Reconnaissance de contexte stable pour l'habitat intelligent

Paoli Pietropaoli

ANNÉE 2013

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Bastien PIETROPAOLI

préparée à l'unité de recherche IRISA – UMR6074
Institut de Recherche en Informatique et Systèmes Aléatoires
ISTIC

Reconnaissance de contexte stable pour l'habitat intelligent

Thèse soutenue à Rennes le 10 décembre 2013

devant le jury composé de :

Jean DEZERT / *Rapporteur*
Ingénieur de Recherche (Dr.), ONERA

François SPIES / *Rapporteur*
Professeur, Université de Franche-Compté

Myriam FRÉJUS / *Examinatrice*
Chercheur expert, EDF R&D

Jean-Marie BONNIN / *Examinateur*
Professeur, Télécom Bretagne

Michel BANÂTRE / *Directeur de thèse*
Directeur de recherche, INRIA Rennes, Bretagne-Atlantique

Frédéric WEIS / *Co-directeur de thèse*
Maître de Conférences, HDR, Université de Rennes 1

Table des matières

Table des figures

Liste des tableaux

Introduction

L'habitat intelligent (*smart home* en anglais) est l'objet de nombreux travaux de recherche [1]. Il a pour objectif d'améliorer le confort au sein d'une maison, en automatisant par exemple l'éclairage et le chauffage. Il peut traiter d'aspects liés à la sécurité, en gérant le portail, le garage et les volets. Il peut viser également des services de divertissement avec la gestion des équipements vidéo et hi-fi [2]. Son développement profite aujourd'hui de l'émergence rapide des principes de l'informatique ubiquitaire [92, 93, 94]. Dans cette branche de l'informatique, les systèmes d'information sont omniprésents et l'interaction avec ces derniers est implicite. En effet, de nombreux systèmes profitent de la *sensibilité au contexte* (*context-awareness* en anglais). Cela permet à des services de s'adapter au mieux aux besoins d'un utilisateur sans que celui-ci n'ait besoin de les expliciter. Ainsi, dans un habitat, le chauffage peut être géré finement en tenant compte des préférences des utilisateurs présents dans l'habitat mais aussi leurs activités en cours. Ainsi, une personne assise dans le canapé n'a pas besoin du même niveau de chauffage qu'une personne qui fait le ménage activement.

Aujourd'hui, les technologies vues comme nécessaires au développement de l'informatique ubiquitaire dans les articles visionnaires de Mark Weiser sont disponibles : Internet, les communications sans fil, les terminaux de différentes tailles, les capteurs et actionneurs peu coûteux et bientôt la démocratisation de l'Internet des Objets (*Internet of Things* en anglais) [5, 74].

La sensibilité au contexte passe évidemment par la reconnaissance de contexte. Ainsi, il est possible de trouver des travaux de recherche sur la reconnaissance de contexte dans les habitats intelligents s'appuyant sur des technologies telles des sols et meubles capacitifs [89], des caméras [76] ou encore des gants munis de lecteurs RFID [1] [71]. Dans le premier cas, le remplacement du sol dans une maison ou un appartement déjà existant est beaucoup trop contraignant pour être mis en place dans de nombreux habitats. Dans le second cas, l'utilisation de caméras et de microphones peut entraîner un rejet de la part des habitants qui tiennent à ce que leur vie privée soit préservée [13, 67]. Enfin, l'utilisation d'équipements à placer sur les habitants est une contrainte trop importante. En effet, ces équipements peuvent être gênants, peu acceptables et être tout simplement oubliés.

La démocratisation des habitats intelligents est grandement freinée par des coûts exhorbitants pour des systèmes simples mais aussi à cause de problèmes d'acceptabilité liés notamment aux technologies utilisées. De nombreux travaux cherchent à répondre à ces problèmes [27, 86]. Il est donc crucial de respecter des contraintes strictes sur les matériels qu'il est possible de déployer

1. http ://fr.wikipedia.org/wiki/Radio-identification

dans un habitat intelligent. Ainsi, dans nos travaux, nous souhaitons respecter les contraintes matérielles suivantes :

- *L'instrumentation doit être légère* : elle doit pouvoir être déployée dans n'importe quel habitat sans nécessiter de travaux de grande ampleur.
- *Le coût du système doit être le plus faible possible* : le bas coût des capteurs et des calculateurs limite leurs capacités. La consommation électrique du système doit aussi être négligeable voire largement compensée par les gains apportés par celui-ci.
- *Les habitants ne doivent pas être instrumentés* : aucun capteur ni équipement ne doit se trouver sur les habitants. Ainsi, l'usage du smartphone doit être réduit et ne permettre qu'un supplément facultatif d'informations.
- *L'instrumentation ne doit pas être invasive* : les caméras et les microphones sont à proscrire. Tous les capteurs/équipements violant la vie privée et l'intimité des habitants doivent être bannis.

Les services d'un habitat intelligent peuvent reposer sur un large panel de données contextuelles. Pour s'en convaincre, considérons le service le plus couramment déployé dans les habitats intelligents : l'éclairage automatique [86]. Il est souvent implémenté de façon assez simple à l'aide de quelques capteurs et de règles simples. Par exemple, il peut être mis en place avec un capteur de luminosité et un capteur de mouvement, le tout appuyé par une temporisation. Ainsi, dès qu'un mouvement est détecté et si la luminosité est faible, l'éclairage est allumé pour un certain temps. Malheureusement, ce type d'implémentation ne fonctionne pas pour des personnes travaillant devant un ordinateur par exemple. On peut remarquer que la donnée qui devrait être réellement utilisée ici est la présence et non le mouvement (bien qu'il soit une preuve de présence). Cependant, si la présence peut être déterminante pour ce service, elle ne permet pas de gérer toutes les situations. En effet, si une personne dort dans la pièce, la lumière ne doit pas être allumée. On constate donc avec ce simple exemple que de nombreuses données contextuelles sont nécessaires à la construction d'un service sensible au contexte efficace et non frustrant. De plus, ces données ne peuvent pas toutes êtres obtenues directement via des capteurs élémentaires.

La plupart des données contextuelles dont nous avons besoin sont en réalité des abstractions. Comme suggéré par Coutaz *et al.* [19], nous classons ces abstractions en différents niveaux. La figure 1.1 illustre le modèle en couches suggéré. Il est découpé en quatre couches :

- *Sensing* : c'est la couche de plus bas niveau, elle correspond à l'ensemble des données brutes récoltées par les capteurs disséminés dans l'environnement ;
- *Perception* : cette couche correspond à un premier niveau d'abstraction indépendant des aspects technologiques. Dans notre exemple d'éclairage automatique, cela correspond à la détection de présence qui peut être obtenue de diverses manières ;
- *Situation and context identification* : cette couche correspond aux situations de plus haut niveau comme le fait que quelqu'un soit en train de dormir dans la pièce par exemple ;
- *Exploitation* : cette couche correspond aux services que l'on souhaite déployer. Ils reposent sur les trois couches fournissant des données contextuelles.

Les données contextuelles abstraites dont ont besoin les services sensibles au contexte peuvent être obtenues à l'aide de méthode de fusion de données. Nous nous intéressons donc dans ce document aux différentes théories et méthodes existantes nous permettant d'obtenir des données

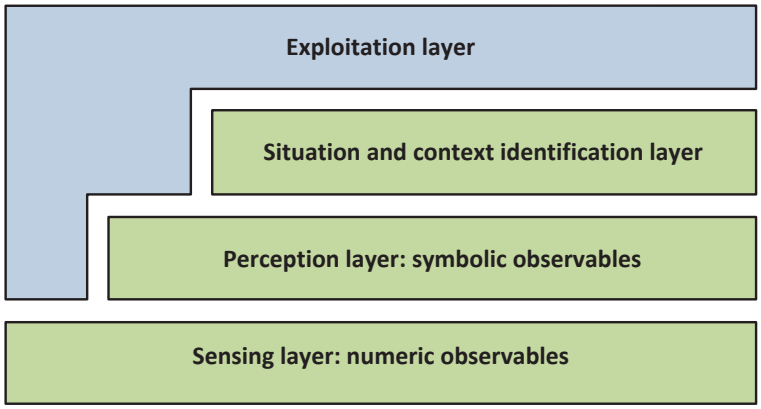

FIGURE 1 – Modèle en couches, proposé par Coutaz *et al.* [19], pour la modélisation et le calcul des données contextuelles.

contextuelles abstraites. Nous retenons quelques théories qui nous semblent pertinentes pour la construction d'une architecture en couches. La totalité de l'architecture étant un projet très large, nous concentrons nos travaux sur la partie basse de l'architecture.

L'objectif de cette thèse est donc de montrer qu'il est possible de faire de la reconnaissance de contexte stable pour l'habitat intelligent tout en respectant des contraintes matérielles améliorant l'acceptabilité du système par les habitants.

Contributions

Dans ce document, nous présentons l'ensemble des travaux réalisés dans le but d'effectuer une reconnaissance de contexte stable pour l'habitat intelligent.

Théorie des fonctions de croyance : application et implémentation

À bas niveau, pour la reconnaissance de contexte, nous avons retenu la théorie des fonctions de croyance. Nous montrons au chapitre 3 comment cette théorie peut être appliquée pour la reconnaissance de contexte à partir de données de capteurs élémentaires. Nous présentons ainsi comment nous construisons les croyances à partir de données de capteurs, comment ces croyances sont fusionnées et enfin comment le système prend une décision sur ce qu'il considère être l'état réel du monde.

Stabilisation des données contextuelles : temporisation des croyances

L'application de la théorie des fonctions de croyance nous permet de mettre en avant un ensemble de problèmes : l'asynchronicité des preuves recueillies, l'instabilité des mesures retournées par les capteurs, les pertes de données dues aux communication sans fil et enfin la nécessité d'accumuler des preuves dans le temps.

Ainsi, au chapitre 4, nous proposons deux algorithmes de temporisation des croyances nous permettant de résoudre tous ces problèmes. Nous présentons en détails le comportement de ces deux algorithmes et montrons leur pertinence dans une application réelle.

Architecture en composants : répartition et flexibilité

Afin de permettre une mise à l'échelle, une flexibilité mais aussi une répartition de la charge de calcul, nous présentons une architecture en composants que nous avons déployée. Cette dernière permet l'ajout de nouvelles fonctionnalités sans avoir à modifier l'existant, de déployer à la volée de nouveaux composants et donc de nouveaux modèles ainsi que de nouveaux services. Nous présentons chapitre 5 les outils nous permettant la mise en place de cette architecture et chapitre 6 en quoi consiste cette architecture.

Capteurs virtuels : stabilité des modèles

En déployant des capteurs et des modèles de croyance dans différents environnements, nous posons la question de la stabilité de nos modèles. Au chapitre 6, nous montrons comment l'ajout d'un nouveau type de composant, les capteurs virtuels, permettant de simuler des capteurs permet de répondre à cette question.

Implémentation d'un prototype

Enfin, nous présentons un prototype complet auquel nous avons intégré nos travaux. Ce prototype nous permet de valider la possibilité de faire de la reconnaissance de contexte stable pour l'habitat intelligent avec une instrumentation fortement contrainte tout en offrant flexibilité et efficacité. Le déploiement de nombreux services différents permet de valider aussi la pertinence des différents niveaux d'abstractions de données contextuelles et les possibilités offertes par une telle approche.

Plan du document

Ce document de thèse est décomposé en trois grandes parties. Dans la première partie, nous présentons un exemple courant de service déployé dans les habitats intelligents, l'éclairage automatique. Cette description nous permet de comprendre le besoin en données contextuelles d'un tel service et surtout le besoin en données contextuelles de différents niveaux d'abstraction. Ces données contextuelles vont donc être obtenues par l'intermédiaire de capteurs et d'équipements augmentés mais aussi et surtout via des mécanismes de fusion de données. Nous présentons ensuite quelques méthodes et théories existantes utilisées en fusion de données.

La seconde partie s'intéresse au premier niveau d'abstraction des données brutes issues des capteurs avec l'application de la théorie des fonctions de croyance. Cette théorie repose sur l'accumulation de preuves. Nous présentons donc les bases de la théorie et comment celle-ci peut être appliquée à un ensemble de capteurs pour déduire des abstractions telles que la présence dans une pièce. Des problèmes d'instabilité rencontrés lors de l'application de la théorie à des

capteurs réels sont ensuite mis en avant pour justifier le développement de deux algorithmes de temporisation. Ces algorihtmes nous permettent de compenser les pertes de données, principalement dues aux communication sans fil, mais aussi de récolter des preuves asynchrones ou encore d'accumuler des preuves dans le temps.

La dernière partie traite du développement et du déploiement d'un prototype qui nous a permis de valider notre approche. Ainsi, nous démontrons la pertinence des différents niveaux d'abstraction, la possibilité d'intégrer dans un même système différentes approches de fusion de données et enfin de déployer un grand nombre de services avec une instrumentation limitée.

Enfin, en conclusion, nous discutons une partie de nos choix et les perspectives qui s'offrent à la suite de ce travail de thèse.

Première partie

Informatique ubiquitaire et sensibilité au contexte

Chapitre 1

Services sensibles au contexte dans l'habitat intelligent

First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives.
Mark D. Weiser

Quand vous citez un texte con, n'oubliez pas le contexte.
Jacques Prévert

Dans les années 80 est née l'automatique pour la maison aussi appelée *domotique*. Son but est d'améliorer le confort, en automatisant les lumières et les chauffages, la sécurité, en gérant le portail, le garage et les volets, ainsi que le divertissement avec la gestion des équipements tels que le téléviseur ou encore la chaîne hi-fi [2]. Devenu aujourd'hui l'habitat intelligent (*smart home* en anglais), il est l'objet de nombreux travaux de recherche [1].

Le développement de l'habitat intelligent connaît un renouveau aujourd'hui avec l'introduction dans son développement d'idées issues de l'informatique ubiquitaire [92, 93, 94]. Cette branche de l'informatique s'intéresse à l'introduction de systèmes d'information omniprésents. Dans le cas de l'habitat intelligent, ces sytèmes peuvent se trouver dispersés dans l'environnement ou intégrés aux équipements habituels des maisons tels que l'électroménager ou encore le

téléviseur. Un des principaux concepts de ce paradigme est l'invisibilité de l'interaction. En effet, l'utilisateur n'interagit pas toujours consciemment ni directement avec les systèmes mais ces derniers s'adaptent au contexte pour fournir des services appropriés quand ils sont nécessaires. C'est ce qu'on appelle la *sensibilité au contexte* (*context-awareness* en anglais). Afin d'avoir une définition fixe du contexte, nous adoptons la définition donnée par Dey [24] :

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Dans le cas d'un habitat intelligent, sont donc considérées comme données contextuelles toutes les données concernant les habitants, leurs activités mais aussi l'habitat et son environnement (données météo, prix de l'énergie, etc).

Dans ses articles visionnaires, Mark Weiser décrit un environnement dans lequel tous les objets de la vie courante deviennent intelligents et s'adaptent aux besoins de l'utilisateur. Ce dernier dispose de divers terminaux pour recevoir des informations mais aussi interagir avec le système lorsqu'il le souhaite [92]. Il peut aussi utiliser des commandes vocales [94].

Aujourd'hui, toutes les technologies nécessaires au développement de l'informatique ubiquitaire sont là : Internet, les communications sans-fil (wi-fi, bluetooth, 802.15.4, GSM, 3G/4G, etc), les terminaux de différentes tailles (smartphones, tablettes) mais aussi toutes sortes de technologies de capteurs/actionneurs peu coûteux (Phidgets [1], EnOcean [2], etc) ainsi que des technologies comme les tags RFID et enfin l'arrivée de l'Internet des Objets (*Internet of Things* en anglais) [5, 74] et des équipements augmentés comme les téléviseurs ou les électroménagers capables de communiquer leur statut ainsi que des informations sur leur environnement. A ces avancées technologiques significatives, il faut bien entendu rajouter toutes les recherches en intelligence artificielle, sur la reconnaissance de contexte, la fusion de données, la standardisation des protocoles de communication, la reconnaissance de la parole, la vision par ordinateur, etc.

Nous avons donc à notre disposition tous les outils nécessaires à la construction d'un habitat intelligent capable de s'adapter aux besoins de ses occupants pour leur fournir des services adaptés aux contextes afin de les aider dans leurs tâches quotidiennes mais aussi améliorer leur sécurité et leur confort.

Dans ce chapitre, nous présentons quelques travaux existants sur l'habitat intelligent. En mettant en avant les principaux défauts des approches présentées, nous dressons la liste des contraintes matérielles que nous souhaitons respecter pour la mise au point d'un habitat intelligent réaliste. Enfin, en décrivant un exemple de service courant dans l'habitat intelligent, l'éclairage automatique, nous illustrons l'intérêt d'avoir à notre disposition un large panel de données contextuelles distribuées en plusieurs niveaux d'abstraction.

1.1 L'habitat intelligent : analyse des contraintes

Bien que toutes les technologies et méthodes nécessaires au développement d'un habitat intelligent soient accessibles aujourd'hui, leur intégration et la construction d'un réel habitat

1. http ://www.phidgets.com
2. http ://www.enocean.com

intelligent ne sont pas triviales. En effet, les maisons déjà équipées de tels systèmes ne sont pas toujours source de satisfaction [86]. Les systèmes reposant sur du tout automatique et des règles sont souvent mis en échec. Les exceptions peuvent être nombreuses dans l'habitat : partir au/rentrer du travail à des heures différentes, se coucher plus ou moins tard, etc [34, 88]. La sensibilité au contexte est donc nécessaire pour permettre au système de s'adapter en temps réel aux activités des habitants.

Malheureusement, la reconnaissance de contexte n'est pas une tâche facile. Elle repose souvent sur des instrumentations coûteuses et qui peuvent être contraignantes. Nous développons ces aspects dans la suite de ce chapitre.

1.1.1 Présentation de quelques exemples

Dans cette section, nous ne cherchons pas faire un état de l'art de tout ce qui se fait en recherche sur les habitats intelligents (pour cela, voir [1]) mais plutôt décrire quelques exemples typiques en analysant leurs limites.

Reconnaissance d'activités complexes

Les activités quotidiennes sont des données contextuelles concernant les utilisateurs et leur usage des équipements essentielles si l'on souhaite les assister et leur permettre d'économiser de l'énergie. Dans les travaux de Philopose *et al.*, des activités quotidiennes (se laver, se brosser les dents, faire le ménage, etc) sont reconnues en attachant à divers objets d'intérêt des tags RFID [3] et en équipant les habitants d'un gant muni d'un lecteur RFID [71]. En procédant ainsi, il est possible de détecter toute utilisation d'un objet auquel un tag RFID a été attaché. Ensuite, un modèle attribuant à chaque objet une probabilité d'être utilisé pour chacune des activités modélisées permet de reconnaître l'activité menée par l'habitant.

La principale limite de cette approche est l'obligation d'équiper les habitants avec un équipement contraignant (ici, un gant). Il est peu probable que quelqu'un n'ayant pas besoin d'être assisté accepte de s'équiper. Quand bien même ce serait le cas, un oubli empêcherait purement et simplement le système de fonctionner. De plus, certaines activités, par nature, empêchent le port d'équipement comme le fait de se laver par exemple. La seconde limite d'une telle approche concerne la possibilité de prendre en charge plusieurs utilisateurs. Si les activités d'une personne seule peuvent être reconnues, ils ne précisent pas si leur modèle permet de prendre en compte plusieurs utilisateurs dont les activités peuvent être indépendantes, collaboratives ou encore entremêlées. La présence de plus d'un habitant peut grandement modifier les services à rendre. Par exemple, une personne lisant tranquillement allongée dans le canapé n'a pas besoin du même niveau de chauffage qu'une personne faisant le ménage activement. Un compromis est certainement nécessaire.

3. http ://fr.wikipedia.org/wiki/Radio-identification

Localisation au sein de l'habitat

La détection de la présence dans les différentes pièces d'un habitat est essentielle pour faciliter la compréhension des activités en cours mais aussi pour adapter l'usage de certains équipements tels que la lumière ou le chauffage. Dans les travaux de Valtonen *et al.*, la position des habitants est détectée via un sol et des meubles capacitifs (capables de détecter lorsqu'ils sont touchés). Le sol est composé de dalles permettant de situer une personne dans la pièce [89].

