit fausse (0).

Une expression floue est une composition de propositions floues élémentaires ou d'autres expressions à l'aide d'opérateurs logiques : négation (non), conjonction (et) et disjonction (ou), le XOR n'étant pas géré.

Les propositions floues sont des cas particuliers d'expressions floues. Par exemple, soit deux propositions floues « *V est A* » utilisant (V, X_V, T_V) et « *W est B* » utilisant (W, X_W, T_W). Alors, « *V est A et W est B* », « *V est A ou non W est B* » sont des expressions floues. La valeur de vérité des expressions floues est obtenue en appliquant les opérateurs sur les valeurs de vérité de leurs opérandes.

Ces expressions sont donc utilisées pour exprimer les connaissances des experts humains dans les règles floues. Une règle floue est composée d'une prémisse (ou antécédent) et d'une conclusion (ou conséquence) et est de la forme « SI prémisse ALORS conclusion ».

Une prémisse est une expression floue alors qu'une conclusion est une déclaration floue de nature différente.

Une conclusion peut être une proposition floue. Dans ce cas, la règle est de la forme : *« SI température est froide ALORS le thermostat est élevé. »* Une conclusion peut être une fonction mathématique des entrées de la prémisse. Ces règles sont dites de la forme Takagi-Sugeno et sont de la forme *« SI température est froide ALORS thermostat=f(température) ».*

Ainsi les règles associent les entrées du système aux sorties à l'aide d'une implication floue (dénotée par le ALORS).

3.4 Inférence floue

Maintenant que le formalisme utilisé pour représenter les connaissances a été présenté, nous pouvons nous intéresser au processus d'inférence, c'est-à-dire le calcul des sorties. En effet, dans la section précédente nous avons vu comment une règle fonctionnait ; nous allons à présent voir comment les règles fonctionnent ensemble. Il existe deux types d'inférence en logique floue : nous nous limitons dans cet article à l'utilisation et la description du processus d'inférence de Mamdani [29], résumé dans la figure 3.

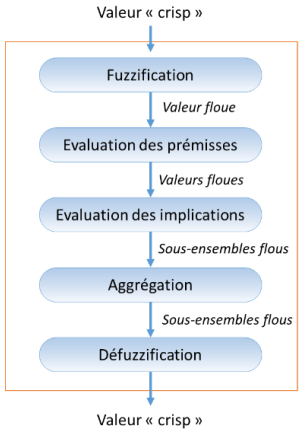

Figure 3 : Vue d'ensemble de l'inférence de Mamdani

La première étape consiste à évaluer chacune des propositions floues élémentaires présentes dans les prémisses des règles : cette étape est appelée fuzzification. Elle prend les valeurs « crisp » des entrées et associe une valeur floue en fonction des fonctions d'appartenance des termes. Par exemple, en utilisant la variable linguistique « température » de la figure 2 et le terme « moyenne », pour une température de 15°C, « *température est moyenne* » est évaluée à 1. Pour une température de 1°C, « température est froide » et « température est moyenne » valent 0.5 (approximativement).

Les prémisses sont ensuite calculées. La valeur des prémisses devient alors la valeur d'activation de la règle. Par exemple, si on prend les opérateurs de Zadeh, pour une température de 15°C, la

valeur de la prémisse « température est froide ou température est moyenne » est *max*(1,0)=1.

La fonction d'implication est ensuite appliquée à la conclusion des règles. Le résultat est un sous-ensemble flou qui peut être interprété comme une distribution des valeurs pour les sorties. La figure 4 montre le sous-ensemble flou résultant de l'application d'une implication *minimum*, lorsque la règle est activée à 0,75.

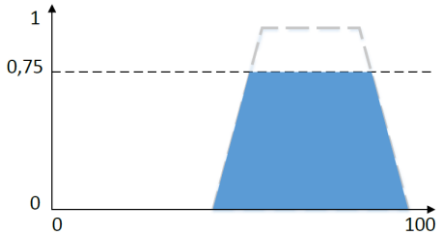

Figure 4 : Résultat de l'implication

Les sous-ensembles flous des différentes règles concernant la même sortie sont ensuite agrégés par une méthode particulière, très souvent la fonction *maximum* (ex en figure 5).

Enfin, le sous-ensemble flou de chaque sortie est transformé en valeur « crisp » : cette étape est appelée défuzzification. Il existe différentes fonctions de défuzzification ; la plus connue consiste à calculer le barycentre de l'aire sous la courbe (figure 5).

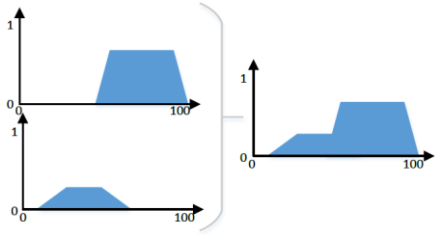

Figure 5 : Application de l'agrégation maximum à deux sous-ensembles flous

4 Le système Fuzzy4U

Dans cette section, nous montrons les apports de la logique floue au sein d'un système d'adaptation des IHM. Nous commençons par présenter l'architecture de ce système avant de discuter de l'intégration de la logique floue dans son moteur d'adaptation.

4.1 Présentation de l'architecture

Le système Fuzzy4U s'appuie sur une architecture proposée dans notre équipe pour l'adaptation des IHM (figure 6). Elle comprend principalement 3 composants : un moteur d'inférence, un moteur d'adaptation et le système interactif.

Le moteur d'inférence est en charge de déduire dynamiquement la valeur du contexte d'usage à partir de données captées. Par exemple, le moteur d'inférence peut se baser sur le nombre de clics « valides » (ie : effectués sur des interacteurs tels que liens, boutons, liste, …), ainsi que le nombre de clics « invalides » (à côté de tout interacteur) afin de déterminer si l'utilisateur a une bonne précision de la main.

D'autres données peuvent être déclarées par l'utilisateur à travers une méta-UI c'est-à-dire une interface regroupant un ensemble de caractéristiques que l'utilisateur peut manipuler pour contrôler l'adaptation, telles que ses caractéristiques utilisateurs (dyslexie, daltonie, acuité visuelle, etc.) ou encore directement des caractéristiques de l'interfaces (taille de police, espacement des mots, etc.). La méta-UI se limite ici aux caractéristiques de l'utilisateur et ne permet pas de visualiser ou de modifier les règles d'adaptation à appliquer.

Les données inférées et déclarées sont envoyées au moteur d'adaptation. En cas de données inférées et déclarées portant sur la même caractéristique, ce seront les données fixées par l'utilisateur (donc déclarées) qui seront prises en compte.

Le moteur d'adaptation va ensuite se servir des données, ainsi que d'une base de règles d'adaptation afin de sélectionner une version adéquate de l'interface (nommée un variant) sur laquelle des paramètres de présentation (couleurs, police de caractères…) seront appliqués. Par exemple, si la précision de la main de l'utilisateur est faible, le moteur d'adaptation peut augmenter l'espace pris en compte pour le clic sur les interacteurs afin d'améliorer l'utilisabilité de l'interface pour cet utilisateur spécifique. Le moteur d'adaptation s'appuyait initialement sur des règles en logique booléenne.

Le moteur d'inférence est distinct du moteur d'adaptation dans un but de séparation des préoccupations : le premier est chargé de transformer des valeurs brutes (par exemple, 130 clics sur un interacteur et 212 dans une zone vide) en valeurs utilisables par le second (précision de la main « très faible »).

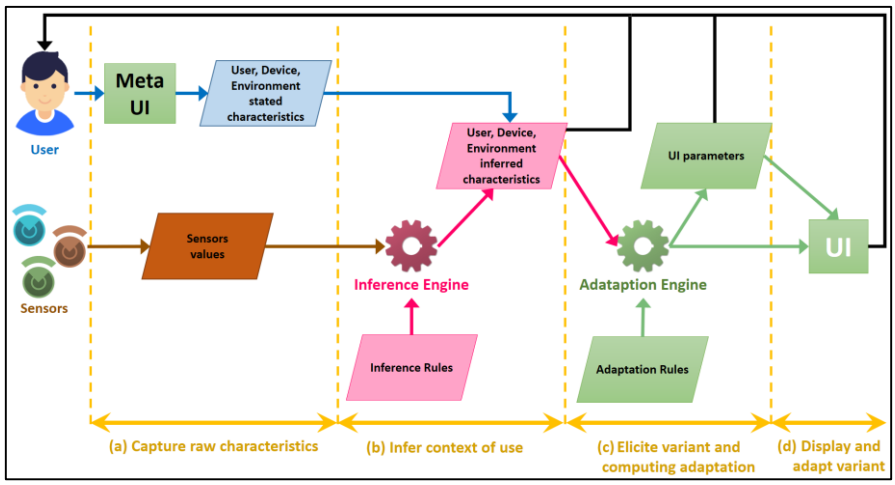

Figure 6 : Architecture logique utilisée par l'équipe pour l'adaptation d'IHM

4.2 Intégration de la logique floue

L'originalité de notre travail réside dans le fait de proposer un moteur d'adaptation en logique floue en complément d'un moteur déjà existant en logique booléenne. Dans les sous-sections suivantes, nous allons voir comment les règles d'adaptation peuvent s'exprimer en logique floue. Nous mettrons en avant comment la logique floue a le potentiel pour aider à résoudre les verrous que sont l'incertitude, la complexité des règles, ainsi que leur combinaison.

4.2.1 Incertitude

Toutes les caractéristiques ont un degré d'incertitude, qu'elles soient inférées par le moteur (ex : imprécision du calcul de la précision des mains grâce à un ratio entre clics valides et clics invalides) ou déclarées par l'utilisateur (ex : deux utilisateurs ayant une acuité visuelle identique pourraient chacun la définir différemment).

En logique booléenne, aucune gestion de l'incertitude n'est présente, ni dans la définition des valeurs des variables, ni, par conséquent, dans les règles d'adaptation. Par exemple, 4 valeurs disjointes (normal, low, veryLow et extraLow) sont définies pour l'acuité visuelle et une règle se basant sur celle-ci peut avoir la forme suivante (les valeurs nominales sont ici converties en valeurs numériques sur le domaine [0;4]) :

```
if        visualAcuity <= 1  then fontSize = 12 ;
else if   visualAcuity <= 2  then fontSize = 14 ;
else if   visualAcuity <= 3  then fontSize = 16 ;
else      fontSize = 18 ;
```

La règle ci-dessus met en place une adaptation plus ou moins importante en fonction de l'acuité visuelle de l'utilisateur, divisée en 4 états aux seuils de valeurs 1, 2 et 3 : de 0 à 1 pour une acuité visuelle normale, de 1 à 2 pour une acuité visuelle légèrement réduite, etc. Cependant, cette règle traite de la même façon une valeur d'entrée de 0 ou de 1 : en logique booléenne, toutes les valeurs sont traitées comme ayant la même certitude au sein d'un même état.

Nous allons pouvoir utiliser les fonctions d'appartenance des variables en logique floue pour exprimer l'incertitude des différentes valeurs. Par exemple, la figure 7 montre la définition de la variable acuité visuelle : la valeur est définie en abscisses (ici de 0 à 4), et la certitude des 4 différents états (normal, low, veryLow et extraLow) est spécifiée en ordonnées. L'état « normal » est ainsi certain lorsque que l'acuité visuelle varie de 0 à 0.5. A partir de 0.5 et jusqu'à 1.5, la certitude de l'état « normal » décroît au profit de l'état « low » qui gagne en certitude. A 1.5, l'état « low » a donc 100% de certitude alors que l'état « normal » est à 0%. Le croisement des courbes (aux abscisses 1, 2 et 3) indique donc une probabilité de 50% pour les deux états.