Bien que les résultats de leur plateforme soit satisfaisants, ce dispositif requiert une installation lourde notamment en équipant la totalité du sol d'un appartement. En plus du coût certainement très élevé en temps et en argent d'une telle installation, le réalisme du déploiement d'une telle solution à grande échelle est plus que discutable. De plus, le port de chaussures peut perturber la détection et une connaissance de la disposition des meubles est nécessaire si l'on souhaite reconnaitre des activités via ce système (l'emplacement de la personne peut suggérer telle ou telle activité en fonction des équipements/meubles alentours). Enfin, aucun scénario à plusieurs habitants n'a été testé.

Détection de posture

Dans le cas d'habitats destinés à des personnes âgées ou des personnes handicapées, la détection de la posture peut être intéressante pour détecter la chute d'une personne et ainsi pouvoir intervenir si nécessaire. Dans les travaux de Ricquebourg *et al.*, plusieurs capteurs sont utilisés pour détecter la posture d'une personne : une caméra munie d'un miroir à vision omnidirectionnelle, des capteurs de pressions sur les chaises et enfin un accéléromètre sur la ceinture de la peronne [76].

Une fois de plus, on retrouve l'équipement des personnes qui n'est pas acceptable dans l'habitat intelligent grand public mais on retrouve aussi un autre équipement très problématique : la caméra. Les caméras et les microphones peuvent être perçus comme une violation de l'intimité et de la vie privée [13, 67]. En effet, les habitants peuvent avoir l'impression d'être écoutés et/ou observés par quelqu'un. Cela pose évidemment d'importants problèmes d'acceptabilité et de confort. L'instrumentation ne doit pas déranger les habitants.

1.1.2 Analyse

Dans toutes les approches décrites, le système peut se tromper. Aucun système n'est infaillible, que ce soit les capteurs utilisés ou encore les algorithmes permettant de reconnaître des activités. C'est pourquoi, toutes les approches présentées mettent en place une gestion de l'incertitude, via l'utilisation de probabilités pour les deux premières et via l'utilisation de la théorie des fonctions de croyance pour la troisième. Il est donc évident que nous devrons nous aussi prendre en compte l'incertitude dans la construction de notre prototype.

De ces quelques exemples, nous pouvons également tirer un ensemble de contraintes à respecter vis-à-vis du matériel qu'il est possible de déployer dans un habitat intelligent. Ces contraintes, très importantes pour l'acceptatibilité et donc pour la possibilité d'un déploiement à grande échelle, vont fortement influencer les équipements que nous utiliserons dans la construction de

notre prototype. Les contraintes que nous essayerons au maximum de respecter sont donc les suivantes :

- *L'instrumentation doit être légère* : elle doit pouvoir être déployée dans n'importe quel habitat sans nécessiter de travaux de grande ampleur.
- *Le coût du système doit être le plus maîtrisé possible* : le bas prix des capteurs et des calculateurs limite leurs capacités. La consommation électrique du système doit aussi être négligeable voire largement compensée par les gains apportés par celui-ci.
- *Les habitants ne doivent pas être instrumentés* : aucun capteur ni équipement ne doit se trouver sur les habitants. L'usage du smartphone sera minimum et ne représentera qu'un supplément facultatif d'informations. Il pourra être utilisé lorsque les habitants sont à l'extérieur pour, par exemple, prédire un retour à la maison.
- *L'instrumentation ne doit pas être invasive* : les caméras et les microphones sont à proscrire. Tous les capteurs/équipements violant de façon évidente la vie privée et l'intimité des habitants doivent être bannis.

Ces contraintes matérielles sont les premières difficultés que l'on peut rencontrer dans la mise au point d'un habitat intelligent. En effet, si l'on souhaite les respecter, nous devons donc nous limiter fortement en termes de possibilités dans l'obtention de données contextuelles. Les caméras sont par exemple d'excellents outils pour situer des personnes dans un habitat. Nous devrons faire sans. De même, si un sol capacitif permet de situer des personnes avec précision et ainsi détecter des activités spécifiquement liées à certains équipements/meubles (travailler sur un bureau par exemple), une telle installation n'est pas réaliste à grande échelle.

1.2 Analyse d'un exemple de service : l'éclairage automatique

En analysant quelques projets typiques de la recherche dans le domaine des habitats intelligents, nous avons pu mettre en avant des éléments matériels qui en limitent la diffusion et l'acceptabilité. De plus, chacun des travaux présentés traite d'un type de données contextuelles différent. Ainsi, nous avons décrit un système permettant de détecter des activités quotidiennes, un système permettant de localiser les personnes dans un appartement et enfin un système permettant de détecter la posture d'une personne. On constate donc que l'habitat intelligent peut reposer sur un large panel de données contextuelles. Cette diversité est nécessaire si l'on souhaite mettre au point des services complexes.

Nous souhaitons analyser ici cette "diversité" de données contextuelles peuvent être nécessaires au travers d'un service simple *a priori* : l'éclairage automatique. Ce dernier est un des services les plus communément retrouvés dans les habitats intelligents. Malheureusement, il repose le plus souvent sur un ensemble de règles simples peu adaptées aux réels besoins des habitants. De fait, ce service est souvent source de frustration [86]. Cette frustration laisse à penser que ce service est plus complexe qu'il n'y paraît. Dans cette section, nous essayerons donc de comprendre en détail comment un service d'éclairage automatique peut fonctionner.

1.2.1 Approche contextuelle du service

L'éclairage automatique a deux fonctionnalités : allumer et éteindre la lumière. Ces deux fonctionnalités ne doivent évidemment pas intervenir dans les mêmes situations. Afin de comprendre la complexité de la mise en place d'un tel service, nous allons décrire quelques situations usuelles dans lesquelles ce service peut être utile.

L'usage le plus usuel est de lier la lumière à l'usage d'un interrupteur. Lorsqu'on rentre dans une pièce, si la luminosité est faible, on allume. On a donc une double condition facile à respecter avec un capteur de luminosité et un capteur de passage dans la porte ou un capteur de mouvement. On s'appuie donc ici deux données contextuelles : **la luminosité** et **le mouvement** dans la pièce. Ces deux données peuvent être obtenues directement à l'aide de *capteurs élémentaires*.

Autre cas possible, la luminosité peut avoir baissé alors qu'une personne est restée présente dans la pièce. Il faut alors allumer la lumière dès que la luminosité est trop faible. Le capteur de mouvement est donc préférable au capteur de passage dans la porte. Malheureusement, la vraie donnée contextuelle qui nous intéresse ici n'est pas la détection d'un mouvement mais plutôt une présence dans la pièce. Les capteurs de mouvement sont souvent utilisés avec une temporisation pour l'éclairage automatique. Si cela fonctionne bien pour les lieux de passages, les couloirs par exemple, ce système peut être mis en échec dans des toilettes ou des bureaux, soit parce que la temporisation n'est pas assez longue, soit parce que le mouvement n'est pas suffisamment représentatif de la présence dans la pièce. Ainsi, il n'est pas rare de voir des gens lever et agiter le bras pour se faire détecter chaque fois que la lumière s'éteint. De plus, une temporisation trop longue peut amener un éclairage à être allumé beaucoup plus longtemps que ce qu'il est nécessaire. Si l'on ne fait que passer dans une pièce, il n'est pas nécessaire d'allumer la lumière pour plusieurs minutes.

On voit donc ici que **la présence** et **l'absence** sont des données contextuelles essentielles pour l'éclairage automatique. La présence n'est pas facilement détectée avec un seul capteur. C'est donc, semble-t-il, une donnée contextuelle de plus haut niveau que les données fournies par les capteurs. La présence et l'absence sont donc des abstractions facilement compréhensibles par un humain mais qu'il n'est pas possible d'obtenir aussi simplement avec un système. De même, la gestion de l'incertitude est essentielle pour gérer les situations décrites. Il est évident que nos capteurs ne pourront pas toujours détecter des indices de présence (par exemple avec la détection de mouvement), et quand bien même ils le feraient, ils pourraient ne pas être suffisants pour déduire une présence. Il faut donc rajouter aux abstractions l'incertitude les concernant.

De même, si la présence est essentielle, une présence dans une pièce ne requiert pas toujours d'allumer la lumière, y compris si la luminosité est faible. C'est le cas par exemple dans une chambre à coucher lorsqu'une personne dort. On évitera d'allumer la lumière dans ces cas là, y compris lorsqu'une autre personne rentre dans la pièce. On a donc besoin ici d'une nouvelle donnée contextuelle concernant le fait qu'**une personne dort** ou pas dans la pièce. Encore une fois, la donnée "une personne dort" ne peut pas être obtenue directement à l'aide de capteurs, notamment si nous nous interdisons d'équiper les habitants. Nous avons donc ici, une fois de plus, une abstraction de plus haut niveau que les données brutes issues des capteurs.

Il est possible de décrire toute sorte de situations où l'éclairage nécessite une gestion plus fine

que le simple fait d'allumer la lumière lorsqu'une personne est présente dans la pièce. Ce peut être le cas si l'on considère par exemple qu'il peut exister plusieurs éclairages différents dans une pièce, chacun créant une ambiance différente, pouvant déranger plus ou moins. Par exemple, une personne regardant un film n'aura pas nécessairement besoin du même éclairage qu'une personne qui lit un livre, etc.

Avec ces situations simples, on constate facilement que la mise en place d'un service d'éclairage automatique efficace et non frustrant n'est pas simple. Il est possible de s'appuyer sur un grand nombre de données contextuelles si l'on souhaite rendre les services les plus adaptés possibles. Ici, nous avons vu que les données sur **la luminosité**, **le mouvement**, **la présence** (et donc **l'absence**) et enfin le fait qu'**une personne dort** ou pas dans la pièce peuvent toutes permettre d'offrir un service plus ou moins raffiné aux habitants. De plus, la présence de plusieurs habitants augmente aussi grandement la complexité avec l'obligation de faire des compromis. Ces derniers passent certainement par une prise en compte de l'incertitude liée aux équipements mais aussi aux algorithmes nous permettant d'identifier les situations en cours.

En décrivant ces quelques situations courantes dans l'habitat associées à un service *a priori* simple, nous avons pu mettre en avant le besoin de récolter des données de capteurs mais aussi et surtout la nécessité de disposer de données contextuelles abstraites telles que la présence.

1.2.2 Architecture en couches

L'idée d'utiliser différents types de données contextuelles n'est pas nouvelle. Dans la littérature, il est possible de trouver des modèles génériques pour les applications sensibles au contexte [19]. La figure 1.1 illustre le modèle suggéré par Coutaz *et al.*. Ce modèle considère plusieurs niveaux d'abstraction pour les données contextuelles. Il est composé des quatres couches suivantes :

- *Sensing* : c'est la couche de plus bas niveau, elle correspond à l'ensemble des données brutes récoltées par les capteurs disséminés dans l'environnement ;
- *Perception* : cette couche correspond à un premier niveau d'abstraction indépendant des aspects technologiques. Dans notre exemple d'éclairage automatique, cela correspond à la détection de présence qui peut être obtenue de diverses manières ;
- *Situation and context identification* : cette couche correspond aux situations de plus haut niveau comme le fait que quelqu'un soit en train de dormir dans la pièce par exemple ;
- *Exploitation* : cette couche correspond aux services que l'on souhaite déployer. Ils reposent sur les trois couches fournissant des données contextuelles.

Dans le modèle d'origine, sont aussi considérés la sécurité, l'historique et la découverte (de capteurs, d'équipements, de données accessibles). Nous laisserons ces aspects de côté dans ce document pour nous intéresser exclusivement à la reconnaissance de contexte et aux services.

Comme suggéré par Coutaz *et al.*, seule la couche la plus basse fournit des données directement issues des équipements. Les couches supérieures fournissent des données totalement déconnectées des aspects technologiques et correspondent donc à des abstractions stables qui ne varieront pas d'un déploiement à un autre. Ces abstractions sont stables car, du fait qu'elles soient intelligibles facilement par des humains, elles ne dépendront pas de l'environnement dans lequel elles sont

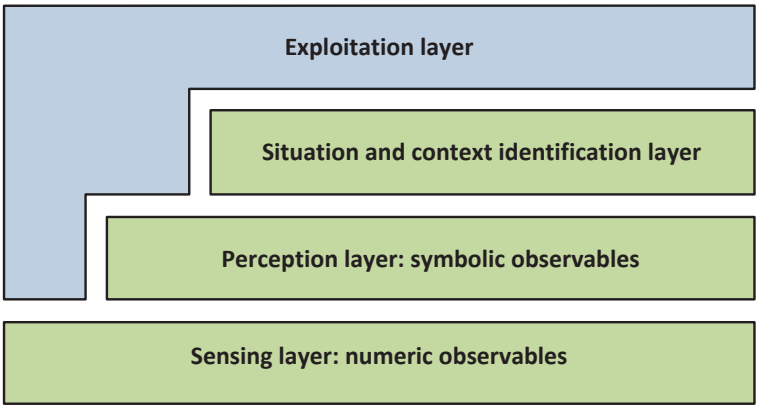

FIGURE 1.1 – Modèle en couches, proposé par Coutaz *et al.* [19], pour la modélisation et le calcul des données contextuelles.

utilisés. Ainsi, il est possible de définir la présence aussi bien dans une cuisine que dans une chambre à coucher ou un bureau. Il se peut aussi que d'un déploiement à l'autre, ces abstractions ne soient pas obtenues de la même manière, avec des configurations différentes de capteurs par exemple, mais elles restent inchangées. Ainsi, les services reposant sur ces abstractions ne seront pas modifiés d'un déploiement à un autre.

Il est donc possible de faire reposer une grande partie des services sur des données contextuelles usuelles qu'il sera toujours possible d'obtenir quelque soit l'environnement. Seule la manière d'obtenir ces abstractions changera d'un environnement à l'autre. C'est le cas par exemple de la détection de la présence. On ne détectera certainement pas la présence de la même manière dans une cuisine, une chambre à coucher et une salle de bain. Un service de lumière intéressé par la présence pourra pourtant disposer de cette abstraction dans toutes ces pièces.

1.2.3 Définitions des couches pour un service d'éclairage

Dans les situations décrites précédemment, on peut constater une imbrication des données contextuelles nécessaires à leur reconnaissance. Un capteur de mouvement peut-être utile pour allumer la lumière immédiatement lorsque que quelqu'un entre dans la pièce. Malheureusement, cela n'est pas suffisant pour offrir un service adapté à une personne lisant tranquillement assise sur un fauteuil. Pour cette personne, la détection de la présence est nécessaire. La détection de la présence peut s'appuyer en partie, mais pas uniquement, sur la détection de mouvement. Enfin, le fait qu'une personne soit en train de dormir peut aussi être déterminant pour savoir s'il faut allumer ou pas la lumière. La détection d'une personne endormie peut s'appuyer en partie, mais pas uniquement, sur la détection de la présence.

Avec toutes ces données, on constate donc qu'un service d'éclairage automatique doit reposer sur différents niveaux d'abstraction de données contextuelles comme suggéré par l'architecture en couches pour la construction de service sensibles au contexte. La figure 1.2 présente schématiquement les différents niveaux d'abstraction mis en avant dans les situations décrites précédem-

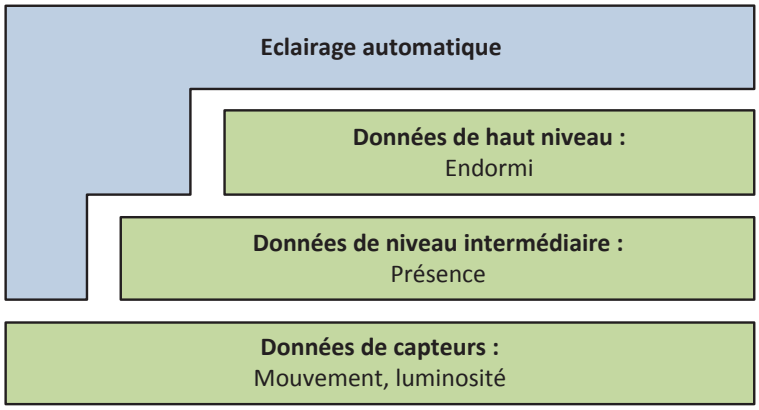

FIGURE 1.2 – Service d'éclairage automatique et niveaux d'abstraction nécessaire à sa mise en place effective.

ment adaptés au couches de l'architecture de Coutaz *et al.*. Tout en haut, le service d'éclairage automatique repose sur les trois couches offrant des données contextuelles. Les deux couches intermédiaires reposent chacune sur les données issues de la couche sous-jacente. On peut donc constater la pertinence du modèle en couches suggéré par Coutaz *et al.*.

Grâce à la description du service le plus couramment déployé dans les habitats intelligents [86], nous avons pu mettre en avant la nécessité de disposer d'un large panel de données contextuelles. Ces données se répartissent en différents niveaux d'abstraction, comme suggéré par Coutaz *et al.* dans leur architecture générique pour la mise au point de services sensibles au contexte. Ces niveaux d'abstraction sont nécessaires aux services et ils permettront aussi de déboguer et de mieux comprendre ce qui se produit dans l'habitat.

1.3 Conclusion

Dans ce chapitre, nous avons décrit quelques exemples représentatifs de la recherche dans le domaine des habitats intelligents. De ces recherches, nous avons pu déduire un ensemble de contraintes matérielles à respecter pour la construction d'un habitat intelligent réaliste : instrumentation légère, bas coût, non intrusive et aucun équipement sur les habitants. Toutes les approches présentées prennent en compte l'incertitude due à l'utilisation de capteurs. C'est une démarche que nous souhaitons suivre également dans la suite de nos travaux.

Enfin, nous avons décrit comment un service d'éclairage automatique pouvait être pensé. Nous avons pu constater la nécessité de faire reposer ce service simple *a priori* sur un large panel de données contextuelles incluant le mouvement, la présence et enfin le fait que quelqu'un dorme dans la pièce. Ces données ne peuvent pas toutes être obtenues directement via des capteurs. Elles relèvent donc de différents niveau d'abstraction. Nous avons pu constater la pertinence d'un modèle d'architecture en couches générique pour la construction de services sensibles au contexte. Cette architecture suggère que les services doivent reposer sur l'ensemble des niveaux d'abstraction de données contextuelles exactement comme nous avons pu le constater dans la

description du service d'éclairage automatique. Les abstractions utilisées doivent être stables d'un déploiement à un autre tout en reposant sur des configurations différentes de capteurs.

Si ces abstractions sont déconnectées des aspects technologiques, c'est qu'on ne peut les obtenir directement avec des capteurs. Le modèle de Coutaz *et al.* suggère donc qu'il est nécessaire de transformer les données brutes issues des capteurs pour en faire des abstractions humainement intelligibles. Il faut donc mettre en place ce que l'on appelle de la fusion de données. L'utilisation de plusieurs niveaux d'abstraction a l'avantage de permettre de construire l'ensemble d'abstractions nécessaire au système en langage naturel. On peut ainsi décrire des activités de haut niveau telles que *quelqu'un dort* à l'aide d'abstractions telles que *présence dans la chambre*, *quelqu'un est allongé*, etc sans se soucier directement de comment ces abstractions peuvent être obtenues. La fusion de données est essentielle si l'on souhaite reconnaître des situations complexes. C'est cette problématique que nous abordons dans le chapitre suivant.

Chapitre 2

Fusion de données pour la reconnaissance de contexte

Errors using inadequate data are much less
than those using no data at all.
Charles Babbage

When human judgment and big data intersect
there are some funny things that happen.
Nate Silver

Nous souhaitons être capables de récupérer des données contextuelles reliées à une maison, ses habitants et leurs activités. Pour cela, nous pouvons estimer que nous disposons d'un ensemble d'équipements augmentés, capables de communiquer leur état, mais aussi de capteurs dispersés dans la maison. Le développement de l'internet des objets rend ces technologies de plus en plus disponibles [5, 74].

Malheureusement, les capteurs et les équipements augmentés ne renvoient pas des données permettant une compréhension immédiate des situations ou des activités en cours au sein d'une habitation. Si l'on souhaite respecter l'architecture en couches suggérée par Coutaz *et al.* [19], il est nécessaire d'obtenir plusieurs niveaux d'abstraction de données contextuelles. Pour obtenir des informations de haut niveau, compréhensibles facilement par des humains ou interprétables par des services, nous avons besoin de mettre en place ce que l'on appelle de la *fusion de données.* Il en existe de nombreuses définitions mais nous ne retenons que la suivante [9] :

"Information fusion is the study of efficient methods for automatically or semi-automatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making."