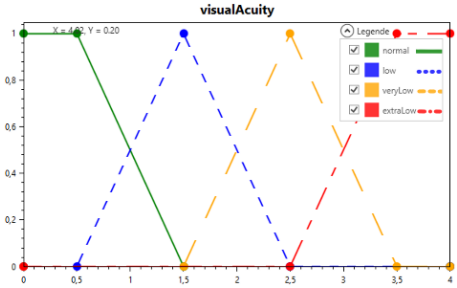

Figure 7 : Définition de la variable linguistique visualAcuity

L'introduction de cette incertitude permet de limiter les effets de seuils qui induisent un passage brutal d'une valeur de contexte à une autre. Elle est particulièrement utile dans la gestion des règles pour lesquelles plusieurs variables d'entrée influent sur une même variable de sortie, comme le montre la section 4.2.3.

4.2.2 Complexité

Outre la prise en compte de l'incertitude, des difficultés sont liées à l'expression des règles d'adaptation. Certaines règles disposant de nombreux seuils peuvent devenir assez lourdes en logique booléenne.

Prenons comme exemple la règle se basant sur le niveau d'acuité visuelle de l'utilisateur afin d'adapter la taille de la police de caractères, présentée en section 4.2.1. Comme nous l'avons vu, plusieurs seuils sont nécessaires (ici 1, 2, 3 et 4) et la granularité finale de la variable de sortie est relativement faible (ici 4 valeurs possibles : 12, 14, 16, 18). Cette granularité peut être améliorée, mais au détriment du nombre de seuils.

En logique floue, l'implémentation de cette règle comprend la définition des variables linguistiques d'entrée et de sortie (figure 8), et la création des règles en logique floue (figure 9).

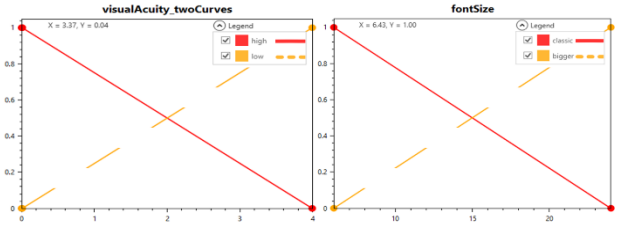

Figure 8 : Définition des variables linguistiques visualAcuity et fontSize

```
If  visualAcuity_fontSize  is  high   then  fontSize  is  classic
If  visualAcuity_fontSize  is  low    then  fontSize  is  bigger
```

Figure 9 : Règles floues définies pour l'adaptation de la taille de police en fonction de l'acuité visuelle

Compte tenu des définitions des domaines de variables, la règle en logique floue est plus simple et ne comporte que deux cas.

En utilisant la méthode de défuzzification des barycentres pondérés (ou Centre de Gravité), la valeur de sortie fontSize est comprise entre 12 et 18 points. Son principe est de calculer le barycentre du sous-ensemble flou de sortie pour chacune des règles, et d'en faire une somme pondérée par l'aire sous la courbe. En considérant toutes les règles ayant une même sortie, la valeur défuzzifiée peut s'écrire :

$$d(sortie) = \sum_i C_i \times A_i \; iAi$$

où Ci est la valeur du barycentre et Ai l'aire sous la courbe pour la $i^{ème}$ règle concernant cette sortie.

Cette défuzzification est utilisable sur des variables linguistiques continues et nous avons pu l'adapter aux variables discrètes car dans notre cas, les domaines discrets sont ordonnés ce qui nous permet de donner du sens au barycentre. Dans ce cas, l'aire sous la courbe est simplement remplacée par la valeur correspondante dans l'histogramme qui sert de fonction d'appartenance.

Afin de comprendre le principe de cette méthode, deux exemples sont donnés en figures 10 et 11.

Sur la figure 10, on peut voir que pour une acuité visuelle à 0 en entrée, le terme « high » est activé à 100% et le terme « low » à 0%. Selon la première règle, l'activation du terme de sortie « classic »

se fait lui aussi à 100%, représenté par l'aire rouge. Le barycentre de ce triangle rectangle est situé au premier tiers de sa base, soit une valeur de 12 pour la variable de sortie fontSize ayant un domaine entre 6 et 24.

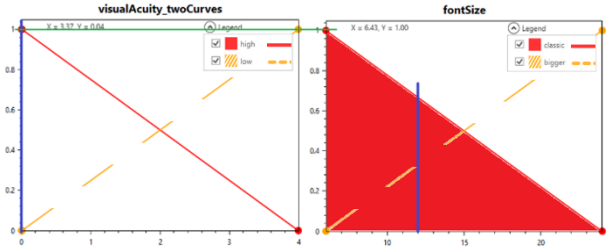

Figure 10 : Schéma explicatif de la méthode de défuzzification des barycentres pondérés pour une valeur d'entrée de 0

Pour une acuité visuelle à 1 en entrée (figure 11), le terme « high » est cette fois activé à 75% et le terme « low » à 25%. Cela est représenté par les deux lignes vertes et l'activation du terme de sortie « classic » à 75%, représenté par l'aire rouge et l'activation du terme de sortie « bigger » à 25%, représenté par l'aire orange. Les barycentres de ces deux quadrilatères sont ensuite calculés (12.3 pour le barycentre lié à « classic » et 10.071 pour celui lié à « bigger »), puis pondérés en fonction de leurs aires (respectivement 84.375 et 39.375), afin d'obtenir le résultat final de 13.5 points pour la variable fontSize. Nous avons ainsi l'état « bigger » qui a pris un peu plus de poids comparé à l'état « classic », donnant une taille de police de 13.5 au lieu de la valeur 12 du cas précédent. Ces résultats sont donc continus et suivent une fonction « fontSize = 12 + visualAcuity * 1.5 ».

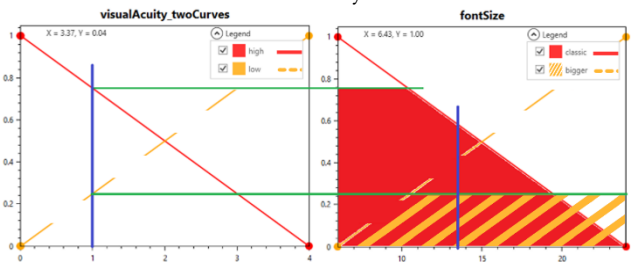

Figure 11 : Schéma explicatif de la méthode de défuzzification des barycentres pondérés pour une valeur d'entrée de 1

Cet exemple montre comment la logique floue évite les effets de seuil. Elle permet de simplifier l'expression des règles d'adaptation tout en permettant d'obtenir des valeurs de sortie plus précises.

4.2.3 Combinaison des règles

Le dernier verrou à aborder est la combinatoire des règles qui peut survenir lorsque plusieurs variables d'entrées peuvent influer sur une même variable de sortie [17]. Par exemple, il peut être nécessaire d'augmenter l'espace cliquable autour des interacteurs soit quand l'acuité visuelle est faible soit quand la précision des clics (et donc des mains) est faible. Un problème se pose quand l'une des deux variables d'entrée est forte alors que l'autre est faible.

En logique booléenne, il n'y a pas de protocole permettant de définir aisément la marche à suivre dans ce cas. Il est possible de choisir l'une ou l'autre des valeurs, auquel cas il faut donc établir un système de pondération ou de priorisation des règles ; ou de faire une moyenne des valeurs, mais cela n'est valable que pour des valeurs continues et non discrètes.

La logique floue a l'avantage de gérer cela de manière intrinsèque, notamment par le biais de la gestion de l'incertitude : sans avoir un poids différent, une certitude plus importante a plus d'impact. Il reste toutefois possible de spécifier des poids au besoin.

Considérons l'exemple de la taille de la zone de clic des interacteurs, nommée ici clickCorrection. Elle se base sur l'acuité visuelle (figure 7) et la précision des mains (dont la forme du graphique est identique à celui de la figure 7) de l'utilisateur afin de définir un niveau plus ou moins important de correction de la taille ; la variable linguistique de sortie, clickCorrection, étant composée de 4 valeurs discrètes représentant 4 degrés de correction allant de noCorrection à highCorrection.

En logique booléenne, les règles utilisées pour le calcul de la variable clickCorrection sont les suivantes :

```
   if    handPrecision <= 1 then clickCorrection = noCorrection
elsif    handPrecision <= 2 then clickCorrection = lowCorrection
elsif    handPrecision <= 3 then clickCorrection = mediumCorrection
else     clickCorrection = highCorrection
   if    visualAcuity <= 1 then clickCorrection = noCorrection
elsif    visualAcuity <= 2 then clickCorrection = lowCorrection
elsif    visualAcuity <= 3 then clickCorrection = mediumCorrection
else     clickCorrection = highCorrection
```

En logique floue, les règles sont listées dans la figure 12.

```
If handPrecision is normal then clickCorrection is noCorrection
If handPrecision is low then clickCorrection is lowCorrection
If handPrecision is veryLow then clickCorrection is mediumCorrection
If handPrecision is extraLow then clickCorrection is highCorrection
If visualAcuity is normal then clickCorrection is noCorrection
If visualAcuity is low then clickCorrection is lowCorrection
If visualAcuity is veryLow then clickCorrection is mediumCorrection
If visualAcuity is extraLow then clickCorrection is highCorrection
```

Figure 12 : Règles floues définies pour le calcul du clickCorrection en fonction de la précision des mains et de l'acuité visuelle

Nous pouvons constater que la formulation des règles est similaire dans les deux logiques, la différence résidant dans le fonctionnement interne de chacune des logiques.

Les 4 premières règles de chaque logique se basent sur l'état de la variable handPrecision – normal, low, veryLow, extraLow – et fournissent une valeur de sortie pour clickCorrection – allant respectivement de noCorrection à highCorrection. Les 4 dernières règles sont identiques à l'exception de la variable d'entrée qui est cette fois visualAcuity. On peut noter qu'il n'est pas nécessaire dans l'expression de ces règles de se soucier de leur éventuelle contradiction ou de leur combinaison ; en logique floue, la valeur de sortie est déterminée par la défuzzification.

Dans l'exemple de la correction de la taille des interacteurs pour obtenir une transition lissée d'une valeur à l'autre des paramètres de sortie, nous avons opté pour la défuzzification dite des barycentres pondérés (cf. sous-section 4.2.2). Ainsi, avec une acuité visuelle à 0 et une précision des mains à 4 en entrée, le résultat est « lowCorrection ». Alors qu'une acuité visuelle à 1 et une précision des mains à 4 en entrée nous donneront une sortie « mediumCorrection ».

La logique floue gère donc la combinaison des règles d'adaptation par la défuzzification sans avoir à mettre en place un algorithme spécifique de priorisation.

5 ANALYSE COMPARATIVE

Les règles d'adaptation en logique floue semblent avoir des avantages par rapport à une approche classique en logique booléenne. Afin de valider ces avantages, nous avons mené une analyse comparative entre les 2 types de règles, accompagnée de l'avis d'experts.

5.1 Méthode d'analyse

L'analyse comparative des 2 types de règles se base sur l'expression et l'analyse des résultats de règles issues majoritairement de la littérature [11, 30, 31, 32, 33, 34], mais aussi de notre expérience (clickCorrection par exemple).