Dans notre cas, cela consiste donc à abstraire des données brutes issues de capteurs ou des

données issues d'équipements augmentés pour en déduire des informations contextuelles de plus haut niveau. Par exemple, la reconnaissance d'une situation *repas* (une famille présente dans la cuisine et en train de manger) n'est pas faite par un seul capteur ni un seul équipement. On pourrait déduire cette situation à partir d'un ensemble d'indices comme l'utilisation de divers équipements de cuisine suivie de la présence de plusieurs personnes assises autour d'une table dans la salle à manger, etc. Nous cherchons donc des méthodes permettant d'effectuer ce type de raisonnement.

Dans ce chapitre, nous présentons la problématique générale de la fusion de données ainsi que les contraintes théoriques que nous nous fixons dans le cadre de la mise au point d'un habitat intelligent. Nous présentons ensuite, sans rechercher à être exhaustif, un ensemble de méthodes et théories couramment utilisées pour la fusion de données notamment dans le cadre des maisons intelligentes et de la reconnaissance d'activités.

2.1 Problématique générale

La fusion de données est un domaine de recherche à part entière [9, 49]. Il est possible, indépendamment de toute théorie mathématique, d'en caractériser les principales difficultés. Nous présentons donc dans cette section les challenges que posent la fusion de données de façon générale, ainsi que les contraintes théoriques que nous nous fixons pour le cadre spécifique qu'est l'habitat intelligent, en plus des contraintes technologiques déjà présentées au chapitre précédent. Ces contraintes vont conditionner en partie notre choix de la ou les théories sur la ou lesquelles doivent s'appuyer nos travaux.

2.1.1 Imperfection des données

La principale difficulté de la fusion de données repose dans l'imperfection des données. Ces imperfections peuvent être de différentes natures [30, 49, 62]. En effet, ces données peuvent être :

- *Aléatoires* à cause de systèmes physiques tels que les capteurs. Les valeurs retournées peuvent en effet varier y compris quand une grandeur physique mesurée est en réalité constante ;
- *Incohérentes* si des sources sont en conflit ou à cause d'un surplus d'informations ;
- *Incomplètes* notamment si des données sont perdues dans les communications comme ce peut être le cas lors des communications sans fil (voir figure 2.1) ;
- *Biaisées*, causant une erreur systématique. Cela peut être dû à la précision de l'électronique mais aussi à l'environnement qui peut être plus ou moins bruité ;
- *Incertaines* lorsque les sources ne sont pas fiables. De plus, la précision des capteurs a ses limites. Les données reccueillies ne sont pas totalement certaines ;
- *Ambiguës* à cause du langage naturel qui est utilisé pour construire les modèles par exemple ;
- *Redondantes* lorsque plusieurs sources mesurent le même paramètre.

Toutes ces imperfections doivent être prises en compte lors de la construction d'un système de fusion de données. La redondance peut être utilisées comme un avantage pour gagner en

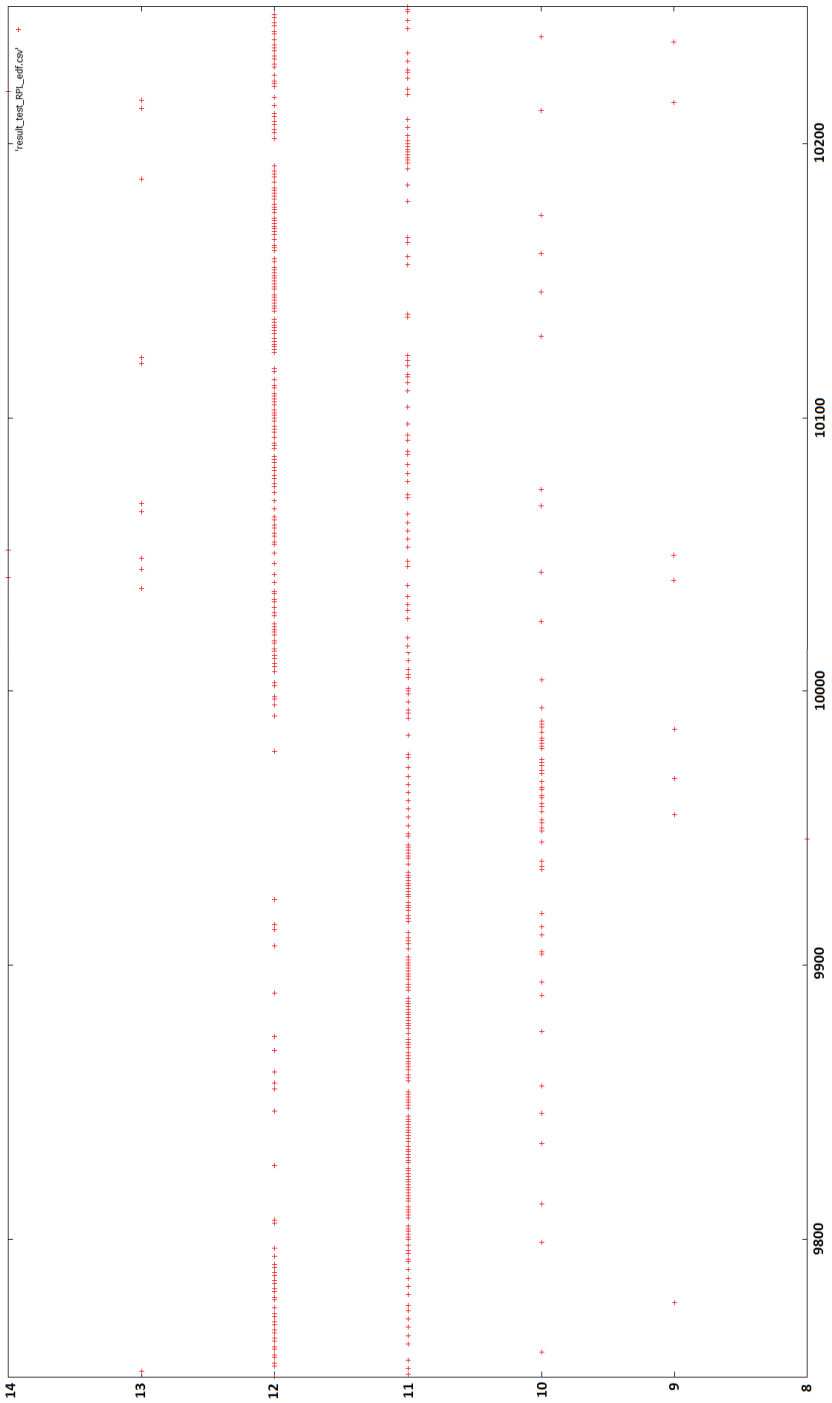

FIGURE 2.1 – Nombre de données reçues chaque seconde dans un réseau de 12 noeuds communiquant via 6LoWPAN [79] et un routage RPL[96]. Les noeuds envoient chacun un paquet par seconde. Douze paquets devraient donc être reçus à chaque fois. Il y a donc une perte chaque fois que moins de douze paquets sont reçus. On peut aussi voir un surplus de paquets arriver parfois dû au temps de propagation au sein du réseau.

précision et en fiabilité. L'ambiguïté peut, quant à elle, être réduite autant que possible dans l'élaboration des modèles.

2.1.2 Contraintes théoriques

En plus des contraintes matérielles que nous avons décrites au chapitre précédent et des imperfections des données qu'il nous faut prendre en compte, nous souhaitons prendre en compte aussi quelques contraintes théoriques sur le système. Ces contraintes découlent de la spécificité de l'habitat intelligent vis-à-vis d'autres systèmes ubiquitaires. Par exemple, il est parfaitement possible de considérer qu'un utilisateur porte toujours sur lui son smartphone dans des applications de ville intelligente (*smart city* en anglais) permettant de le guider et de lui notifier des lieux qui pourraient l'intéresser. Dans l'habitat intelligent, il est peu probable que quelqu'un soit en permanence équipé, même d'un smartphone. De même, nous ne souhaitons pas développer un habitat intelligent pour des personnes âgées ou handicapées mais pour un utilisateur quelconque qui ne sera donc pas prêt aux mêmes concessions qu'une personne ayant réellement besoin d'une assistance. Ces contraintes vont donc nous permettre de répondre aux besoins d'un habitat intelligent réaliste et acceptable par tout type d'habitants. Elles doivent également orienter nos choix pour ce qui est des méthodes et théories qu'il est possible d'utiliser dans notre problème de fusion de données.

Ignorance totale

La première contrainte à respecter est que le système doit être capable de statuer "Tout ce que je sais est que je ne sais rien". L'ignorance de ce qui se passe réellement dans la maison est intéressante car nous n'essayons pas de comprendre tous les contextes possibles (ce serait tout simplement impossible). Ainsi, il n'est pas toujours nécessaire de comprendre ce qui se passe. Cela dépend des services que l'on souhaite fournir. De même, il vaut mieux dire que l'on ne sait pas que de s'engager dans des actions qui peuvent déranger un usager ou être risquées. Il est alors possible de redonner le contrôle à l'utilisateur via une interface adaptée lorsque le système n'a aucune idée de ce qu'il doit faire. Ce peut être le cas par exemple lorsqu'une plaque chauffante est restée allumée depuis un certain temps dans la cuisine alors que personne n'a été détecté dans cette même pièce. Il est alors plus intéressant de demander confirmation à l'habitant plutôt que d'arrêter la plaque sous un plat qui peut simplement être en train de mijoter. Cela peut aussi être nécessaire s'il existe des parties non instrumentées de l'habitat comme le jardin. Ainsi, si toute la famille est dans le jardin pour un barbecue par exemple, l'habitat doit être capable de comprendre que la famille n'est pas partie et qu'il n'est pas nécessaire de fermer tous les volets roulants.

Cette contrainte très forte met de côté la majeure partie des approches probabilistes. En effet, communément dans la théorie de Bayes, le principe d'indifférence [48] est appliqué lorsqu'il manque une ou plusieurs informations. Ce principe simple attribue à chaque possibilité la même probabilité d'occurrence. Malheureusement, le résultat final du raisonnement dépend de la donnée à laquelle le principe a été appliqué, donnant lieu à de nombreux paradoxes [85]. Enfin, un niveau de confiance trop faible en ce qui est reconnu ne doit pas conduire à une décision mais bel et

bien à une indécision. L'habitat intelligent doit donc être capable de reconnaître qu'il ne sait pas ce qui se passe.

Pas ou peu d'apprentissage

L'apprentissage pose deux principaux problèmes : un coût en temps conséquent, un surapprentissage entraînant la reconnaissance de *pattern* là où il n'y en a pas. De fait, l'apprentissage s'appuie sur des analyses statistiques qui demandent de longues périodes d'expérimentations que nous souhaitons éviter pour des raisons évidentes de temps de déploiement et d'acceptation du système. De plus, des statistiques fiables sont souvent difficiles à obtenir et posent le problème de ce que l'on souhaite connaître. De fait, elles n'offrent que difficilement la possibilité d'être non-spécifiques à une étude de contexte particulier. L'apprentissage doit parfois être entièrement refait lorsque l'on change d'environnement. Ce n'est bien entendu pas acceptable dans un système censé s'adapter à de très nombreux habitats et habitants tous différents.

Pour ce qui est du surapprentissage, ce problème recoupe en partie celui d'être capable de "ne pas savoir". Pour un système, le fait même de ne pas avoir à tout reconnaître peut le mettre en échec, il n'est alors pas capable de différencier ce qui est important de ce qui ne l'est pas. Il faudrait donc être capable, pendant l'apprentissage, de détecter des situations où il n'y a rien à reconnaître. Cela va à l'encontre du principe même des systèmes reposant sur de l'apprentissage.

Enfin, les activités domestiques ne sont pas répétées parfaitement à l'identique, peuvent se croiser et être suffisamment non déterministes pour ne pas être modélisées et donc reconnues facilement [34, 88]. C'est donc un argument supplémentaire à l'encontre des systèmes d'apprentissage.

Pas de configuration prédéterminée

Dernière grande contrainte qui doit être respectée, le système ne doit pas reposer sur une configuration prédéterminée de capteurs. Dans de nombreux travaux, un ensemble de capteurs disposés sur des objets permettent de reconnaître des activités simples telles que *se préparer une boisson chaude*. Ces travaux reposent sur des méthodes *ad hoc* ne pouvant être utilisées que dans le cas d'une activité précise à reconnaître ou d'une tâche à assister [15, 43, 52, 59, 89]. Ces travaux sont particulièrement utiles dans le cadre d'habitats intelligents pour le maintien à domicile de personnes handicapées ou malades. Elles permettent une adaptabilité à un handicap ou une maladie en particulier et donc une meilleure gestion de celle-ci. Malheureusement, dans notre cas, on s'adresse à toute personne pour une assistance sur des activités très diverses. Si l'on souhaite déployer notre système dans un très grand nombres de maisons différentes, il est évident que nous ne pourrons pas le faire reposer sur un ensemble fixe d'équipements. Pour des raisons d'extensibilité mais aussi de souplesse du système, une approche plus généraliste est nécessaire. Les méthodes *ad hoc* sont donc à proscrire sauf à servir de "capteurs" élaborés.

Ces contraintes théoriques se justifient par le fait qu'on souhaite être capable de déployer notre système facilement, rapidement et dans un maximum d'habitations. Elles vont nous aider à effectuer un choix sur la théorie la plus appropriée pour le calcul des données contextuelles qui nous intéressent. On peut déjà noter que l'on rejette la plupart des approches probabilistes

comme les réseaux bayésiens [7, 42], les chaînes de Markov [46, 63] ou encore les filtres de Kalman [95]. Il nous faut donc chercher d'autres théories pour notre système.

2.1.3 Niveaux d'abstractions

Dans le chapitre 1, nous avons vu comment, à partir d'un service simple d'éclairage automatique, il est possible de penser rapidement à des abstractions de différents niveaux. Ainsi, nous avons d'abord considéré le mouvement, la présence puis enfin le fait que quelqu'un dorme dans la pièce.

Point de vue humain

Des abstractions apparaissent d'autant plus facilement lorsque l'on observe, en tant qu'humain, des situations complexes. Par exemple, on peut placer des personnes devant quelqu'un qui est debout, dans une cuisine, devant une plaque de cuisson et remuant un outil en bois dans une casserole. Si on demande à toutes ces personnes de décrire ce que fait la personne qu'elles observent, toutes ou presque répondront que cette personne cuisine. On obtiendrait certainement le même résultat si elles observaient une personne en train de casser des oeufs, de mélanger quelque chose ou encore d'utiliser le four. De même, si on demande à la personne de décrire elle-même ce qu'elle fait, elle répondra certainement qu'elle cuisine, indépendamment de l'action qu'elle est réellement en train d'effectuer.

Cuisiner est donc une abstraction de haut niveau qui peut servir à décrire de façon générique un vaste ensemble d'actions et qui décrit tout une classe d'activités. Ces abstractions ont deux principaux avantages. Tout d'abord, elles sont compréhensibles facilement par des humains puisque ce sont elles qui nous viennent en premier à l'esprit quand on doit décrire une activité. Enfin, ces abstractions sont stables, tout d'abord parce qu'elles ne dépendent pas des personnes qui les observent mais aussi parce qu'elles peuvent servir à décrire en réalité tout un ensemble d'activités ou d'actions (*cuisiner* peut par exemple correspondre soit à *casser des oeufs*, *mélanger une mixture* ou *cuire un morceau de viande* et pourquoi pas tout à la fois). Le fait qu'elles correspondent à ce qu'un humain pourrait dire en observant l'activité d'une personne facilite la modélisation et le déboggage du système mais aussi sa compréhension et son appropriation par les habitants. Le fait qu'elles couvrent toute une classe d'activités permet d'accumuler un grand nombre de preuves (plus on observe d'actions/contextes appartenant à une classe, plus on sera sûr que l'abstraction concernée correspond au monde réel) et surtout de déployer des configurations non-figées de capteurs [53].

Point de vue système

Cet aspect est très important. En effet, les activités domestiques ont la particularité de ne pas être effectuées toujours de la même manière d'une exécution à l'autre [34, 88]. Il est donc important pour le système d'être capable de fournir une abstraction stable générique sans pour autant en appréhender tous les détails lorsque cela n'est pas nécessaire. Ainsi, un service intéressé par le fait de détecter que quelqu'un cuisine n'a pas besoin de savoir quels sont les éléments sur

lesquels le système s'est appuyé pour valider cette abstraction. Ces abstractions de haut niveau permettant de décrire des activités de façon générale correspondront donc à la partie haute de l'architecture. Pour rappel, nous avons décidé de suivre l'architecture en couches suggérée par Coutaz *et al.* qui comporte quatre couches :

- *Exploitation* : cette couche exploite les données contextuelles pour fournir des services adaptés ;
- *Situation and context identification* : cette couche analyse les situations en cours et peut potentiellement prédire celles à venir ;
- *Perception* : cette première couche d'abstraction transforme les données brutes issues des capteurs en petits morceaux de contexte comme la présence ou la posture d'une personne. Les données de sortie de cette couche peuvent aussi être obtenues directement via certains appareils augmentés capables de communiquer leur statut par exemple ;
- *Sensing* : cette couche correspond aux capteurs dispersés dans l'environnement, ce sont les données de ce niveau que l'on souhaite abstraire en données compréhensibles par un humain.

Ces différentes couches d'abstraction nous semblent particulièrement pertinentes pour décrire les activités domestiques et les contextes dont nos services pourraient avoir besoin. En effet, on constate une fois de plus avec cet exemple qu'il est possible de trouver une correspondance entre les abstractions que nous avons décomposées et ce modèle en couches.

Nous devons maintenant nous interroger sur ce qui peut nous permettre d'obtenir ces abstractions. Si l'on considère l'activité *cuisiner*, on peut s'apercevoir qu'un grand nombre de preuves peuvent appuyer cette hypothèse. On peut par exemple observer une présence dans la cuisine, l'utilisation successive ou simultanée de divers appareils électroménagers utiles en cuisine, etc. Ces données sont elles-mêmes abstraites. La présence dans la cuisine par exemple peut être obtenue de diverses manières. Ainsi, des capteurs de mouvement, de pression sur des chaises ou encore l'ouverture/fermeture d'une fenêtre sont autant de preuves de la présence de quelqu'un.

De même, l'abstraction *cuisiner* pourrait servir à prédire une future situation *repas*. Ainsi, si l'on identifie qu'une personne a cuisiné, puis qu'une autre personne à mis la table et enfin que les activités se concentrent de plus en plus autour de la cuisine et de la salle à manger, alors, il devient possible de déduire qu'un repas est sûrement en train de se mettre en place. La dynamique des situations reconnues doit donc pouvoir servir à prédire des situations pour mieux assister les habitants dans leurs tâches et activités. Une situation *repas* peut par exemple permettre de détecter des oublis dans d'autres pièces laissées vides lorsque le repas a démarré.

Avec la description de ces quelques activités simples, on s'aperçoit que le découpage des données contextuelles en niveaux d'abstraction se fait naturellement. Une fois de plus, la pertinence du modèle en couches apparaît évidente. Enfin, on constate que chaque niveau d'abstraction peut reposer sur un ensemble de capteurs mais aussi et surtout sur d'autres abstractions. Chaque abstraction peut reposer sur diverses données et donc diverses configurations de capteurs. Cette flexibilité est nécessaire pour un déploiement à grande échelle du système.

2.2 Méthodes et théories

Nous avons donc besoin de différents types de données contextuelles relevant des différentes couches du modèle. Nous cherchons maintenant à identifier la ou les théories utilisées en fusion de données et en informatique ubiquitaire qui peuvent nous permettre d'obtenir ces différents niveaux d'abstractions.

Dans cette section, nous présentons succinctement un ensemble de méthodes et de théories existantes pour la fusion de données. Certaines théories présentées ici sont aussi utiles pour la représentation des données contextuelles. Elles offrent donc les outils pour à la fois modéliser et raisonner autour des contextes. Le but ici n'est pas d'être exhaustif (pour cela, voir [49]) mais plutôt de présenter les méthodes qui nous semblent les plus pertinentes pour notre projet d'habitat intelligent, en prenant comme référence les niveaux d'abstractions donnés dans l'architecture en couches précédemment présentée.

2.2.1 Reconnaissance de plan

Reconnaître des activités domestiques, par exemple *cuisiner*, peut être considéré comme étant le plus haut niveau de données contextuelles qu'il est possible d'obtenir. Pour reconnaître ces activités, il peut être intéressant de comprendre les buts des habitants et ainsi pouvoir prévoir une suite d'actions ou de situations à venir. Pour faire cela, il est possible d'utiliser des algorithmes de *reconnaissance de plan*. Il en existe de nombreux [14]. Nous en avons retenu un : PHATT [38].

PHATT repose sur un problème de planification d'un *réseau hiérarchique de tâches* (en anglais *hierarchical task network* ou HTN) qu'il doit inverser afin d'identifier les plans en cours. Ce problème de plannification consiste à générer automatiquement un plan à partir d'un ensemble de tâches à exécuter et de contraintes [37, 50].

Le but est de définir une bibliothèque de plans possibles. Chacun de ces plans est divisé en *tâches*. Ces tâches sont *primitives* si elles ne requièrent pas de plannification et peuvent donc être exécutées directement, *ouvertes* sinon. Des *méthodes* sont aussi définies afin de décrire comment les tâches peuvent être décomposées en un ensemble ordonné ou partiellement ordonné de sous-tâches (chaque tâche peut être décomposée de plusieurs manières différentes). Un algorithme de planification décompose alors récursivement l'ensemble des tâches jusqu'à n'obtenir que des tâches primitives. La figure 2.2 donne un exemple de décomposition possible pour la situation *repas*.

L'algorithme PHATT se décompose en trois phases : construire la bibliothèque de plans, modéliser les exécutions de plan et enfin reconnaître l'exécution en cours à partir d'observations [35, 36, 38]. La suite d'actions observées permet de déduire un ensemble d'actions possibles par la suite. Cet ensemble permet de générer un ensemble d'explications plausibles justifiant l'ensemble d'actions observées. Une probabilité associée à chacune de ces explications est alors calculée.

Ce type de méthode peut être particulièrement efficace pour réussir à prédire des situations et des actions à venir de la part des habitants. Cela est utile lorsque l'on souhaite mettre en place des services d'assistance à l'exécution de certaines tâches domestiques. Dans notre cas, comme décrit précédemment, cela peut permettre la prédiction de la situation *repas* suite à l'observation

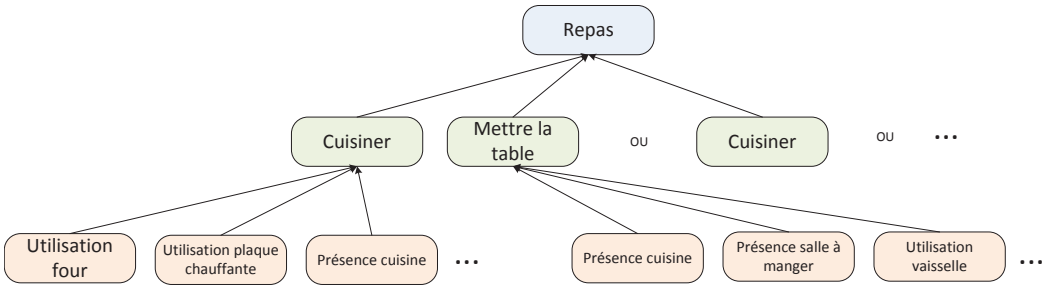

FIGURE 2.2 – Exemple de décomposition d'une situation en sous-situations et en tâches.

des situations *quelqu'un cuisine* et *quelqu'un met la table*. Le principal intérêt de PHATT repose dans le fait que la suite des évènements à observer peut ne pas être ordonnée. Ainsi, il n'est pas très important de savoir dans quel ordre le repas et la table ont été préparés. Seule la combinaison de ces situations importe.

2.2.2 Modèle de contexte

Le modèle des *Context Spaces* permet de définir simplement des contextes à partir de données brutes capturées dans l'environnement [69, 70, 68]. Plus précisément, la modélisation des *Context Spaces* repose sur trois niveaux : les capteurs, les contextes et les situations. Ainsi, les données issues de capteurs sont transformées puis interprétées afin d'obtenir une représentation factuelle du contexte mais aussi des suppositions et des prédictions [10, 12]. Un moteur de raisonnement permet ensuite de répondre à la question "telle situation est-elle en cours ?" avec un certain degré de confiance. Ce moteur de raisonnement repose sur des heuristiques qui lui sont propres mais aussi de la logique floue [100, 101] mais peut en réalité reposer sur n'importe quelle théorie de fusion de données.

La modélisation des *Context Spaces* repose sur une métaphore géométrique. Chaque capteur et chaque donnée obtenue via une fusion est appelée *attribut de contexte* (en anglais *context attributes*). Ces attributs de contexte peuvent chacun prendre un ensemble de valeurs et servent ainsi à définir les dimensions de l'espace dans lequel sont définies les situations (voir figure 2.3).

L'état courant du monde est donné par le *context state* qui correspond à un point dans l'espace défini par tous les attributs de contexte. Si cet état se trouve dans un ou plusieurs solides, alors les situations correspondantes ont toutes les chances d'être observées. Cette représentation des situations et de l'état permet donc au système de ne pas toujours savoir ce qui se passe. En effet, l'état peut ne correspondre à aucune des situations modélisées. Il est donc possible de ne modéliser que les situations pertinentes et/ou qu'il est possible de réellement identifier. L'exhaustivité n'est pas nécessaire pour appliquer cette identification de situations.

Les *Context Spaces* pourraient être utilisés pour reconnaître par exemple les situations sans forcément les prédire. Ainsi, en observant la présence dans la cuisine ainsi que l'utilisation de divers équipements permettant de cuisiner comme le four, les plaques chauffantes et ainsi de suite, il est alors possible de déduire que le repas est en train d'être préparé. Les situations ainsi

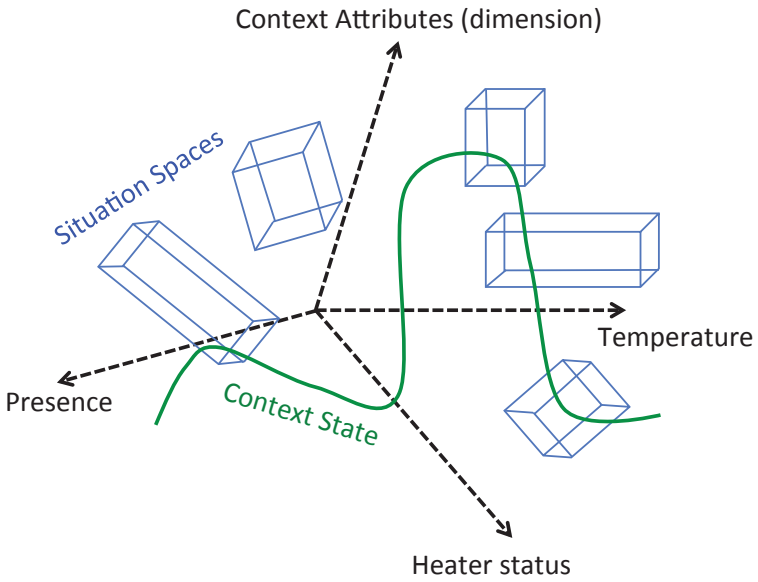

FIGURE 2.3 – Modélisation de situations dans le *framework* des *Context Spaces*. Ici, la présence, la température et l'état du radiateur sont des dimensions. Les situations sont des solides définis dans cet espace. Le *context state* qui correspond à l'état courant du monde est représenté par un point se déplaçant dans cet espace au cours du temps.

déduites peuvent ensuite servir à la prédiction de situations futures à l'aide d'un algorithme de reconnaissance de plan comme illustré dans la section précédente.

2.2.3 Logique floue

La logique floue, introduite par Zadeh [101, 100], consiste en la définition d'un degré d'appartenance à des classes. L'intérêt de cette théorie repose dans le fait de pouvoir modéliser des classes définies en langage naturel imprécis. Un ensemble flou $F \subseteq X$ est caractérisé par sa fonction d'appartenance μ_F définie par :

$$\mu_F(x) \in [0,1], \forall x \in X \tag{2.1}$$

Par exemple, il est possible de définir l'ensemble des nombres beaucoup plus grands que 1. L'ensemble flou "beaucoup plus grand que 1" est alors défini par une fonction d'appartenance $\mu_{>>1}$ que l'on pourrait définir ainsi :

$$\begin{aligned} \mu_{>>1}(10) &= 0.2 \\ \mu_{>>1}(100) &= 0.9 \\ \mu_{>>1}(1000) &= 1 \end{aligned} \tag{2.2}$$

On exprime ici le fait que 10 n'appartient que peu à la classe "beaucoup plus grand que 1" alors que 1000 y appartient complètement. Il est donc possible d'exprimer à l'aide de la logique floue

une incertitude sur l'appartenance à des classes aux contours mal définis. Chaque donnée issue d'un capteur peut ainsi se voir associer un degré d'appartenance à un contexte à reconnaître ou à un état possible de ce contexte.

Pour effectuer une fusion de données, il est alors possible de définir des règles de combinaison conjonctives ou disjonctives. Par exemple, on peut définir les règles conjonctives suivantes :

$$\mu_{\cap 1}(x) = min[\mu_{F_1}(x), \mu_{F_2}(x)], \forall x \in X \tag{2.3}$$

$$\mu_{\cap 2}(x) = \mu_{F_1}(x).\mu_{F_2}(x), \forall x \in X \tag{2.4}$$

Et il est possible de définir les règles disjonctives suivantes :

$$\mu_{\cup 1}(x) = max[\mu_{F_1}(x), \mu_{F_2}(x)], \forall x \in X \tag{2.5}$$

$$\mu_{\cup 2}(x) = \mu_{F_1}(x) + \mu_{F_2}(x), \forall x \in X \tag{2.6}$$

Les règles conjonctives peuvent être utilisées pour fusionner des données fiables et les règles disjonctives peuvent être appliquées en cas de conflit ou de données peu fiables.

On peut par exemple définir la température actuelle dans une pièce avec des expressions floues comme par exemple "il fait froid" et "il fait chaud" qu'il est possible de moduler en exprimant "il fait un peu froid" ou "il fait très froid". Chaque expression dépend bien entendu d'une température idéale qui peut dépendre de chaque habitant et le flou autour de "très froid" et "un peu froid" peut être modélisé à l'aide de fonctions d'appartenance.

La logique floue est donc particulièrement efficace lorsqu'on cherche à travailler avec des classes mal définies comme cela peut être le cas lorsqu'on exploite des données en langage naturel. Bien évidemment, nous essayerons de définir des classes strictes plutôt que des ensembles flous. Donc même si cette approche est intéressante, nous ne l'utiliserons pas *a priori* pour effectuer la fusion de données nous permettant de reconnaître des situations et des activités. Elle pourra par contre éventuellement être exploitée à bas niveau pour interpréter des données de capteurs, c'est-à-dire passer de données objectives à des abstractions définies en langage naturel.

2.2.4 Fonctions de croyance

La théorie des fonctions de croyance, aussi appelée théorie de Dempster-Shafer [21, 78], repose sur l'accumulation de preuves. Les preuves sont représentées par des fonctions de masses définies par :

$$\begin{aligned} m : 2^{\Omega} &\longmapsto [0,1] \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned} \tag{2.7}$$

Ces fonctions de masse permettent de modéliser à la fois l'incertitude en associant des degrés de croyance à des états possibles mais aussi l'imprécision en associant ces degrés de croyance à des ensembles d'états possibles.

Les applications de cette théorie sont multiples [90] et elle offre d'excellents outils de fusion

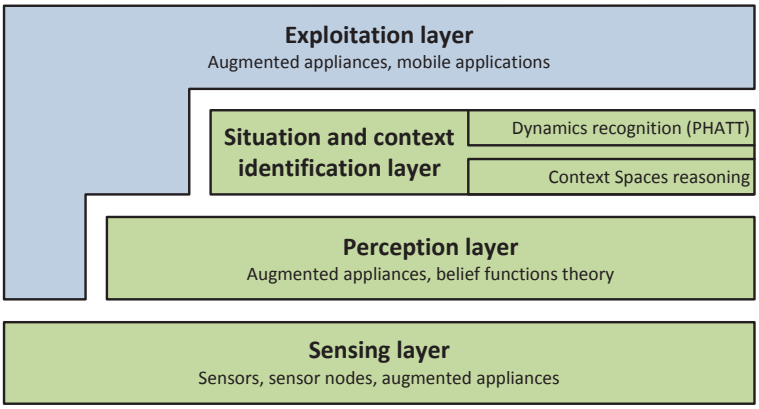

FIGURE 2.4 – Modèle en couches adoptée pour l'architecture de notre habitat intelligent.

mais aussi de gestion des erreurs et des fautes [76]. Les bases de la théorie ainsi que son application à la fusion de données issues de capteurs hétérogènes sont présentées en détail au chapitre suivant.

2.3 Fusion de données et architecture en couches

Nous avons passé en revue un ensemble de théories permettant le calcul de données contextuelles à partir de données de capteurs. Toutes ces théories proposent des modèles de contexte et/ou des algorithmes de fusion de données différents, chacun avec ses avantages et ses inconvénients. Nous souhaitons être capables de fournir des données contextuelles de différents niveaux d'abstraction comme suggéré par Coutaz *et al.* dans leur architecture en couches. De nombreux travaux appuient le fait qu'aucune des théories utilisées en fusion de données ne permet, à elle seule, de répondre aux problèmes généraux de modélisation et de fusion de données à tous les niveaux [8, 20]. Ils incitent donc à l'utilisation de plusieurs méthodes et théories à la fois.

Pour notre architecture d'habitat intelligent, nous avons donc décidé d'adapter le modèle en couches de Coutaz *et al.* comme montré figure 2.4.

La couche supérieure a été divisée en deux sous-couches. La première, assurée par la théorie des *Context Spaces* permet la modélisation et la reconnaissance de situations en cours. La deuxième, assurée par l'algorithme de reconnaissance de plan PHATT, permet la reconnaissance d'activités en cours et la prédiction de situations.

La reconnaissance de plan nous semble particulièrement adaptée à la reconnaissance d'activités. En effet, ces dernières peuvent souvent se décomposer en un ensemble d'actions et de situations successives. La description en langage naturel des activités offre donc le modèle nécessaire aux algorithmes de reconnaissance de plan. Dans notre cas, il repose donc en partie sur des situations haut niveau comme le ferait un humain décrivant des activités.

La théorie des *Context Spaces* a été créée spécifiquement pour la reconnaissance de situations. Malheureusement, ces situations ne sont pas toujours faciles à décrire en termes de capteurs. C'est pourquoi, nous avons décidé de faire reposer la théorie sur un premier niveau d'abstraction que l'on appellera ici *attributs de contexte*.

Ces attributs de contexte, qui correspondent donc à des "petits morceaux de contexte" comme la présence ou l'état d'un appareil, sont obtenus directement via des appareils augmentés ou via la fusion de données issues de capteurs. Pour cette dernière, nous avons retenu la théorie des fonctions de croyance pour sa capacité à modéliser à la fois l'imprécision et l'incertitude mais aussi parce qu'il existe de nombreux mécanismes pour prendre en compte et corriger les fautes et les erreurs des capteurs [44, 64, 76].

Enfin, toutes ces méthodes offrent des moyens de modéliser l'incertitude soit par des degrés de croyance, soit par des niveaux de confiance ou encore des probabilités. Il est donc possible à chaque niveau soit de prendre une décision sur l'état considéré comme étant l'état réel du monde et de renvoyer un couple (état, certitude) aux couches supérieures soit de renvoyer un ensemble de couples (état, certitude) afin de permettre aux couches supérieures de considérer toutes les possibilités.

En procédant à un tel découpage, nous pouvons fournir aux couches les plus hautes des abstractions stables valides dans tous les déploiements. Il n'est donc pas nécessaire, *a priori*, de revoir les modèles de contexte à chaque déploiement. De plus, le fait de raisonner sur des abstractions de haut niveau facilite grandement la mise au point mais aussi la compréhension du système. Cela facilite donc le déboggage mais aussi l'acceptation par les habitants. En effet, la compréhension de ce qui se passe et la possiblité de comprendre les problèmes lorsqu'ils se présentent sont cruciales pour éviter les frustrations [86]. Enfin, en raisonnant sur des abstractions de haut niveau, nous réduisons drastiquement le nombre de données traitées à chaque niveau, réduisant ainsi la complexité des algorithmes utilisés.

2.4 Conclusion

Nous avons présenté dans ce chapitre les difficultés rencontrées de façon générale lors de la mise en place de mécanismes de fusion de données. Nous nous sommes aussi fixés des contraintes théoriques répondant aux besoins spécifiques de l'habitat intelligent. Cela nous a permis de sélectionner une partie des méthodes existantes. Nous avons ensuite présenté succinctement le principe des algorithmes de reconnaissance de plan, des *Context Spaces*, de la logique floue et des fonctions de croyance.

Nous proposons de conserver trois de ces théories afin d'adapter l'architecture en couches suggérée par Coutaz *et al.* pour le calcul de données contextuelles. Les théories retenues s'adaptent naturellement aux niveaux d'abstraction à fournir et permettent de propager l'incertitude au travers de toutes les couches de l'architecture.

Les niveaux d'abstraction sur lesquels nous faisons reposer les méthodes de fusion de données ont l'avantage de faciliter la modélisation des contextes en permettant l'utilisation d'abstractions stables définies en langage naturel. Ces abstractions permettent bien évidemment de faciliter le déboggage du système par leur facilité d'interprétation par un humain. Elles permettent aussi de réduire nettement le nombre de données traitées à chaque niveau et entraîne donc par la même occasion une diminution de la complexité des algorithmes mis en place.

Enfin, chaque niveau d'abstraction possède ses propres échelles physique et temporelle. Ainsi, les capteurs renvoient des données à très court terme dans un espace physique limité. Les abstrac-

tions de la couche perception peuvent s'installer sur quelques secondes à l'échelle d'une pièce. Les situations peuvent durer quelques minutes et toucher un ensemble de pièces de la maison. Enfin, les activités peuvent concerner la maison dans son ensemble et s'installer sur plusieurs heures. Bien entendu, tout cela n'est pas figé de façon stricte mais cela permet tout de même d'avoir une bonne estimation du niveau d'abstraction correspondant à chaque donnée contextuelle qui nous intéresse.

Dans un souci de rendre notre habitat intelligent le plus acceptable possible pour les habitants et réaliste en termes d'instrumentation, de déploiement et de coût, nous avons identifié un ensemble de contraintes à respecter. Tout d'abord des contraintes matérielles : instumentation légère et bas coût, pas d'instrumentation des habitants et enfin pas d'équipements trop intrusifs. Enfin des contraintes théoriques : la possibilité de ne pas savoir ce qui se passe et donc redonner le contrôle à l'utilisateur, ne pas faire reposer la reconnaissance de contexte sur des méthodes d'apprentissage afin d'éviter des temps de déploiement trop long ainsi qu'un surapprentissage dû à la non constance des activités domestiques et enfin ne pas faire de la reconnaissance de contexte de façon *ad hoc* pour des raisons évidentes de flexibilité et de mise à l'échelle.

En résumé, nous cherchons à réaliser de la fusion de données de façon générique pour un grand ensemble de services possibles. Chaque donnée contextuelle devra pouvoir être obtenue avec différentes configurations de capteurs et d'équipements. Chacune de ces données pourra être utilisée par plusieurs services à la fois. Tout cela doit fonctionner avec des équipements bas coût, peu fiables et dont les capacités sont fortement limitées.

Cette architecture globale est ambitieuse. Dans la suite du document, nous concentrons nos travaux essentiellement sur les couches basses de l'architecture (*sensing* et *perception*). En effet, si l'on souhaite pouvoir faire de la reconnaissance de situations et d'activités en nous reposant sur des abstractions stables, il nous faut tout d'abord les obtenir. Dans les chapitres suivants, nous montrons comment la théorie des fonctions de croyance nous permet de les obtenir tout en répondant à toutes ces contraintes. Nous proposons également des améliorations à cette théorie afin d'augmenter la stabilité des abstractions (les attributs de contexte obtenus). Enfin, nous présentons comment ce système de fusion de données a été implémenté et intégré dans un prototype complet respectant le modèle en couches.

Deuxième partie

Théorie des fonctions de croyance : applications et stabilité

Chapitre 3

Théorie des fonctions de croyance : application aux capteurs

It's raining outside but I don't believe that it is.
G.E. Moore

Croyance : milieu entre l'opinion et le savoir.
E. Kant

Dans les chapitres précédents, nous avons mis en avant le besoin de mettre en place une méthode de fusion de données. Plus précisément, nous avons besoin, à partir de données brutes issues de capteurs, d'extraire des informations contextuelles. Dans le cadre de cette thèse, nous nous limitons à des éléments de contexte que nous appelons *attributs de contexte*. Ces attributs sont simples et ne représentent que de "petits" morceaux de contexte comme par exemple la présence dans une pièce, la posture d'une personne, etc.. Dans la suite de notre travail, nous nous intéressons donc aux couches basses de l'architecture présentée dans le chapitre précédent : il s'agit d'exploiter un ensemble de capteurs élémentaires distribués dans l'habitat et de fusionner les données qu'ils produisent dans une première couche d'abstraction.