Les règles sont implémentées dans deux modules déterminant l'adaptation nécessaire, l'un fonctionnant en logique floue et l'autre en logique classique, tous deux intégrés dans l'architecture décrite en section 3.4. Un paramètre de l'application permet de choisir quelle logique est utilisée. Dans cette étude, l'architecture a été implantée avec des technologies Web et repose sur NodeJS, permettant ainsi d'exécuter du JavaScript aussi bien côté serveur que côté client. En particulier, les objets respectent des structures similaires des deux côtés et sont stockés dans des fichiers textes au format JSON sur le serveur, permettant de conserver cette structure commune tout en autorisant une modification manuelle simple. Le modèle de contexte et le modèle utilisateur en particulier sont réalisés selon ces caractéristiques. Le module en logique booléenne est exécuté en interne, alors que, pour le moteur en logique floue, nous utilisons l'outil ExpressIF développé par CEA Tech, qui est appelé grâce à un Web Service, via une requête HTTP. Les tests vont consister à faire varier les valeurs du contexte d'usage en entrée afin de mettre en évidence les différences de résultats entre les deux.

Nous utilisons 14 entrées concernant l'utilisateur et 4 autres entrées pour décrire l'environnement (matériel, paramètres exogènes). Les entrées qui concernent l'utilisateur décrivent sa capacité à manipuler des interfaces modernes (précision du toucher, etc.), son âge, l'éventualité d'une dyslexie, etc. Les entrées environnementales décrivent le matériel utilisé en termes d'orientation, luminosité et taille d'écran. Nous supposons qu'il est possible d'avoir des valeurs pour ces entrées que ce soit grâce à des capteurs ou par saisie. Ces entrées sont ensuite liées par des règles d'adaptation à 20 paramètres de sortie permettant d'agir sur l'interface, comme par exemple l'espacement entre les lettres ou les mots, la présence ou non d'animations, les tailles de police, etc. Les règles d'adaptation sont au nombre de 34 en logique booléenne, et 81 en logique floue. Cette différence est due à la façon dont les règles ont été implémentées en logique floue. Par exemple, les règles présentées dans la figure 12 sont au nombre de 8 en logique floue, mais elles se regroupent en 2 règles en logique booléenne, par le biais du format « if /else if / else ».

5.2 Résultats obtenus

Par manque de place, nous ne pouvons présenter ici l'ensemble des résultats obtenus. Nous synthétisons dans le tableau 2 des résultats représentatifs qui diffèrent entre logique booléenne et floue. Par exemple, la règle permettant d'adapter la taille de la police en fonction de l'acuité visuelle est élémentaire, mais permet d'apporter une meilleure granularité au résultat, justifiant son intégration au tableau. Le tableau peut se lire de la manière suivante.

La première colonne du tableau ci-dessous, Situations d'entrée, est subdivisée en deux sous-colonnes. La 1ère sous-colonne va détailler un ensemble de plusieurs caractéristiques du contexte d'usage impactant une même variable de sortie. Chaque caractéristique ayant sa propre règle d'adaptation. La 2ème sous-colonne définit quant à elle des exemples de valeurs correspondant respectivement à chacune des variables d'entrées.

Par exemple, la 1ère situation d'entrée du tableau correspond à un utilisateur ayant les caractéristiques suivantes : ictExperience à 1, abstractSignsUnderstanding à 2 et languageReception à 2.

N°	Situations d'entrée		Résultats en logique booléenne	Résultats en logique floue	Évaluation des experts
	Variables d'entrée	Valeurs			
1	ictExperience abstractSignsUnderstanding languageReception	1/2/2	displayMode = textOnly		◐
		1/2/3	displayMode = textOnly	displayMode = textAndGraphics	
		1/4/4	displayMode = textOnly	displayMode = textAndGraphics	
		3/3/3	displayMode = textAndGraphics		
2	languageReception attention	1/1	verbose = throrough		●
		1/2	verbose = throrough	verbose = summarized	
		1/3	verbose = throrough	verbose = concise	
		3/4	verbose = concise	verbose = laconic	
3	itExperience ictExperience	1/1	guidanceLevel = medium		◐
		1/3	guidanceLevel = medium	guidanceLevel = high	
		3/3	guidanceLevel = high		
4	visualAcuity handPrecision	1/1	clickCorrection = noCorrection		○
		0/1.49	clickCorrection = noCorrection		
		1/2.99	clickCorrection = noCorrection	clickCorrection = lowCorrection	
		0/3.49	clickCorrection = noCorrection	clickCorrection = lowCorrection	
		2/3	clickCorrection = lowCorrection	clickCorrection = mediumCorrection	
		3/3	clickCorrection = mediumCorrection		
		2.5/4	clickCorrection = mediumCorrection	clickCorrection = highCorrection	
5	visualAcuity	0	fontSize = 12		●
		0.5	fontSize = 12	fontSize =12.75	
		1	fontSize = 12	fontSize =13.5	
		1.5	fontSize = 14	fontSize = 14.25	
		…	…	…	

Accord des experts : ● Total, ◐ Partiel, ○ Faible

Tableau 2 : Synthèse des résultats de l'analyse comparative entre logique booléenne et logique floue

Les deux colonnes suivantes précisent les résultats en logique booléenne et en logique floue.

Enfin, la dernière colonne donne un résumé d'une évaluation effectuée auprès de 3 experts sous forme de boules de Harvey. Les résultats de cette évaluation sont discutés dans la section suivante.

Les différences de résultats en logique booléenne et en logique floue, pour une même situation d'entrée, permettent de mettre en avant deux points.