Il existe de nombreuses théories et méthodes utilisées en fusion de données [8, 80]. Nous avons décidé d'utiliser la théorie des fonctions de croyance [21, 78] (aussi appelée théorie de Dempster-Shafer). Cette théorie fonctionne par accumulation de preuves sur ce que l'on cherche à observer. Elle est utilisée dans de nombreux domaines tels que la reconnaissance de formes, l'aide à la décision, le traitement d'images et, bien entendu, la perception multi-capteurs [90]. Les nombreux avantages de cette théorie dans notre cas sont exposés tout au long de ce chapitre, au fur et à mesure que les outils sont introduits.

L'enjeu de ce chapitre est de montrer comment cette théorie peut être appliquée dans le cadre d'un habitat faiblement instrumenté en s'appuyant sur un ensemble "raisonnable" de capteurs

élémentaires imprécis et peu fiables. Nous présentons donc les bases de la théorie des fonctions de croyance puis comment mettre en place cette théorie et comment l'appliquer à des capteurs afin d'obtenir des attributs de contexte. Ce chapitre se conclue par un exemple complet d'application de la théorie pour la détection de présence dans une pièce.

3.1 Modéliser des croyances

Comme son nom l'indique, la théorie des fonctions de croyance repose sur des croyances. Ces croyances représentent en réalité la connaissance limitée qu'on peut avoir de l'environnement qu'on observe. Dans notre cas, nous observons la maison et ses habitants à l'aide de capteurs offrant une vision partielle et imprécise de leur environnement. Les croyances vont donc nous servir à exprimer l'incertitude et l'imprécision des données issues des capteurs.

3.1.1 Cadre de discernement

La première chose à définir pour pouvoir construire des croyances est l'ensemble des mondes (ou états) possibles, que l'on appelle *cadre de discernement*, pour chacun des attributs de contexte que l'on souhaite calculer. Il est généralement noté :

$$\Omega = \{\omega_1, \omega_2, ..., \omega_n\} \tag{3.1}$$

ω_i représente un monde possible et n le nombre de mondes définis. Par exemple, pour les attributs de contexte *présence* et *posture*, on peut définir deux cadres de discernement :

$$\begin{aligned} \Omega_{présence} &= \{Oui, Non\} \\ \Omega_{posture} &= \{Assis, Debout, Allongé\} \end{aligned}$$

Les mondes ainsi définis doivent être *exclusifs*, c'est-à-dire qu'ils ne se recoupent pas, un seul d'entre eux représente l'état réel du monde à un moment donné (*i.e.* $\omega_i \cap \omega_j = \emptyset$ *si* $i \neq j$). Enfin, le cadre de discernement doit être si possible *exhaustif*, c'est-à-dire que l'état réel du monde a été défini et se trouve nécessairement dans le cadre de discernement.

3.1.2 Fonction de masse

Maintenant que nous pouvons définir des cadres de discernement pour représenter les états possibles de ce que l'on cherche à observer, nous voulons représenter des croyances vis-à-vis de ces états. Pour cela, il est possible de définir une *fonction de masse*, notée m, représentant le degré de croyance associé à chaque sous-ensemble de mondes possibles. Elle est définie par :

$$\begin{aligned} m : 2^{\Omega} &\longmapsto [0, 1] \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned} \tag{3.2}$$

Il est important de noter que cette fonction n'est pas définie directement sur le cadre de discernement Ω mais sur l'ensemble des disjonctions des éléments de Ω noté 2^Ω et appelé *ensemble puissance* (2^Ω peut-être vu comme l'ensemble des sous-ensembles de Ω). Il est défini comme suit :

$$2^\Omega = \{\emptyset, \{\omega_1\}, \{\omega_2\}, \{\omega_1 \cup \omega_2\}, \{\omega_3\}, \{\omega_1 \cup \omega_3\}, \{\omega_1 \cup \omega_2 \cup \omega_3\}, ..., \{\omega_1 \cup ... \cup \omega_n\}\} \tag{3.3}$$

Les sous-ensembles $A \subseteq \Omega$ pour lesquels $m(A) > 0$ sont appelés *éléments focaux*. Lorsqu'un élément focal A contient plusieurs mondes possibles, le degré de croyance (ou masse) $m(A)$ accordé à cet ensemble de mondes possibles ne peut pas être distribué entre ces différents mondes. Il représente une indécision entre ces différents mondes. La masse est donc entièrement accordée au fait que le monde réel se trouve parmi ces mondes possibles sans plus de précision.

Reprenons l'exemple de la posture. L'ensemble puissance pour le cadre de discernement $\Omega_{posture}$ est le suivant :

$$\begin{aligned} 2^{\Omega_{posture}} = \quad & \{\emptyset, \{Assis\}, \{Debout\}, \{Allongé\}, \\ & \{Assis \cup Debout\}, \{Assis \cup Allongé\}, \\ & \{Debout \cup Allongé\}, \Omega_{posture}\} \end{aligned}$$

Il est alors possible de définir une croyance sur la posture d'une personne. Par exemple :

$$\begin{aligned} m_{posture}(\{Assis\}) &= 0.5 \\ m_{posture}(\{Assis \cup Allongé\}) &= 0.3 \\ m_{posture}(\Omega_{posture}) &= 0.2 \end{aligned} \tag{3.4}$$

Dans cette fonction de masse, on accorde la moitié de la croyance au fait que la personne est assise. 30% de la croyance est accordée de façon imprécise entre les états *Assis* et *Allongé*. Enfin, 20% de la croyance est accordé à ce que l'on appelle l'*ignorance totale* et définie par $m(\Omega)$. Dans cet exemple, on voit donc qu'il est possible de représenter à la fois l'incertitude avec des degrés de croyance et l'imprécision avec des ensembles de mondes possibles non-atomiques (*i.e.* $|A| > 1$).

Autre fait intéressant, contrairement aux probabilités, les masses sont non-additives. Ainsi, si l'on connaît la masse accordée à deux états, il n'est pas possible d'en déduire la masse accordée à l'union de ces deux états. Une masse n'est donc accordée à une union de mondes possibles que dans le cas d'une réelle indécision entre ces états.

3.1.3 Cas particuliers

Les fonctions de masse sont les éléments de base de la théorie des fonctions de croyance. Il existe certains cas particuliers de fonctions de masse que l'on retrouve régulièrement dans les applications. Nous présentons ici les plus courants.

Fonction de masse vide

Dans l'exemple donnné en (3.4), on peut remarquer que de la masse est accordée à l'ensemble complet des mondes possibles. Cette masse, appelée ignorance totale peut permettre au système d'exprimer le fait qu'il ne sait pas du tout ce qui se passe. Cela s'exprime à l'aide de *la fonction de masse vide* comportant pour unique élément focal Ω (*i.e.* $m(\Omega) = 1$).

Fonction de masse catégorique

Une fonction de masse est dite *catégorique* si, comme son nom l'indique, elle ne contient qu'un seul élément focal. Elle s'exprime donc sous la forme suivante :

$$m(A) = 1,\ A \subseteq \Omega \tag{3.5}$$

La fonction de masse vide est un cas particulier de fonction de masse catégorique.

Fonction de masse non-dogmatique

Une fonction de masse est dite *non-dogmatique* lorsqu'elle contient une part d'ignorance totale (*i.e.* $m(\Omega) > 0$). Ces fonctions évitent des cas de conflit total lors de la combinaison des fonctions de masse.

Fonction de masse simple

Une *fonction de masse simple* n'a que deux éléments focaux, un élément focal quelconque et Ω. Elle a donc la forme suivante :

$$\begin{aligned} m(A) &= p,\ A \subset \Omega \\ m(\Omega) &= 1 - p \end{aligned} \tag{3.6}$$

Fonction de masse consonante

Une fonction de masse est dite *consonante* si ses éléments focaux sont imbriqués les uns dans les autres. Ainsi, si $\mathcal{F}$ désigne l'ensemble des éléments focaux d'une fonction de masse, on a alors :

$$\forall A \in \mathcal{F},\ A_0 \subset A_1 \subset ... \subset A_n \tag{3.7}$$

Ces fonctions de masse ont la particularité de n'avoir aucun conflit interne car l'intersection de deux de leurs éléments focaux n'est jamais vide (*i.e.* $\forall A, B \in \mathcal{F},\ A \cap B \neq \emptyset$). Ces fonctions permettent notamment de représenter des possibilités [28] au sein de la théorie des fonctions de croyance.

Fonction de masse non-normalisée

Dans le modèle des croyances transférables de Smets [84], la masse sur l'ensemble vide peut ne pas être nulle (*i.e.* $m(\emptyset) \neq 0$). Dans ce cas, la fonction de masse est dite *non-normalisée*.

La masse accordée à l'ensemble vide peut être interprétée comme correspondant à un monde non-défini dans le cadre de discernement. Dans le cadre de cette théorie, les fonctions de masse non-normalisées correspondent donc à l'hypothèse de *monde ouvert* pour lequel tous les états possibles n'ont pas été définis.

Fonction de masse probabiliste

Il est possible de représenter des probabilités d'occurrence des états possibles du monde à l'aide d'une fonction de masse. Pour cela, chaque élément focal doit être atomique. On a donc :

$$m(A) > 0 \Rightarrow |A| = 1 \tag{3.8}$$

Il est donc possible de représenter à la fois de réelles croyances, des possibilités et des probabilités à l'aide des fonctions de masse.

3.2 Construire des fonctions de masse

Nous avons vu comment représenter des croyances mais cette représentation ne donne pas de méthode pour les construire. Il n'existe pas qu'une seule manière de construire les fonctions de masse. Ainsi, on peut trouver dans la littérature des méthodes reposant sur les statistiques [4], des densités de probabilité [3] ou encore des méthodes de classification [22, 75]. Toutes ces méthodes présentent des avantages et des inconvénients mais surtout, elles ne sont pas toujours facile à mettre en œuvre.

Dans cette section, nous nous intéressons aux méthodes simples nous permettant de construire des fonctions de masse à partir de données fournies par des capteurs.

3.2.1 Projection simple

Dans notre étude, nous souhaitons abstraire des données brutes pour en retirer des données contextuelles, des attributs de contexte. Ces données sont normalement compréhensibles par des humains. C'est le cas des abstractions *Présence* et *Posture* par exemple. Nous allons donc chercher à mesurer ces attributs de contexte à l'aide de capteurs.

Prenons l'exemple de la présence. Un capteur couramment utilisé pour la détection de présence est le capteur de mouvement. Une méthode très simple pour détecter la présence à partir de ce capteur consiste à considérer la plage de mesures de ce capteur et à la diviser en deux parties : une qui indique "oui, il y a quelqu'un" et une autre qui indique "non, il n'y a personne". On crée donc une projection (*mapping* en anglais) simple entre les valeurs de notre capteur et les états possibles donnés dans le cadre de discernement.

Imaginons donc que l'on ait un capteur de mouvement qui retourne une valeur entre 0 et 1000, avec 0 correspondant à aucun mouvement détecté et 1000 correspondant à "beaucoup" de mouvement détecté. Les valeurs intermédiaires correspondent alors à des "quantités" de mouvement intermédiaires. On peut par exemple définir deux plages : $[0 - 100]$ correspond à aucun

mouvement détecté et $[101 - 1000]$ correspond à un mouvement détecté. [1]

La première solution exposée consiste à construire, à l'aide d'une projection de chaque mesure donnée par le capteur, une fonction de masse catégorique sur notre cadre de discernement correspondant à la présence (pour rappel : $\Omega_{présence} = \{Oui, Non\}$). Par exemple, si notre capteur retourne 500, on aura alors la fonction de masse suivante :

$$m(\{Oui\}) = 1 \tag{3.9}$$

On a donc créé une correspondance entre les valeurs retournées par notre capteur de mouvement et les états possibles de notre cadre de discernement $\Omega_{présence}$. Cette solution très simple n'est pas vraiment acceptable. En effet, nos capteurs ne sont pas infaillibles et la fonction de masse résultante n'offre aucune incertitude. Nous n'exploitons pas, avec cette méthode, les avantages de l'expressivité des fonctions de croyance.

3.2.2 Affaiblissement

Une opération couramment utilisée pour prendre en compte les erreurs possibles des capteurs est l'*affaiblissement* [43, 52, 58, 103]. Elle est donnée par :

$$m_\alpha(A) = \begin{cases} (1-\alpha).m(A) \text{ si } A \subset \Omega \\ (1-\alpha).m(A) + \alpha \text{ si } A = \Omega \end{cases} \text{ avec } \alpha \in [0,1] \tag{3.10}$$

Cette opération, pour un facteur α donné, transfert une partie de la masse de tous les éléments focaux sur l'ignorance totale ($m(\Omega)$). Le plus souvent, α est calculé à partir du taux de défaillance du capteur. Ce taux peut bien évidemment évoluer avec le temps pour des raisons d'usure ou encore de variation du voltage fourni par des batteries. Ce facteur peut aussi être calculé via un ensemble de fonctions de masse servant à mesurer le même attribut de contexte pour, par exemple, affaiblir les preuves en fonction de leur désaccord les unes avec les autres [76].

Pour notre capteur de mouvement, s'il a un taux d'erreur de 5%, on peut alors appliquer un affaiblissement pour le prendre en compte. Ainsi, la fonction de masse (3.9) devient :

$$\begin{aligned} m(\{Oui\}) &= 0.95 \\ m(\Omega_{présence}) &= 0.05 \end{aligned} \tag{3.11}$$

Cette opération d'affaiblissement nous permet donc de tenir compte de l'incertitude sur les mesures retournées par nos capteurs. Malheureusement, cette donnée n'est pas toujours fournie par les constructeurs. Estimer les taux d'erreur dans le cadre de nos expériences est une tâche complexe qui requiert énormément d'expérimentations et qui prend du temps. De plus, les taux d'erreur peuvent dépendre des voltages auxquels les capteurs doivent être alimentés et donc des

1. Il est possible de construire des croyances sur la mesure retournée par le capteur en discrétisant la plage de mesures de celui-ci pour en faire un cadre de discernement. Bien entendu, l'ensemble puissance résultant sera très grand et pourra donc poser des problèmes de calculabilité si on ne procède pas à quelques optimisations élémentaires des calculs.

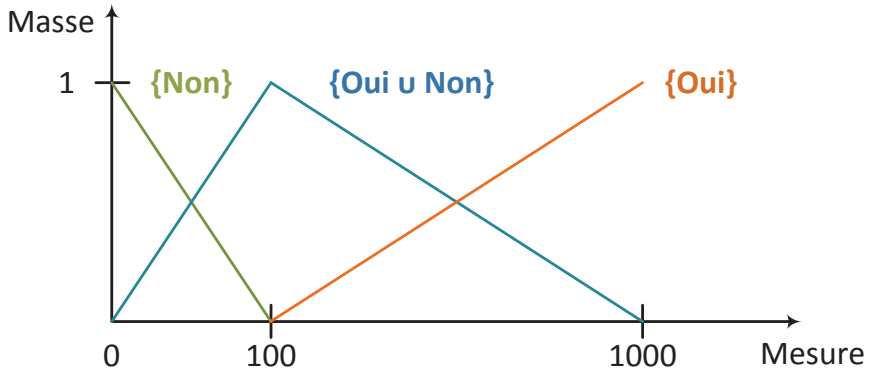

FIGURE 3.1 – Ensemble de fonctions de masse associé à un capteur de mouvement et à la détection de présence.

équipements auxquels ils sont attachés. Donc, si dans les faits cette opération peut s'avérer très utile, sa mise en œuvre n'est pas toujours possible.

Il est aussi possible d'appliquer un affaiblissement pour prendre en considération la précision des mesures de nos capteurs. Mais pour cela, il serait en fait préférable de travailler directement sur la valeur retournée par le capteur plutôt que sur sa projection sur un cadre de discernement. Cela est possible avec les fonctions de croyance appliquées aux nombres réels [82].

3.2.3 Ensemble de fonctions de masse

La projection simple de plages de mesures sur un cadre de discernement ne permet pas une expressivité fine de ce qui est réellement observé. En effet, nos capteurs ne sont pas tous binaires et nous donnent donc plus d'informations qu'un simple "oui ou non". Les ensembles flous [101] nous permettent de modéliser ce type d'informations.

Dans le cas de notre capteur de mouvement, sur les deux plages définies, on peut alors définir un degré d'appartenance aux classes $\{Oui\}$ et $\{Non\}$ en fonction de la mesure réellement retournée. On peut alors définir deux fonctions d'appartenance linéaires définies par :

$$\begin{aligned} f_{\{Non\}}(0) &= 1 \\ f_{\{Non\}}(100) &= 0 \\ f_{\{Oui\}}(100) &= 0 \\ f_{\{Oui\}}(1000) &= 1 \end{aligned} \tag{3.12}$$

Si l'on souhaite transformer cet ensemble de fonctions floues en un ensemble de fonctions de masse, il faut respecter la contrainte de somme imposée aux fonctions de masse (*i.e.* $\sum_{A\subseteq\Omega} m(A) = 1$). On peut donc définir une autre fonction floue pour la classe $\{Oui \cup Non\}$ afin de compléter les deux autres. On obtient alors l'ensemble des fonctions de masse représenté figure 3.1. Pour obtenir la fonction de masse correspondant à chaque mesure, il suffit alors de projeter la mesure

sur cet ensemble. Par exemple, si le capteur retourne 90, on obtient une fonction de masse à deux éléments focaux : $m(\{Non\}) = 0.1$ et $m(\Omega_{présence}) = 0.9$. Avec cet ensemble, on a donc une croyance mitigée lorsqu'un léger mouvement est détecté. Ceci est représenté par une ignorance totale non nulle et même maximale à la limite entre les deux plages que nous avions définies intialement.

Dans notre cas, nous souhaitons calculer des attributs de contexte. Il faudra donc construire pour chacun de ces attributs un ensemble de fonctions de masse tel que présenté précédemment pour chaque capteur pouvant servir de preuve. Pour le moment, nous construisons ces ensembles sur la base de quelques expérimentations rapides ainsi qu'une part d'intuition.

Bien entendu, il faut faire attention à respecter quelques contraintes élémentaires comme celle sur la somme des masses mais aussi le *principe de moindre engagement* (*least commitment principle* en anglais). Ceci peut se traduire dans notre cas par le fait qu'une croyance ne doit pas être trop spécifique quand elle ne peut pas l'être. Dans l'exemple de la détection de présence, le capteur de mouvement ne devrait pas pouvoir être une preuve de l'absence. En effet, il est possible de lire assis dans un fauteuil, être donc présent dans une pièce, sans qu'aucun mouvement ne soit pour autant détecté.

Ces ensembles de fonctions de masse sont utiles pour expérimenter rapidement des capteurs comme preuves sur des attributs de contexte. De plus, ils peuvent être réglés pour donner des résultats satisfaisants après seulement quelques expérimentations. Cette méthode requiert donc moins de temps que des méthodes reposant sur les statistiques mais est bien évidemment moins fiable.

3.2.4 Classification

Afin d'automatiser une partie du travail d'ingénierie nécessaire à la création des modèles de croyance requis pour calculer les attributs de contexte, il est possible d'utiliser des algorithmes de classification. Un algorithme couramment utilisé pour faire de l'apprentissage non supervisé est la méthode des K voisins les plus proches (*k-nearest neighbors algorithm* en anglais) [22]. Cet algorithme s'applique à un ensemble de données pour lesquelles une distance a été définie. Sur la figure 3.2, on peut voir un exemple de nuage de points classés en deux ensembles distincts. Pour appliquer cela à des capteurs, il est alors possible de prendre les mesures relevées lors d'expérimentations. Ces mesures sont alors réparties sur un espace à une dimension correspondant à la plage de valeurs possibles du capteur. Une supervision est alors nécessaire pour indiquer le nombre de classes à construire ainsi que leur nom une fois celles-ci construites.

Malheureusement, cette classification simple revient à faire la simple projection que nous avions faite dans un premier temps. Si l'on souhaite introduire un degré d'appartenance à la classe pour chaque nouvelle mesure, il est alors possible de considérer la distance de cette mesure par rapport au centroïde de la classe dans laquelle elle se situe. Ainsi, plus une mesure sera proche du centroïde d'une classe, plus son degré d'appartenance à cette classe sera grand et inversement lorsqu'elle s'en éloigne. Une fois de plus, on peut résoudre la contrainte sur la somme des masses en complétant avec de l'ignorance totale. Cela revient en fait à affaiblir une fonction de masse catégorique construite à partir de la classification classique.