1) Pour des règles définies avec une complexité similaire entre booléenne et floue, la logique floue permet d'obtenir des résultats plus cohérents au regard des situations d'entrée, et ce, sans avoir besoin de définir et utiliser un système de poids pour les règles, comme c'est le cas pour la logique booléenne. Par exemple, la valeur calculée du displayMode est identique en logique booléenne et floue pour les 1ère et 4ème situations, c'est-à-dire lorsque les 3 valeurs d'entrées sont soit toutes faibles soit toutes élevées. Par contre, lorsque les valeurs d'entrées fournies sont hétéroclites, la logique floue nous renvoie une valeur moyenne, plus cohérente. Ceci se remarque d'autant plus pour les résultats du 2ème jeu de situations d'entrée (adaptation de la verbose en fonction de languageReception et attention), où la variable de sortie verbose peut prendre 4 états (throrough, summarized, concise et laconic). En effet, lorsque l'on fait varier la valeur d'une des variables d'entrée d'un extrême à l'autre, les résultats ci-dessus montrent bien que la valeur finale est lissée, passant par toutes les valeurs intermédiaires, ce qui n'est pas forcément le cas de la logique booléenne.

2) Comme c'est le cas pour la dernière règle d'adaptation du tableau (adaptation de la taille de police en fonction de l'acuité visuelle), la logique floue permet facilement d'obtenir une granularité extrêmement plus fine que la logique booléenne. Cette dernière pourrait permettre d'atteindre une telle granularité, mais cela se ferait au détriment du nombre de seuils à gérer au sein de la règle. Il est important de noter qu'en logique floue, cette granularité n'est d'ailleurs pas limitée aux exemples présents dans le tableau : une valeur de visualAcuity de 0.25 aura comme sortie un fontSize de 12.375, etc.

Cette analyse comparative confirme que la logique floue permet d'obtenir des résultats plus cohérents avec une granularité plus fine sans avoir à gérer la combinaison des multiples règles.

5.3 Experts

Notre évaluation s'est effectuée auprès de trois experts, deux hommes et une femme, tous trois Maîtres de Conférences en IHM depuis une dizaine d'années.

Les différentes situations d'entrée du tableau ont été présentées aux experts et il leur a été demandé de préciser ce qu'ils préconiseraient comme résultat d'adaptation sous forme d'une liste de choix correspondant aux valeurs possibles de la sortie.

Les avis des experts vont être détaillés ci-après.

- Règle n°1 : les experts ont unanimement conseillé l'utilisation d'icônes en plus du texte dans les menus (textAndGraphics), contrairement aux recommandations effectuées notamment par MyUI [11, 35, 36] à partir desquelles nous avons établi notre règle. Ce résultat peut difficilement être interprété du point de vue de la comparaison entre les deux logiques puisqu'il remet en cause la règle en elle-même.

- Règle n°2 : les experts ont tous trois proposé des résultats proches de ceux en logique floue (les 4 résultats possibles étant numérotés de 1 à 4) : 1/2/(2 ou 4)/4 ; 1/2/(3 ou 4)/4 et 1/1/2/3. Les résultats tendent ici vers la logique floue, qui permet d'obtenir plus de linéarité et une prise en compte simplifiée des différentes valeurs d'entrées.

- Règle n°3 : Les 3 experts ont proposé un niveau de guidage medium pour la première situation d'entrée, et un guidage élevé pour la dernière. Pour la situation d'entrée intermédiaire, deux experts nous ont proposé un guidage médium, alors que le dernier nous a indiqué un guidage élevé. Ces résultats ne permettent pas de mettre en avant ni la logique booléenne, ni la logique floue, la règle n'ayant que deux entrées et une sortie à deux états, les différences ne sont pas assez notables.

- Règle n°4 : deux experts sur trois ont précisé que, pour cet exemple, ils utiliseraient la valeur maximale entre les deux entrées afin de calculer la sortie, et non une valeur intermédiaire calculée à partir des deux entrées. Les résultats du 3ème expert penchent quant à eux vers l'utilisation d'une valeur intermédiaire, linéaire. Ici aucune des 2 solutions ne semble actuellement satisfaisante.

- Règle n°5 : deux experts sur trois ont indiqué vouloir des calculs de taille de police précis, ce qui est en accord avec l'adaptation linéaire de la taille de la police, afin de limiter les effets de seuils lors des adaptations dynamiques.

A l'issue d'une discussion à la fin de l'expérimentation, un des experts a précisé que la gestion des valeurs déclarées par les utilisateurs par le biais de l'incertitude était particulièrement adaptée.

5.4 DISCUSSION

D'après les résultats obtenus lors de notre évaluation, la logique floue semble répondre aux besoins d'un moteur d'adaptation des interfaces homme-machine, tout en répondant à nos critères. Toutefois des points peuvent être discutés, en particulier la manière de mettre en œuvre la logique floue lors des phases de fuzzification, défuzzification, et de définition des fonctions d'appartenance, ainsi que la couverture des règles que nous avons utilisées, et le type de variables utilisées.

5.4.1 Fuzzification, défuzzification et fonctions d'appartenance

Comme présenté en section 3.4, la logique floue nécessite des phases de fuzzification et de défuzzification, ainsi que la définition de fonctions d'appartenance pour chacun des états des variables linguistiques. Ces phases sont critiques pour l'obtention de résultats pertinents.

La fuzzification consiste à associer une valeur floue à une valeur d'entrée, selon la fonction d'appartenance du terme de la variable linguistique impliquée dans la proposition floue. C'est la définition du domaine des variables comme nous avons pu le voir pour l'acuité visuelle (figure 7) ou pour la précision des mains. La difficulté est que leur définition reste subjective comme la littérature ne donne généralement pas de valeurs fixées permettant de les définir exactement. Il reste toutefois tout à fait possible de les modifier à postériori si besoin.

En termes de défuzzification, plusieurs solutions sont possibles : premier ou dernier maximum, moyenne des maximas ou un calcul des barycentres pondérés. Le choix de la méthode de défuzzification se fait pour chaque variable linguistique de sortie, et il va dépendre à la fois du type de cette dernière (valeurs continues ou discrètes) et de la logique souhaitée de la règle. Après le test de plusieurs possibilités, les règles ont toutes été définies avec un calcul des barycentres pondérés afin de calculer une moyenne des résultats pour les règles qui influent sur la même variable de sortie. Les

autres méthodes de défuzzification se limiteraient à l'utilisation d'un seul des résultats (par exemple le maxima ou le minima) pour calculer la valeur finale défuzzifiée, ce qui n'est pas souhaitable : pour ce type de règles, cela permettrait la modification de l'une des valeurs d'entrée sans aucun impact sur la valeur de sortie calculée. Toutefois le calcul du barycentre pondéré n'est pas parfait : par exemple, il a nécessité de définir le domaine de valeur de la taille de police de caractères (fontSize, figure 8) entre 6 et 24 pour avoir des résultats pertinents entre 12 et 18. Néanmoins, elle nous a toujours permis d'obtenir des résultats qui nous semblent pertinents.

Enfin, les fonctions d'appartenance peuvent être créées de différentes manières : fonctions continues par morceaux, lignes brisées rectangulaires ou triangulaires, gaussiennes… Certains types de courbes ne correspondraient pas à nos travaux. Par exemple, les fonctions continues par morceaux entraîneraient des effets de seuils au niveau des discontinuités et ne seraient pas des plus adaptées pour représenter des caractéristiques utilisateurs. De même, les lignes brisées rectangulaires reviennent à supprimer la gestion de l'incertitude et auraient donc un fonctionnement similaire à la logique booléenne.

Dans nos travaux, nous avons utilisé des lignes brisées triangulaires car, d'une part, elles permettent des calculs plus simples, mais surtout parce qu'elles sont plus faciles à appréhender par les experts. Les gaussiennes peuvent aussi être utilisées, mais sont préférées lorsque les règles sont apprises automatiquement.

5.4.2 Couverture des règles

D'après notre étude, la logique floue semble adéquate pour supporter un moteur d'adaptation d'interfaces. Néanmoins il se pose toujours la question de la pertinence et de la couverture des règles que nous avons exprimées. Celles-ci peuvent être jugées limitées ou choisies arbitrairement. Pour pallier ces défauts, nous avons exprimé dans notre moteur d'adaptation des règles majoritairement issues de la littérature en adaptation des interfaces utilisateur [11, 30, 31, 32, 33, 34]. Ceci regroupe actuellement 3 règles en logique booléenne (6 en logique floue) relatives au dispositifs (taille de l'écran et orientation), 1 règle (4 en logique floue) relative à l'environnement (luminosité) et 30 règles (71 en logique floue) relatives à l'utilisateur. Concernant l'utilisateur, 23 des 30 règles sont issues de la littérature [11, 30, 31, 32, 33, 34], avec notamment 11 règles tirées du projet européen MyUI [11, 35, 36]. Les règles utilisées nous semblent donc pertinentes et ne sont pas biaisés par notre manière de les mettre en œuvre.

5.4.3 Types de variables utilisées

Dans ces travaux, nous avons utilisé des variables quantitatives (ex : visualAcuity), ordinales (ex : clickCorrection) et nominales (ex : types de daltonie). Comparativement à la logique booléenne, les apports présentés par la logique floue dans cet article touchaient les variables quantitatives ou ordinales. Les variables nominales sont quant à elle utilisées de la même façon entre les deux logiques, sans apport de la logique floue. Il est possible qu'il existe un intérêt pour ce dernier type de variable, mais aucun n'exemple n'a encore été trouvé.

6 CONCLUSION

Cet article propose un système d'adaptation des interfaces homme-machine basé sur un moteur d'adaptation en logique floue. Sa contribution porte la démonstration de l'intérêt de la logique floue pour gérer l'incertitude des données du contexte et la combinaison des règles d'adaptation tout en facilitant l'écriture des règles. Une analyse comparative avec la logique booléenne a permis de montrer ces avantages.

Il reste la question de la difficulté d'aborder les aspects spécifiques au flou, c'est-à-dire la fuzzification et défuzzification. Des expériences avec des ingénieurs en informatique pourraient montrer que cette difficulté est surmontable.

Notre moteur d'adaptation en logique floue permettant d'avoir des résultats plus précis dans certains cas, et il serait intéressant de valider auprès d'experts en ergonomie que les résultats obtenus sont pertinents.

Enfin il est possible d'envisager une extension du moteur d'adaptation qui ne se limite pas à de la logique mais propose aussi des techniques d'apprentissage automatique pour les préférences personnelles des utilisateurs comme par exemple celles liées à l'esthétique.

Remerciements

Ce travail a été soutenu par le Label Persyval-Lab de l'Université Grenoble-Alpes.

REFERENCES

[1] D. Thevenin, J. Coutaz et G. Calvary, «A Reference Framework for the Development of Plastic User Interfaces,» in *Multi-Device and Multi-Context User Interfaces: Engineering and Applications Frameworks*, 2003.

[2] J.-S. Sottet, V. Ganneau, G. Calvary, J. Coutaz, A. Demeure, J.-M. Favre et R. Demumieux, «Model-Driven Adaptation for Plastic User Interfaces,» in *Proc. INTERACT 2007, the eleventh IFIP TC13 International Conference on Human-Computer Interaction*, 2007.

[3] M. Blumendorf, S. Feuerstack et S. Albayrak, «Multimodal User Interfaces for Smart Environments: The Multi-Access Service Platform,» in *Proceedings of the working conference on Advanced visual interfaces*, Napoli, 2008, pp. 478-479.

[4] C. Chesta, F. Paternò et C. Santoro, «Methods and Tools for Designing and Developing Usable Multi-Platform Interactive Applications,» *PsychNology*, vol. 2, n° 11, pp. 123-139, 2004.

[5] M. Florins, Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces, Louvain, 2006.