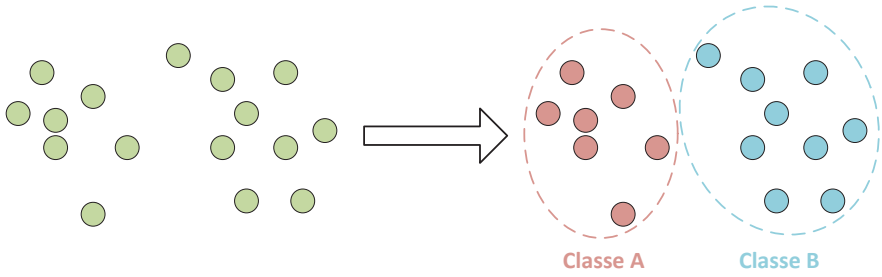

FIGURE 3.2 – Nuage de points classés en deux ensembles distincts.

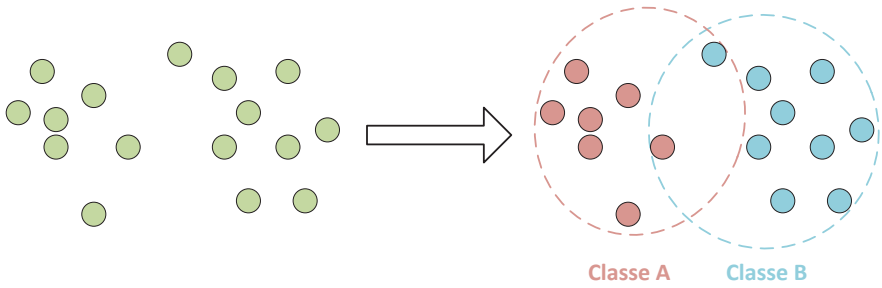

FIGURE 3.3 – Nuage de points classés en deux ensembles dont l'intersection n'est pas nulle.

Reprenons notre exemple d'ensemble de fonctions de masse pour un capteur de mouvement associé à la détection de présence. Sur la figure 3.1, nous pouvons observer un point en particulier où l'ignorance totale est maximale (lorsque le capteur retourne 100). Ce point représente donc un cas où les deux classes sont possibles et où il n'est pas possible de décider franchement entre les deux. Si l'on souhaite obtenir ce même genre d'unions de mondes possibles à l'aide d'algorithmes de classification, il faut utiliser un algorithme de classification recouvrante [17, 75]. La figure 3.3 donne un exemple de classification recouvrante. Ainsi, les deux ensembles ont quelques points en commun. Concrètement, cela se traduit par des éléments focaux non-atomiques (*i.e.* $|A| > 1$) dans les fonctions de masses résultantes.

Ce type de recouvrement peut s'avérer particulièrement utile dans certains cas. Par exemple, si on cherche à calculer la posture d'une personne et qu'on dispose d'un capteur de pression sur un lit, le fait qu'un certain poids soit mesuré peut résulter aussi bien d'une personne allongée que d'une personne assise sur le lit. Il est donc normal d'accorder une croyance à l'union de ces deux états.

3.2.5 Analyse

Différentes approches sont possibles pour construire des fonctions de masse associées à des capteurs élémentaires. La plus simple consiste à construire des ensembles de fonctions de masse

manuellement en s'appuyant sur des expérimentations rapides. Cela permet d'analyser les principales valeurs retournées par un capteur. C'est cette approche élémentaire que nous retenons pour la partie expérimentale de nos travaux.

Il est bien entendu possible d'appliquer un affaiblissement pour prendre en compte les taux d'erreur des capteurs. Nous n'avons pas souhaité mettre en œuvre ce mécanisme dans nos travaux dans la mesure où les données nécessaires pour caractériser ces taux d'erreur sont très rarement disponibles pour les capteurs très basiques que nous utilisons. De même, si les méthodes de classification nous paraissent prometteuses, nous ne pouvons pas disposer de suffisamment de données expérimentales pour appliquer correctement ces algorithmes pour le moment.

3.3 Accumuler des preuves

Maintenant que nous pouvons construire des croyances à partir de données de capteurs, nous souhaitons fusionner l'ensemble de ces preuves recueillies. L'accumulation de preuves a d'autant plus de sens dans la reconnaissance de contexte qu'il n'existe par une unique manière de détecter chaque contexte [53]. L'accumulation de preuves se fait à l'aide de règles de combinaison. C'est ce que nous présentons dans cette section.

3.3.1 Règle de Dempster

La règle de combinaison la plus usuelle est celle de Dempster. Elle est donnée par :

$$m_1 \oplus m_2(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C) \tag{3.13}$$

$$\text{avec } K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \tag{3.14}$$

$$\text{et } m(\emptyset) = 0 \tag{3.15}$$

Cette règle est une règle conjonctive normalisée : conjonctive parce qu'elle fait converger la croyance sur l'intersection des éléments focaux des fonctions de masse combinées et normalisée parce que la masse sur l'ensemble vide est conservée nulle.

Le facteur K de l'équation (3.13) correspond à la règle de combinaison de Smets [84] et est donnée par :

$$m_1 \textcircled{\cap} m_2(A) = \sum_{B \cap C = A} m_1(B) m_2(C) \tag{3.16}$$

Un exemple de combinaison de fonctions de masse est illustré tableau 3.1. Dans cet exemple, on peut voir que la normalisation imposée dans la règle de Dempster fait converger le résultat sur l'hypothèse la plus défendue par les deux fonctions de masse (*i.e.* $m_1 \oplus m_2(\{Oui\}) > m_1(\{Oui\}), m_2(\{Oui\})$).

L'un des principaux avantages de ces deux règles est le fait qu'elles sont totalement indépendantes de la signification réelle des preuves reccueillies. Ces règles permettent donc :

– *D'utiliser des capteurs hétérogènes pour le calcul d'un même attribut de contexte.* On peut

m	$\{Oui\}$	$\{Non\}$	$\{Oui \cup Non\}$	$\emptyset$
m_1	0.7	0.2	0.1	0
m_2	0.5	0.4	0.1	0
$m_1 \textcircled{\cap} m_2$	0.47	0.14	0.01	0.38
$m_1 \oplus m_2$	0.758	0.226	0.016	0

TABLEAU 3.1 – Exemple de combinaison de fonctions de masse à l'aide des règles de Smets et Dempster.

ainsi accumuler des croyances issues de capteurs complètement différents. Par exemple, la présence peut être obtenue à l'aide de capteurs de mouvement, de capteurs de pressions sur des chaises/canapés et divers capteurs binaires sur des équipements.

– *D'utiliser des configurations différentes de capteurs pour calculer les mêmes attributs de contexte.* Par exemple, la présence dans une cuisine ne sera pas forcément détectée à l'aide des mêmes capteurs que dans un salon ou une salle de bain. On a donc une vraie flexibilité du système pour le calcul des abstractions nécessaires.
– *D'avoir un résultat issu de la fusion malgré des pertes de données.* En effet, la souplesse du système sur les capteurs à déployer obtenue grâce à ces règles font que les pertes de données peuvent n'avoir qu'un impact mineur sur le résultat si suffisamment de preuves sont reccueillies quoiqu'il arrive. C'est particulièrement important si les capteurs communiquent via des communications sans fil instables et peu fiables.

L'une des contraintes de cette règle est que les preuves doivent être indépendantes. Ainsi, si l'on utilise plusieurs fois le même type de capteur, pour des raisons de fiabilité par exemple, les preuves recueillies ne doivent pas être toutes fusionnées avec des preuves issues d'autres types de capteurs. En effet, si nous utilisons par exemple trois capteurs de mouvement et un capteur de niveau sonore, tous dans la même pièce, on ne doit pas considérer quatre preuves mais uniquement deux. Les trois capteurs de mouvement ne sont pas indépendants en ce qui concerne la présence (leurs champs de vision peuvent se superposer) et ils doivent donc retourner tous la même preuve. Il ne faut donc pas donner plus de poids au mouvement qu'au niveau sonore lors de la combinaison. Ces trois capteurs de mouvement peuvent cependant être considérés comme indépendants si l'on cherche à calculer le mouvement réellement détecté et uniquement cela.

3.3.2 Conflit

Une des grandes difficultés rencontrées lors de l'application des fonctions de croyance est la gestion du conflit. La caractérisation même du conflit est la source de nombreux travaux [77, 56, 55, 76, 83]. La règle de Dempster n'est pas adaptée pour la combinaison de preuves en conflit. Ceci est parfaitement illustré par l'exemple donné par Zadeh [102] (*cf.* tableau 3.2). Dans cet exemple, en combinant deux fonctions de masse fortement en conflit, on se retrouve, après normalisation, avec une fonction de masse catégorique appuyant une hypothèse qui n'était que faiblement appuyée par les deux fonctions de masse d'origine. Ce résultat est un peu contre-intuitif. En effet, dans un tel cas, on aurait tendance à penser intuitivement que les hypothèses A

m	A	B	C	$\emptyset$
m_1	0.99	0	0.01	0
m_2	0	0.99	0.01	0
$m_1 \textcircled{\cap} m_2$	0	0	0.0001	0.9999
$m_1 \oplus m_2$	0	0	1	0

TABLEAU 3.2 – Exemple de Zadeh : combinaison de deux fonctions de masse fortement en conflit à l'aide de la règle de combinaison de Dempster.

et B sont équiprobables et à considérer l'hypothèse C comme très peu probable. Nous éviterons donc autant que faire se peut de construire des modèles de croyance pouvant conduire à de tels cas.

Un outil pouvant permettre de caractériser un conflit est la masse sur l'ensemble vide (*i.e.* $m(\emptyset)$). On remarque dans cet exemple qu'en utilisant la règle de combinaison de Smets, règle non-normalisée, la fonction de masse résultante est pratiquement catégorique sur l'ensemble vide. Cette masse peut donc être un premier indicateur de conflit. Malheureusement, il n'est pas toujours un bon indicateur. En effet, la règle de combinaison de Smets n'est pas idempotente (*i.e.* $m \oplus m \neq m$) et a tendance à faire converger le résultat vers une fonction de masse catégorique sur l'ensemble vide (*i.e.* $m(\emptyset) = 1$). C'est ce que l'on appelle l'*auto-conflit* [66].

Il existe de nombreuses règles de combinaison [16, 29, 33, 51, 57, 61, 23, 97, 99]. Toutes ces règles permettent de gérer le conflit lors de la combinaison de fonctions masse. Malheureusement, il est difficile au sein d'une même règle de combinaison d'avoir à la fois une convergence du résultat, par exemple avec des règles conjonctives, et une bonne caractérisation du conflit.

La gestion du conflit dans la théorie des fonctions de croyance est un domaine de recherche en lui-même. Suite à quelques expérimentations rapides de différentes règles de combinaison, nous avons décidé pour nos travaux de n'utiliser que la règle de combinaison de Dempster pour la fusion de preuves issues de multiples capteurs. Des mécanismes de gestion du conflit feront l'objet de travaux futurs pour leur capacité à détecter les erreurs et les fautes au sein d'un groupe de capteurs aussi bien homogènes qu'hétérogènes [76].

3.4 Prendre une décision

Nous savons construire des preuves à partir de données de capteurs et les fusionner en une unique croyance représentative de l'ensemble. Nous devons maintenant nous poser la question de la décision. Les attributs de contexte que nous calculons sont destinés aux couches supérieures de l'architecture et/ou à des services. L'interprétation des fonctions de masse n'est pas évidente. Il serait donc préférable d'être capable d'en dégager un couple (état - confiance) correspondant à une décision sur ce que l'on considère comme étant l'état réel du monde vis-à-vis des informations que l'on a reccueillies. Dans cette section, nous présentons les transformations classiques des fonctions de masse qui sont utilisées pour la prise de décision ainsi que la manière dont ces critères peuvent être utilisés.

3.4.1 Critères classiques

Dans la littérature, on trouve deux transformations classiques des fonctions de masse. La première transformation classique est la *crédibilité*. Elle est définie comme suit :

$$bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \tag{3.17}$$

Cette transformation correspond à un critère pessimiste pour la prise de décision. En effet, elle ne prend en compte que la masse directement associée à un élément focal et à ses sous-ensembles. Cela signifie que si l'on souhaite prendre une décision uniquement sur les états possibles du monde, et non pas sur éventuellement un ensemble d'états, alors cela revient à prendre la décision directement sur la fonction de masse sur les éléments focaux atomiques car on a $bel(A) = m(A)$ si $|A| = 1$. La crédibilité correspond donc en quelque sorte au degré de certitude du système. Ce critère pose particulièrement problème lorsque peu de preuves sont disponibles. En effet, il se peut que la fonction de masse résultant de la fusion ne contienne aucun élément focal atomique.

La deuxième transformation classique, considérée elle comme étant optimiste, est la *plausibilité*, elle est définie comme suit :

$$pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{3.18}$$

Cette transformation est considérée comme optimiste car elle orend en compte toute la masse qui peut être potentiellement accordée à un état du monde. La plausibilité d'un état correspond donc au degré de possibilité de cet état. C'est-à-dire que $pl(A) = 1$ signifie uniquement que A est possible. Ce critère est donc trop optimiste car il est possible d'avoir une plausibilité maximale pour chaque état possible du monde simplement avec la fonction de masse vide. En effet, si l'on a $m(\Omega) = 1$, alors on a $pl(A) = 1$ pour tout $A \subseteq \Omega$. Cela ne nous aide donc pas vraiment à prendre une décision...

Une troisième transformation a alors été créée par Smets [82] : la *transformation pignistique*. Elle est définie comme suit :

$$BetP(A) = \sum_{\emptyset \neq B \subseteq \Omega} \frac{|A \cap B|}{|B|} m(B) \tag{3.19}$$

Cette transformation répartit de façon équilibrée la masse accordée à des ensembles de mondes possibles entre ces mondes. Elle peut être considérée comme un pari sur la probabilité des différents états possibles. D'ailleurs, la somme des $BetP$ de tous les ensembles atomiques vaut 1 comme lorsqu'on travaille avec des probabilités et l'on retrouve aussi l'additivité pour les unions d'états (*i.e.* $A, B \in \Omega, BetP(A \cup B) = BetP(A) + BetP(B)$). Ce critère de décision est considéré comme étant le plus neutre des trois. Ces trois transformations respectent d'ailleurs l'inégalité suivante :

$$bel(A) \leq BetP(A) \leq pl(A) \tag{3.20}$$

La répartition de la masse faite par ces trois transformations est explicitée visuellement sur la

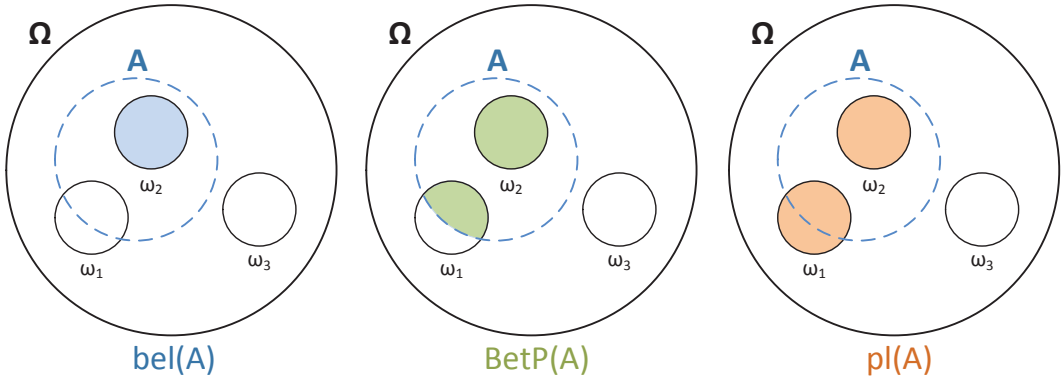

FIGURE 3.4 – Répartition de la masse pour la crédibilité, la plausibilité et la transformation pignistique.

figure 3.4.

3.4.2 Sélection d'un état

Les trois transformations présentées précedemment peuvent être utilisées comme critères de décision pour sélectionner l'état que le système considère comme étant l'état réel du monde. Dans le meilleur des cas, on souhaiterait obtenir une réponse à la fois précise et certaine. Bien entendu, cela n'est pas toujours possible.

Dans la définition du cadre de discernement, nous avons vu que les mondes doivent être exclusifs. On sait donc qu'un seul état représente l'état réel du monde. Intuitivement, on utilise donc un des trois critères et on cherche l'état ayant la plus grande valeur pour la fonction choisie. Ainsi, le couple (état - confiance) que l'on souhaite retourner sera de la forme :

$$(argmax(f(A)), max(f(A)),\ A \in \Omega,\ f = bel,\ BetP\ ou\ pl \tag{3.21}$$

De cette manière, on prend donc une décision précise sur l'état que le système considère comme réel. Malheureusement, les trois transformations présentées précédemment ne permettent pas toujours de prendre une bonne décision. La crédibilité peut être trop pessimiste et revient à prendre une décision directement sur la fonction de masse si on ne considère que les éléments atomiques. Il suffit donc qu'aucun état n'ait de la masse qui lui soit directement accordé (*i.e.* $\forall A \in \Omega, m(A) = 0$) pour qu'on se retrouve à choisir n'importe quel état avec une confiance de 0. De même, pour la plausibilité, on obtient un résultat identique mais ce coup-ci avec une confiance de 1.

Pour comprendre cela, il suffit de considérer la fonction de masse vide (*i.e.* $m(\Omega) = 1$) et on se rend compte qu'effectivement on a bien :

$$\forall A \in \Omega,\ bel(A) = 0,\ pl(A) = 1 \tag{3.22}$$

La transformation pignistique a au moins l'avantage d'accorder un minimum de confiance à

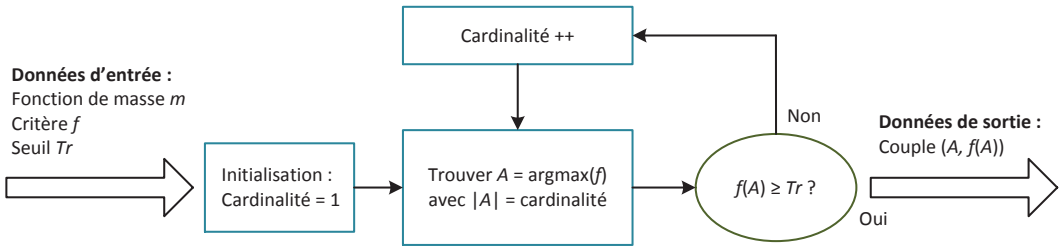

FIGURE 3.5 – Algorithme de prise de décision : compromis entre précision et certitude.

certains mondes possibles. Cela ne permet pas pour autant de toujours pouvoir prendre une réelle décision. Si peu de preuves ont été récoltées, et si le nombre de mondes possibles est important, on peut alors se retrouver avec un cas où il est effectivement possible de trouver un maximum mais que celui-ci ne soit pas satisfaisant car la certitude (ou confiance) est trop faible.

En prenant une décision uniquement sur les états sans considération des unions possibles d'états, on a donc un maximum de précision mais une décision qui peut rapidement perdre tout son sens et donc son intérêt. On va donc chercher à faire un compromis entre la précision (*i.e.* la cardinalité de l'état retourné) et la confiance. On peut alors remarquer que la crédibilité, la transformation pignistique et la plausibilité ont toutes les trois une propriété intéressante :

$$\forall A, B \subseteq \Omega,\ si\ A \subseteq B\ alors\ f(A) \leq f(B)\ avec\ f = bel,\ BetP\ ou\ pl \tag{3.23}$$

En d'autres termes, plus on augmente la cardinalité (*i.e.* on réduit la précision), plus on augmente la confiance que l'on obtient. On tient donc notre compromis. Pour le mettre en place, il suffit alors de se donner un seuil de confiance à atteindre et d'appliquer l'algorithme donné figure 3.5.

Cet algorithme converge toujours. En effet, quel que soit le seuil fixé entre 0 et 1, cet algorithme convergera, dans le pire cas, vers le fait que l'état réel du monde se trouve dans Ω (trivial étant donnée la définition du cadre de discernement). En effet, si m est normalisée (*i.e.* $m(\emptyset) = 0$) on a toujours :

$$bel(\Omega) = BetP(\Omega) = pl(\Omega) = 1 \tag{3.24}$$

Il est alors possible pour le système d'exprimer une indécision totale vis-à-vis de ce qu'il considère être l'état réel du monde.