[6] T.-D. Nguyen, J. Vanderdonckt et A. Seffah, «UIPLML: A Pattern Tool for Engineering Multi-Platforms Information Systems,» in *Tenth International Conference on Research Challenges in Information Science (RCIS)*, 2016.

[7] R. Miñón, F. Paternò, M. Arrue et J. Abascal, «Integrating adaptation rules for people with special needs in model-based UI development process,» *Universal Access in the Information Society*, vol. 15, n° 11, pp. 153-168, 2016.

[8] E. Yigitbas, S. Sauer et G. Engels, «Adapt-UI: an IDE supporting model-driven development of self-adaptive UIs,» in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, Lisbon, 2017.

[9] S. Abrahão, F. Bordeleau, B. Cheng, S. Kokaly, R. F. Paige, H. Störrle et J. Whittle, «User Experience for Model-Driven Engineering: Challenges and Future Directions,» in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2017.

[10] «Virtual User Modeling and Simulation Cluster,» 2013. Available: http://vums.iti.gr/indexb60b.html?page_id=44.

[11] M. Peissner, D. Häbe, D. Janssen et T. Sellner, «MyUI: generating accessible user interfaces from multimodal design patterns,» in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, New York, 2012.

[12] F. Paternò, C. Santoro et L. D. Spano, «MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments,» *ACM Transactions on Computer-Human Interaction*, vol. 16, n° 14, 2009.

[13] P. Akiki, «Engineering Adaptive Model-Driven User Interfaces for Enterprise Applications,» 2016.

[14] G. Lehmann, A. Rieger, M. Blumendorf et S. Albayrak, «A 3-layer architecture for smart environment models,» in *Pervasive Computing and Communications Workshops*, Mannheim, 2010.

[15] B. Collignon, J. Vanderdonckt et G. Calvary, «Model-Driven Engineering of Multi-target Plastic User Interfaces,» in *Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems*, Washington DC, 2008.

[16] D. Thevenin, Adaptation in Human Computer Interaction: the case of Plasticity, Grenoble, 2001.

[17] A. Blouin, B. Morin, O. Beaudoux, . G. Nain, P. Albers et J.-M. Jézéquel, «Combining aspect-oriented modeling with property-based reasoning to

improve user interface adaptation,» in Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems, New York, 2011.

[18] V. Genaro Motti, N. Mezhoudi et J. Vanderdonckt, «Machine Learning in the Support of Context-Aware Adaptation,» in *10th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2012.

[19] K. Z. Gajos, D. S. Weld et J. O. Wobbrock, «Automatically generating personalized user interfaces with Supple,» *Artificial Intelligence,* vol. 174, n° 112-13, pp. 910-950, 2010.

[20] J. Cao, N. Xing, A. Chan, Y. Feng et B. Jin, «Service adaptation using fuzzy theory in context-aware mobile computing middleware,» in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[21] H. Desruelle, D. Blomme et F. Gielen, «Adaptive Mobile Web Applications: A Quantitative Evaluation Approach,» in *International Conference on Web Engineering*, 2011.

[22] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, R. G. Crespo et N. Garcia-Fernandez, «Fuzzy System to Adapt Web Voice Interfaces Dynamically in a Vehicle Sensor Tracking Application Definition,» *Soft Computing,* vol. 20, n° 18, pp. 3321-3334, 2016.

[23] H. O. Nyongesa, T. Shicheng, S. Maleki-Dizaji, . S. T. Huang et J. Siddiqi, «Adaptive Web interface design using fuzzy logic,» *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pp. pp. 671-674, 2003.

[24] E. Papatheocharous, M. Belk, P. Germanakos et G. Samaras, «Proposing a Fuzzy Adaptation Mechanism Based on Cognitive Factors of Users for Web Personalization,» *Artificial Intelligence Applications and Innovations,* p. 135–144, 2012.

[25] R. Riding et I. Cheema, «Cognitive Styles — an overview and integration,» *Educational Psychology,* vol. 11, pp. 193-215, 1991.

[26] M. Soui, M. Abed et K. Ghedira, «Fuzzy Logic Approach for Adaptive Systems Design,» *Human-Computer Interaction. Towards Intelligent and Implicit Interaction,* pp. 141-150, 2013.

[27] L. Zadeh, «Fuzzy sets,» *Information and Control,* vol. 8, n° 13, pp. 338-353, 1965.

[28] Zadeh, L., «The concept of a linguistic variable and its application to approximate,» *Information Sciences,* vol. 9, pp. 43-80, 1975.

[29] B. Bouchon-Meunier, La logique floue, puf, 2007.

[30] G. Moss, Gender, Design and Marketing: How Gender Drives our Perception of Design and Marketing, Gower, 2009.

[31] L. Rello et R. Baeza-Yates, «Good fonts for dyslexia,» in *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 2013.

[32] C. Stephanidis, A. Paramythis, M. Sfyrakis, A. Stergiou, N. Maou, A. Leventis, G. Paparoulis et C. Karagiannidis, «Adaptable and adaptive user interfaces for disabled users in the AVANTI project,» in *Proceedings of the 5th International Conference on Intelligence and Services in Networks: Technology for Ubiquitous Telecom Services*, 2006.

[33] J.-P. Leuteritz, H. Widlroither, A. Mourouzis, M. Panou, M. Antona et A. Leonidis, «Development of Open Platform Based Adaptive HCI Concepts for Elderly Users,» in *Lecture Notes in Computer Science*, vol. 5615, 2009.

[34] C. Denis et L. Karsenty, «Inter-Usability of Multi-Device Systems – A Conceptual Framework,» in *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, 2005, pp. 373 - 385.

[35] «MyUI project Official Web Page,». Available: http://www.myui.eu/.

[36] «MyUI Pattern Browser,» 2013. Available: http://myuipatterns.clevercherry.com/index.php?title=Main_Page.