Cet algorithme peut s'avérer particulièrement utile dans certaines situations. Par exemple, si nous cherchons à savoir si une personne est endormie, il peut être intéressant de savoir qu'elle est soit assise soit allongée avec une confiance élevée plutôt que strictement assise ou strictement allongée avec une confiance plus faible. Certaines unions d'états permettent donc d'augmenter la confiance au détriment de la précision sans que cette perte de précision n'ait nécessairement un impact négatif pour les modèles ou les services exploitant un attribut de contexte.

Maintenant que nous avons un compromis entre la précision et la certitude, on peut se poser la question du critère à choisir. Chacun des critères a sa propre signification. La crédibilité représente la certitude, la plausibilité le fait qu'un état est possible et la transformation pignistique

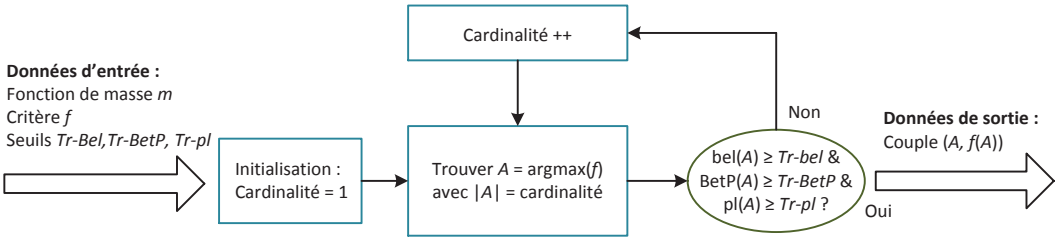

FIGURE 3.6 – Algorithme de prise de décision : compromis entre précision et certitude reposant à la fois sur la crédibilité, la plausibilité et le pari sur la probabilité.

représente une confiance neutre.

On a déjà vu que la fonction de masse vide pose un problème si on utilise la crédibilité ou la plausibilité. Avec l'algorithme proposé figure 3.5, le problème ne persiste que pour la plausibilité. En effet, si c'est la crédibilité qui est choisie, le seul ensemble d'états qui dépassera le seuil fixé sera l'ensemble complet des mondes possibles.

De même, on peut avoir un pari sur la probabilité $BetP$ élevé sans pour autant avoir une plausibilité élevée elle aussi, par exemple, avec la fonction de masse suivante :

$$m(A) = 0.6$$
$$m(B) = 0.4$$

Avec cette fonction de masse, on peut considérer que $BetP(A) = 0.6$ est suffisant pour prendre une décision. Seulement, la plausibilité $pl(A) = 0.6$ est finalement assez faible. D'ailleurs, intuitivement, on pourrait dire que cette fonction de masse est plutôt mitigée entre les états A et B.

C'est donc un mélange des trois critères qui nous intéresse. On souhaite un minimum de certitude, une confiance ni trop pessimiste ni trop optimiste et enfin que l'état sélectionné soit réellement plausible. Il est possible de reprendre l'algorithme de la figure 3.5 et de remplacer la vérification du seuil par une vérification de trois seuils. Un seul des trois critères est choisi comme critère principal servant de confiance associée à l'état. On obtient alors l'algorithme présenté figure 3.6. Pour la confiance retournée dans le couple (état, confiance), il est possible de choisir n'importe lequel des trois critères. Dans notre cas, puisque nous souhaitons une confiance ni trop pessimiste ni trop optimiste, nous choisirons donc le pari sur la probabilité $BetP$.

Les seuils pour chacun des trois critères ont été sélectionnés de façon empirique suite à nos expérimentations. Ainsi, nous avons obtenu des valeurs de seuil de 0.3, 0.6 et 0.9 respectivement pour la crédibilité, le pari sur la probabilité et la plausibilité. Ces valeurs sont cohérentes vis-à-vis de la signification de chacun des critères et permettent d'éviter les cas limites de chacun de ces critères. Ainsi, on souhaite un minimum de certitude pour sélectionner un état unique. Cela est rendu possible notamment par l'utilisation de la règle de combinaison de Dempster qui est conjonctive. Elle a donc tendance à converger vers des éléments focaux avec une cardinalité de plus en plus petite. Une état sera donc sélectionné seul uniquement si suffisamment de preuves

l'appuient.

3.5 Exemple d'application : détection de la présence

Nous savons a présent modéliser des croyances, les construire à partir de données de capteurs et enfin fusionner les preuves puis prendre une décision. Dans cette section, nous illustrons l'utilisation des fonctions de croyance avec la détection de la présence à partir de capteurs placés dans une cuisine.

3.5.1 Modèles

Pour détecter la présence, nous avons placé dans une pièce divers capteurs renvoyant chacun leur mesure toutes les secondes :

- Un capteur de mouvement ;
- Trois capteurs de vibration sur des chaises ;
- Un capteur comptant le nombre de personnes assises (obtenu à partir de trois capteurs de pression sur les chaises) ;
- Un capteur de contact sur la fenêtre ;

Nous limitons le nombre de données envoyées par les capteurs à une par seconde pour des raisons de consommation d'énergie. En effet, certains de nos capteurs sont regroupés en nœuds de capteurs fonctionnant sur batterie et communiquant via radio. Nous ne souhaitons donc pas envoyer des données en continu sous peine de vider les batteries en quelques jours voire quelques heures. De même, le calcul en continu des valeurs des attributs de contexte, en plus d'être inutile, serait coûteux et nous ne souhaitons pas installer une grille de calcul dans la maison instrumentée.

Afin de pouvoir exploiter les capteurs disposés dans la cuisine, nous avons donc défini des modèles de croyance donnés figure 3.7. Dans ces modèles, on peut déjà remarquer que peu de capteurs peuvent appuyer l'hypothèse de l'absence de personne. Ici, les chaises et le mouvement le permettent car nous estimons que dans une cuisine, soit nous nous déplaçons, soit nous nous asseyons. Etant une pièce d'activité, il est peu probable qu'une personne reste debout à ne pas bouger du tout en plein milieu de la cuisine !

Autre fait intéressant sur les modèles, seul le capteur de contact sur la fenêtre permet d'avoir une croyance catégorique au moment où la fenêtre change d'état (ouverture ou fermeture). Tous les autres modèles induisent toujours une part d'ignorance totale. Cela est dû au fait que nous ne sommes effectivement jamais complètement certains de ce que l'on observe. Par exemple, les capteurs de mouvement peuvent être particulièrement bruités. De même, les capteurs de pression sur les chaises peuvent avoir du mal à différencier une personne d'un sac de courses très lourd. On n'accorde donc que rarement une confiance totale au fait que ce que mesure un capteur reflète une réelle présence. Cette ignorance totale, même faible, permet aussi d'éviter de façon plus générale les problèmes de conflit qu'on a pu voir avec l'exemple de Zadeh. On essayera donc toujours de construire des fonctions de masse non-dogmatiques.

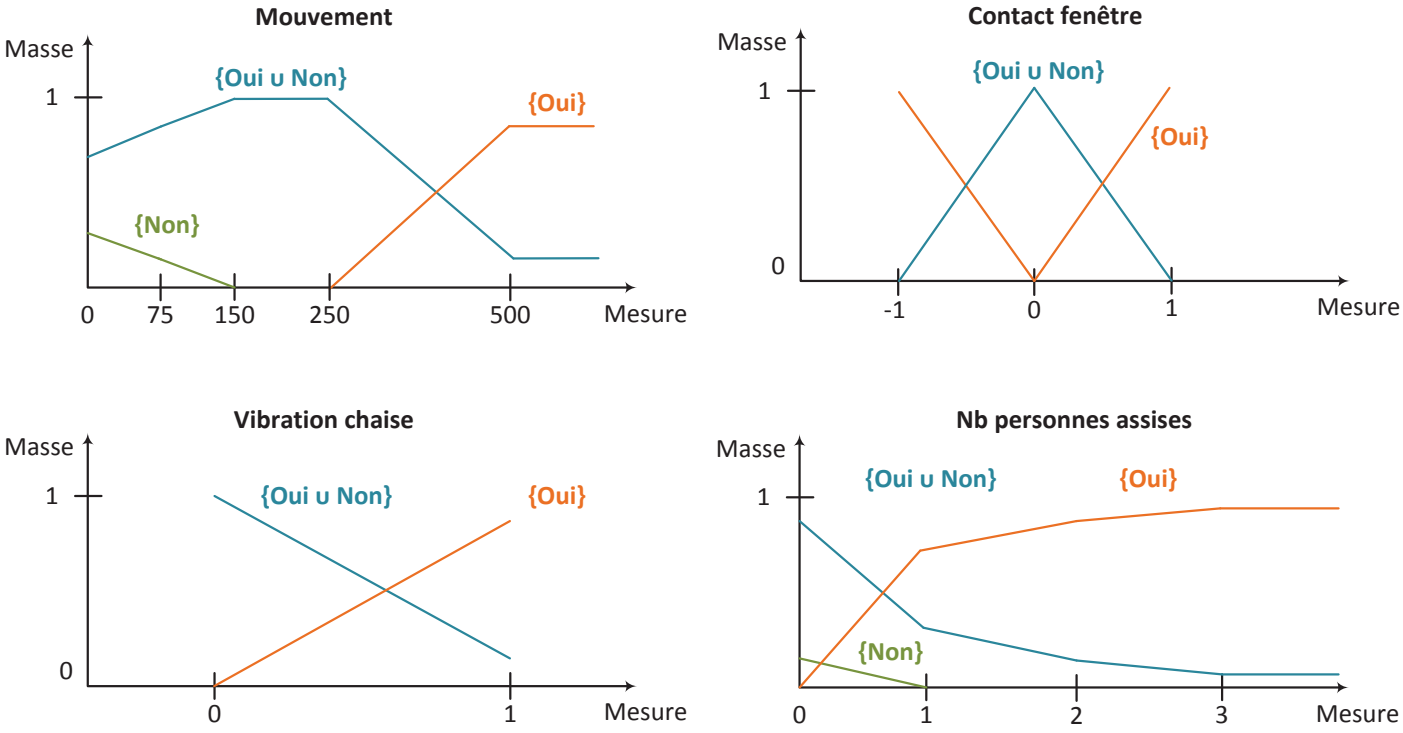

FIGURE 3.7 – Ensembles de fonctions de masse définis pour la détection de la présence dans une pièce.

3.5.2 Résultats

Nous avons mené une courte expérience d'environ cinq minutes pour vérifier que nous étions capables de détecter la présence dans la pièce à l'aide de capteurs. L'expérience s'est déroulée comme suit :

- T = 0s : personne n'est dans la pièce
- T = 30s : une personne rentre puis se déplace dans la pièce
- T = 62s : la personne s'assied sur une chaise
- T = 120s : la personne se lève
- T = 130s : la personne ouvre la fenêtre
- T = 135s : la personne se rassied
- T = 190s : la personne se lève et ferme la fenêtre
- T = 195s : la personne sort de la pièce

Les fonctions de masse résultant des mesures des capteurs sont données figure 3.8. Chaque perte de données résulte en une fonction de masse vide correspondant effectivement à l'absence de preuve. On peut observer plusieurs résultats intéressants.

Le premier est l'**asynchronicité** des preuves récoltées. Il est évident que les capteurs que l'on a utilisés ne verraient pas tous en même temps des preuves de la présence. Une activité normale dans une cuisine ne conduit effectivement pas à l'actionnement de tout à la fois tout le temps ! L'accumulation de preuves est donc finalement assez difficile du fait qu'elles n'interviennent pas nécessairement au même instant.

Deuxième fait intéressant, l'**instabilité** des mesures et donc des croyances. On peut voir notamment que la croyance induite par le capteur de mouvement lorsque la personne se déplace dans la pièce n'est absolument pas stable (cela peut s'expliquer en partie par la mesure non continue qui ne permet pas d'acquérir toutes les variations des mesures issues de ce capteur). Bien que le $\{Oui\}$ domine largement, le système ne donne pas une réponse stable. De même, des pertes de données dans les communications sans fil entraînent facilement une instabilité des croyances comme on peut le voir pour le nombre de personnes assises.

Enfin, avec les capteurs qu'on a mis en place, le système n'a que peu de chances de détecter l'*absence* dans la pièce. Les croyances induites sur le $\{Non\}$ sont en effet trop faibles et trop peu nombreuses pour que le système puisse converger.

Le résultat de la fusion de l'ensemble des croyances et les transformations pour la prise de décision sont donnés figure 3.9. On voit sur ces courbes que la présence semble être assez bien détectée avec quelques instabilités notamment quand la personne se déplace dans la pièce. Si l'on applique le filtre décrit précédemment pour la prise de décision avec des seuils à 0.3, 0.6 et 0.9 respectivement pour la crédibilité, le pari sur la probabilité et la plausibilité, alors l'absence reste toujours indétectée. Nous pourrions "corriger" cela en augmentant l'engagement des modèles capables d'induire de la croyance sur l'absence. Malheureusement, cela pourrait rendre instable notre système et finalement perturber la détection de la présence.

Avec cet exemple simple de la présence, on observe donc que l'instabilité des capteurs doit être corrigée pour éviter une fluctuation trop importante de la réponse du système mais aussi que l'absence reste difficile à détecter.

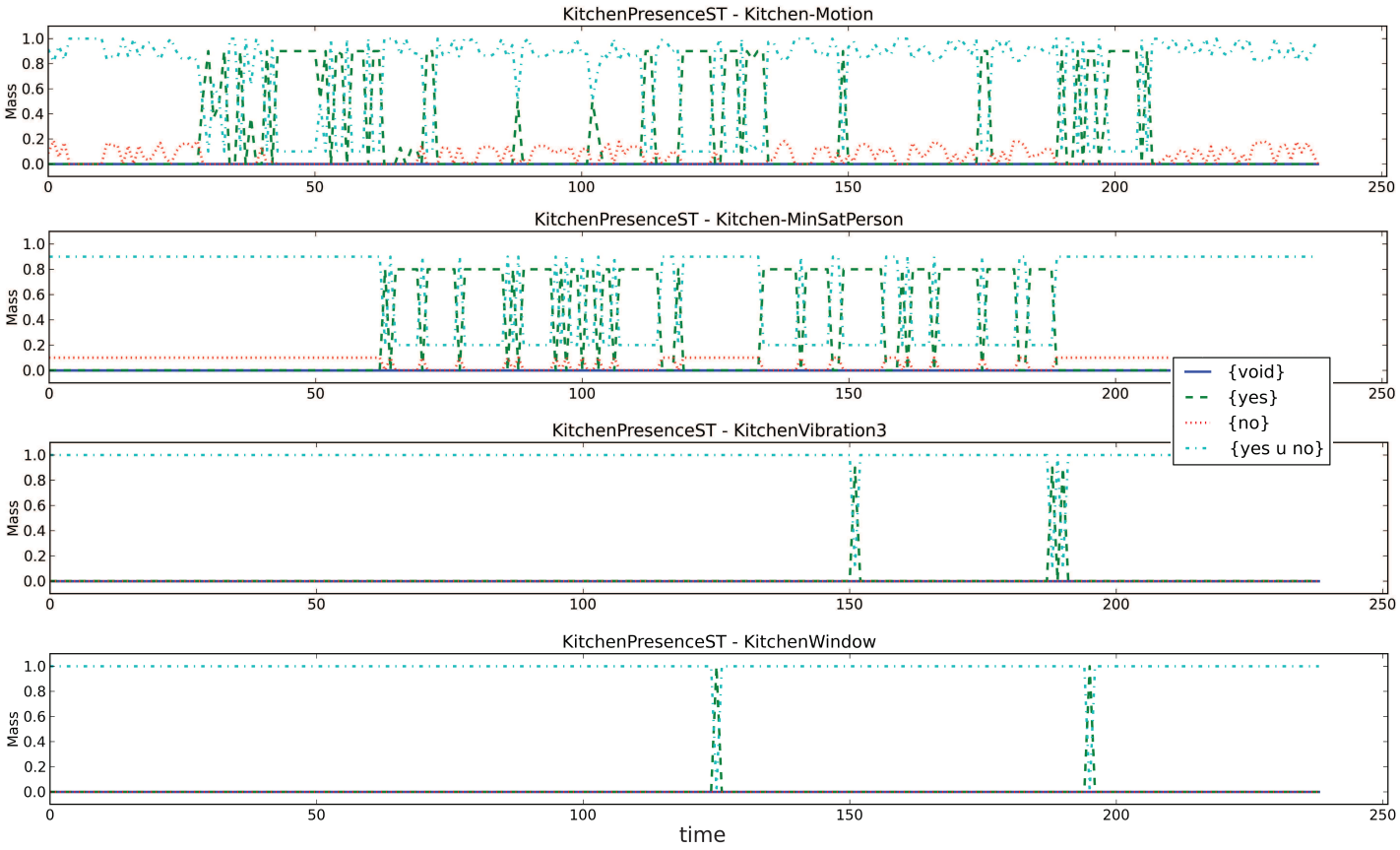

FIGURE 3.8 – Croyances induites par les capteurs placés dans une cuisine pour la détection de la présence. Le temps est donné en secondes. Un seul capteur de vibration apparaît car une seule chaise a été mise à contribution pour l'expérience.

3.6 Conclusion

Nous avons présenté les bases de la théorie des fonctions de croyance et comment il est possible de construire et de fusionner des croyances à partir de données fournies par un ensemble de capteurs élémentaires. Enfin, nous avons mis en avant un mécanisme de prise de décision à partir des fonctions de masse.

Nous avons vu que les fonctions de croyances permettent de prendre en compte à la fois l'imprécision en spécifiant des ensembles d'états possibles et l'incertitude en associant des degrés de croyance à ces ensembles d'états. La fusion de fonctions de masse étant générique, il est possible de calculer des attributs de contexte stables avec des configurations de capteurs différentes d'un déploiement à l'autre. Des mécanismes de gestion de conflit peuvent aussi nous permettre de détecter et de gérer les fautes et les erreurs au niveau des capteurs. Enfin, des mécanismes de prise de décision permettent de gagner en confiance au détriment de la précision permettant au système d'offrir une réponse reflétant la qualité des preuves qu'il a reçues.

L'exemple présenté sur la détection de présence à l'aide des fonctions de croyance met en

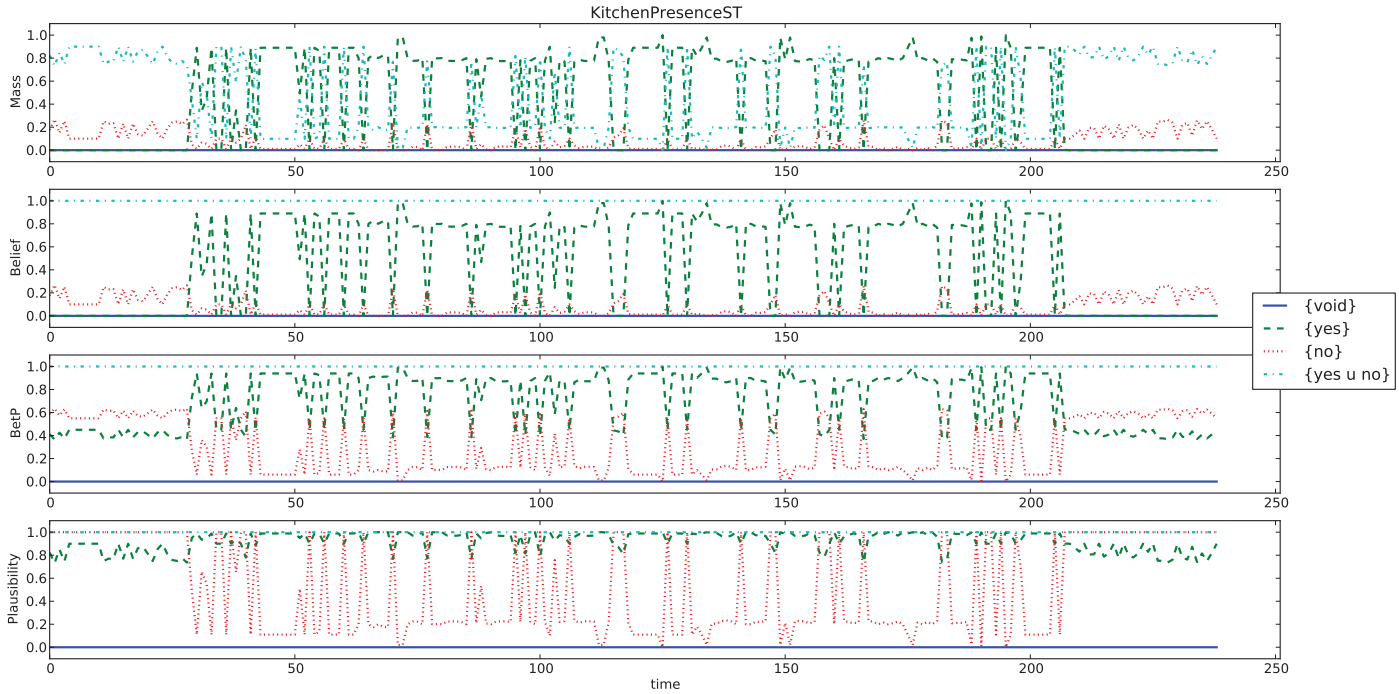

FIGURE 3.9 – Résultat de la fusion des preuves issues des capteurs placés dans une cuisine pour la détection de la présence.

avant deux principales questions. 1°) Comment stabiliser les croyances induites par des capteurs instables et sujets à des pertes de données ? 2°) Comment réussir à faire converger une croyance avec très peu de preuves sans trop perturber le système ?

Chapitre 4

Temporisation des croyances : vers des attributs de contexte stables

Il est payant parfois de savoir prendre son temps. Les tronches défaites du bâfreur hâtif et de l'éjaculateur précoce sont éloquentes à cet égard.
Pierre Desproges

Un seul homme, en temporisant, a rétabli notre situation.
Cicéron

Dans le chapitre précédent, nous avons vu comment appliquer la théorie des fonctions de croyance pour calculer des attributs de contexte (*e.g. présence*) à partir de données de capteurs. De cette première étape de nos travaux, nous avons dégagé plusieurs problèmes. Le premier est l'asynchonicité des preuves reccueillies (*cf* figure 4.1). En effet, même en déployant des capteurs hétérogènes, il n'est pas évident que plusieurs d'entre eux observent quelque chose de pertinent au même moment. Nous souhaiterions donc réussir à fusionner des preuves asynchrones. Cette asynchronicité n'est pas toujours problématique lorsque le temps lui-même est une preuve. Ainsi, il est possible de considérer toutes les preuves reccueillies dans une fenêtre temporelle [58]. Cette dernière se prête bien à la détection de situations ou d'activités inscrites dans le temps et dont les durées sont relativement fixes, ou au moins bornées, d'une fois sur l'autre. Malheureusement, cela ne s'applique pas aux attributs de contexte que l'on cherche à calculer. Par exemple, on ne peut pas considérer un temps de présence moyen. En plus du fait que cela n'aurait pas de sens, le système perdrait grandement en réactivité en inférant toujours *a posteriori*.

Autre problème majeur qu'illustre la figure 4.1, l'instabilité des croyances induites. Elle peut

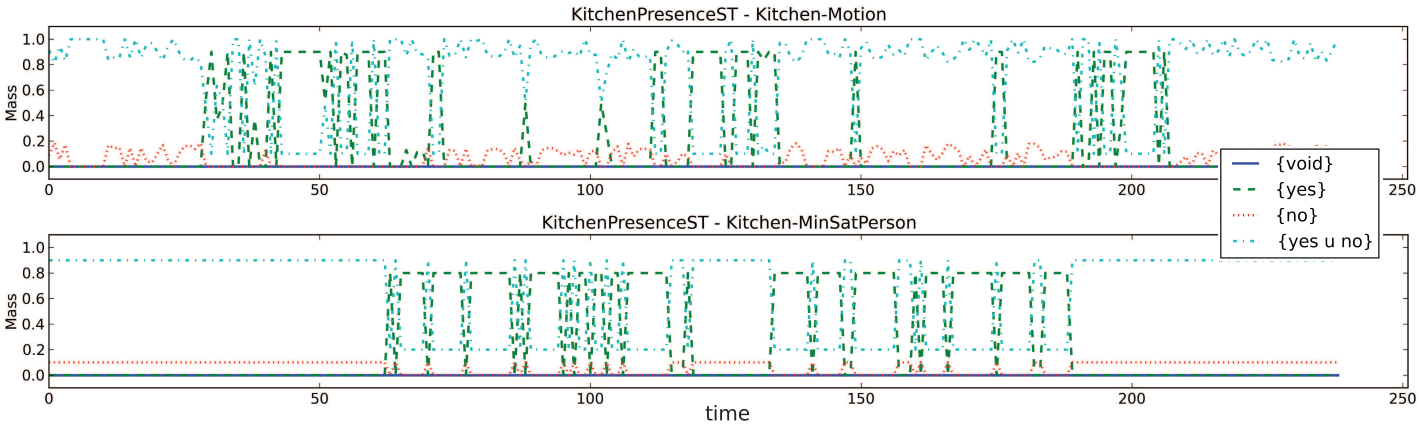

FIGURE 4.1 – Croyances induites par un capteur de mouvement (en haut) et par des capteurs de pression sur des chaises (en bas) pour la détection de la présence dans une pièce. La personne se déplace dans les intervalles [30-62], [120-140] et [190-215]. Elle est assise dans les intervalles [62-120] et [135-190].

être due à deux causes majeures. Premièrement, le mouvement détecté n'est pas toujours très élevé. Ainsi, le degré de croyance associé à la présence peut varier d'une mesure sur l'autre. Deuxièmement, il est possible d'avoir des pertes de données dues au réseau sous-jacent ou aux capteurs. Nous avons fait en sorte que notre système retourne une fonction de masse vide lorsqu'il n'a pas reçu une donnée. Cette fonction de masse correspond en effet à un "je ne sais pas" du système.

Cette fonction de masse vide renvoyée par le système en cas de perte de données pose un problème. D'une part, les pertes peuvent être très fréquentes comme on peut le voir avec le capteur de pression sur la chaise. D'autre part, sur un cas comme la détection de présence, si du mouvement a été observé, on peut considérer sans trop se tromper que la personne qui a bougé est encore présente dans la pièce la seconde qui suit ou au moins défendre partiellement cette hypothèse. Ainsi, il serait préférable lorsqu'on a perdu une donnée de considérer la croyance précédente plutôt que le système dise "qu'il ne sait rien".

Dans ce chapitre, nous proposons l'introduction de mécanismes de temporisation qui permettent de propager les croyances dans le temps. Nous présentons d'abord comment propager dans le temps les croyances. Nous proposons ensuite deux algorithmes permettant de considérer à la fois l'ancienne croyance propagée et la nouvelle croyance créée à partir d'une donnée de capteur. Le comportement de ces deux algorithmes ainsi que des résultats expérimentaux sont présentés dans ce chapitre. Enfin, une mise à l'épreuve des algorithmes est présentée afin de confirmer l'intérêt de ceux-ci.

4.1 Temporisation avec discrimination

On souhaite pouvoir propager des croyances dans le temps afin d'améliorer la stabilité de la réponse de notre système. Pour cela, nous introduisons ici un premier algorithme permettant de

réduire la croyance avec le temps puis d'effectuer une discrimination entre l'ancienne croyance et la nouvelle croyance lorsque l'on en a une.

4.1.1 Propagation dans le temps

Dans le chapitre précédent, nous avons vu les résultats d'une expérience où les capteurs renvoient une mesure chaque seconde. Cette périodicité est imposée par les contraintes de notre architecture : des capteurs portés par des nœuds autonomes et reliés via des liens sans fil offrant des capacités limitées. Chaque fois que l'on reçoit une mesure, on estime qu'elle a été reçue immédiatement et la fonction de masse associée est construite selon cette hypothèse.

Maintenant, on aimerait être capable de considérer une croyance plus ancienne. On pourrait estimer qu'une croyance reste valide à tout instant, même longtemps après sa construction. On aurait ainsi la propagation dans le temps la plus simple qui soit, à savoir :

$$m_{t+1} = m_t \tag{4.1}$$

Cette approche n'est pas acceptable. En effet, on ne peut pas considérer une croyance comme étant toujours valide. Si par exemple un mouvement a été observé à un temps donné, ce mouvement peut permettre de dire qu'il y a certainement quelqu'un à l'instant donné et peut-être pour les quelques secondes à venir mais pas au-delà. Enfin, plus la croyance est ancienne, plus elle doit être faible et peu engagée. C'est pourquoi, pour propager dans le temps les fonctions de masse, nous avons décidé d'utiliser l'affaiblissement donné par :

$$m_{\alpha(t)}(A) = \begin{cases} (1-\alpha(t)).m(A) \text{ si } A \subset \Omega \\ (1-\alpha(t)).m(A) + \alpha(t) \text{ si } A = \Omega \end{cases} \text{ avec } \alpha(t) \in [0,1] \tag{4.2}$$

Ici, le coefficient d'affaiblissement α dépend du temps. Il est possible de définir n'importe quelle fonction pour ce coefficient, nous avons décidé d'utiliser une fonction affine simple dont le seul paramètre est un temps au bout duquel l'affaiblissement est maximum. Dans la suite, on utilisera donc la fonction suivante pour calculer le coefficient d'affaiblissement α :

$$\alpha(t) = \frac{t - t_{ref}}{T_{max}} \tag{4.3}$$

T_{max} correspond au temps au bout duquel $\alpha(t) = 1$. t_{ref} correspond à l'instant auquel la fonction de masse à affaiblir a été construite. Etant donné que l'on ne remonte pas le temps, on aura toujours $t > t_{ref}$.

Avec cet affaiblissement temporel, on sait maintenant comment propager dans le temps des fonctions de masse. Elles sont affaiblies en fonction de leur "âge" jusqu'à devenir des fonctions de masse vide lorsque $\alpha(t) = 1$. On a donc une propagation correspondant à nos attentes. La croyance est oubliée petit à petit jusqu'à ne plus exister.

4.1.2 Spécificité et discrimination

On sait maintenant comment propager une fonction de croyance dans le temps. Seulement, dans notre cas, nous construisons une nouvelle croyance chaque seconde. On a donc à chaque seconde, deux croyances, celle correspondant à la mesure précédente qui a été affaiblie et celle qui vient d'être construite. Comment décider laquelle des deux croyances doit être sélectionnée ? Il existe différents outils pour caractériser et différencier les fonctions de masse. Ici, nous avons décidé d'utiliser la spécificité [98] définie par :

$$S_m = \sum_{A \subseteq \Omega, A \neq \emptyset} \frac{m(A)}{|A|} \text{ où } |A| \text{ est la cardinalité de } A \tag{4.4}$$

La spécificité peut être vue comme le degré de précision d'une fonction de masse ou l'inverse de la moyenne de la cardinalité des éléments focaux. Ainsi, elle est maximale (*i.e.* $S_m = 1$) uniquement pour une fonction de masse catégorique avec un élément focal atomique (*i.e.* $m(A) = 1, A \in \Omega$). Si $S_m = 0.5$, alors la cardinalité moyenne des éléments focaux est de 2. La fonction de masse vide étant la moins spécifique, on a donc :

$$S_m \geq S_{m_{vide}} \ , \forall m \tag{4.5}$$

Le fait que l'affaiblissement transfère une partie de la masse de tous les éléments focaux sur l'ignorance totale (*i.e.* $m(\Omega)$) crée donc une relation facile à exploiter :

$$S_{m_\alpha} \leq S_m, \forall m, \forall \alpha \in [0, 1] \tag{4.6}$$

La discrimination entre une ancienne croyance et une croyance nouvellement construite se fait en suivant l'algorithme 1. Avec ce dernier, on considère comme plus valide la croyance la plus spécifique. La croyance affaiblie se verra donc toujours sélectionnée dans les cas où il y a une perte de donnée (*i.e.* la nouvelle croyance est une fonction de masse vide) et majoritairement dans les cas où un bruit (*i.e.* une mesure incohérente ou fausse) a engendré une croyance faible qui ne doit pas être prise en compte.

4.1.3 Analyse du comportement de l'algorithme

Nous disposons maintenant d'un algorithme nous permettant de temporiser nos fonctions de masse et ainsi les propager dans le temps. Regardons maintenant un peu plus en détail l'effet de cet algorithme sur les fonctions de masse. Dans les exemples données ici, une temporisation avec $T_{max} = 8s$ est appliquée. Une croyance est donc totalement oubliée après huit secondes. Afin de pouvoir observer facilement le comportement réel des algorithmes, nous les testons sur des fonctions de masses parfaites. Dans cette section, les fonctions de masse présentées ne sont donc pas obtenues à partir de données de capteurs. Elles sont construites manuellement.

Algorithme 1 Temporisation et discrimination

m_{new} : la dernière fonction de masse obtenue
m_{old} : la dernière fonction de masse sauvegardée
t : le temps présent
t_{ref} : le temps de référence associé à m_{old}
T_{max} : le temps maximum de persistance de la croyance

$m_\alpha \leftarrow$ affaiblissement(m_{old}, t, t_{ref}, T_{max})
if $S_{m_\alpha} > S_{m_{new}}$ **then**
    return m_α
else
    $m_{old} \leftarrow m_{new}$
    $t_{ref} \leftarrow t$
    return m_{new}
end if

Changement d'état

La temporisation ne doit pas empêcher l'état dominant la croyance de changer. En effet, si l'on souhaite conserver une certaine réactivité du système, un état donné ne doit pas persister trop longtemps lorsque cela n'est pas nécessaire, notamment lorsqu'une preuve vient appuyer une hypothèse inverse à la preuve précédente.

La figure 4.2 illustre le comportement de la temporisation avec discrimination lors du changement d'état. On constate ainsi que celui-ci se fait sans problème lorsque les deux croyances ont le même niveau d'engagement (ici, tous les deux à 0.8, à t = 11s). La transition se fait par contre plus lentement si le nouvel état dispose d'une croyance nettement plus faible (t = 21s). Avec ce comportement, une preuve faible mettra un certain temps à s'imposer face à une preuve forte. Par exemple, la détection d'un mouvement est une bonne preuve de présence tandis que la détection d'aucun mouvement n'est qu'une faible preuve de l'absence. Ainsi, dans notre cas de détection de présence, si la personne s'arrête de bouger quelques secondes, la croyance associée au fait qu'elle est présente persistera jusqu'à devenir suffisament faible pour être discriminée.

Fluctuation et bruit

Intéressons-nous maintenant au cœur du problème, les fluctuations et le bruit. En effet, on a pu constater que les fluctuations et le bruit sont un problème majeur lorsque l'on travaille avec de vrais capteurs. On cherche donc à stabiliser la croyance de notre système. La figure 4.3 illustre le comportement de la temporisation lorsque l'engagement de la fonction de masse fluctue légèrement (t = 11-12s), lorsqu'un bruit induit une croyance différente mais moins engagée (t = 21-22s) et enfin lorsque des pics importants viennent perturber notre système (t = 31s et t = 41s). Cela peut être notamment le cas avec les capteurs de mouvements qui sont parfois sujets à des pics fugaces de bruit.

Ainsi, on constate une meilleure stabilité de la croyance dans le cas d'une fluctuation et

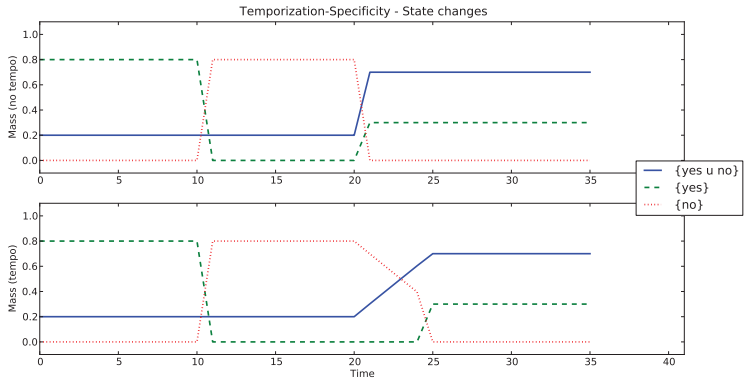

FIGURE 4.2 – Changement d'état dominant une fonction de masse au cours du temps sans temporisation (en haut) et avec temporisation (en bas).

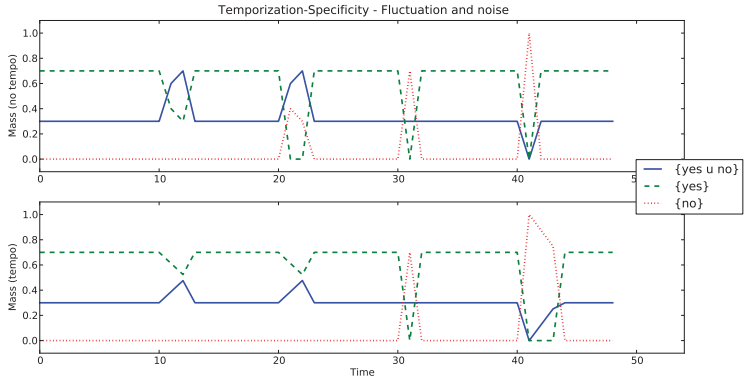

FIGURE 4.3 – Effet des fluctuations et du bruit sur une fonction de masse au cours du temps sans temporisation (en haut) et avec temporisation (en bas).

dans le cas d'un bruit léger. Le pic de croyance à niveau équivalent induit un changement d'état temporaire tandis qu'un pic maximum perturbe le système pour un temps qui peut être long suivant le T_{max} choisit comme paramètre de la temporisation. On a donc une élimination partielle du bruit et une stabilisation de la croyance mais aussi une sensibilité plus importante aux pics.

Cette sensibilité aux pics est ici un argument supplémentaire pour le respect du principe de moindre engagement dans la construction des modèles. En respectant ce principe, on évitera certainement la formation de tels pics. La convergence suite à la fusion pourrait être impactée par un engagement moindre des modèles mais la persistance des croyances permet la fusion de preuves asynchrones et compense donc cette potentielle perte de convergence.

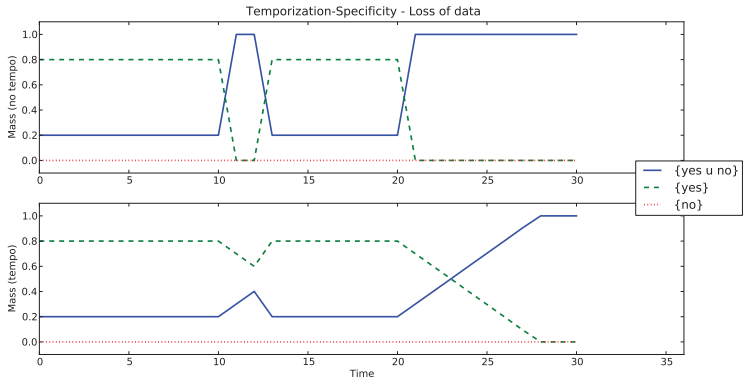

FIGURE 4.4 – Effet de la perte de données sur une fonction de masse au cours du temps sans temporisation (en haut) et avec temporisation (en bas).

Perte de données

Les pertes de données peuvent être importantes notamment à cause des communications sans fil entre les nœuds de capteurs. Nous avons fait en sorte que notre système réponde par une fonction de masse vide par défaut lorsqu'il n'a pas reçu une donnée de capteur. Malheureusement, ces pertes de données conduisent à l'instabilité des fonctions de masse au cours du temps. La figure 4.4 illustre comment la temporisation peut améliorer la stabilité de nos croyances au cours du temps lors de la perte de données.

Ainsi, on peut voir une perte de courte durée (t = 11-12s) compensée partiellement par la temporisation. On voit aussi la croyance persister un certain temps après la perte totale de données (t > 20s). La croyance obtenue à t=20s persiste pour 8 secondes car c'est le temps maximum que l'on a fixé ici.

4.1.4 Analyse d'un exemple : détection de présence

Nous avons vu le comportement de la temporisation avec des fonctions de masse stables obtenues artificiellement. Nous appliquons maintenant cette temporisation à des fonctions de masse obtenues directement à partir des capteurs. Cela a du sens dans des cas comme le mouvement. Une personne ayant bougé sera sûrement encore présente la seconde d'après. De même pour l'ouverture/fermeture d'une fenêtre. On peut considérer que la personne sera présente au moins le temps de traverser la pièce si elle souhaite sortir.

Cependant, certains capteurs ne permettent pas de propager la croyance qu'ils génèrent dans le temps. C'est le cas par exemple du capteur situé à l'entrée de la pièce qui permet de détecter un passage. Si cela indique que la personne est présente dans la pièce au moment exact où elle est détectée, on ne peut pas savoir si celle-ci entre ou sort de la pièce. On ne temporisera donc pas la croyance issue de ce capteur. C'est d'ailleurs pour cela que l'on applique la temporisation directement aux capteurs et non pas à la fonction de masse fusionnée correspondant à la croyance globale sur un attribut de contexte. De plus, le temps de temporisation peut être différent d'un

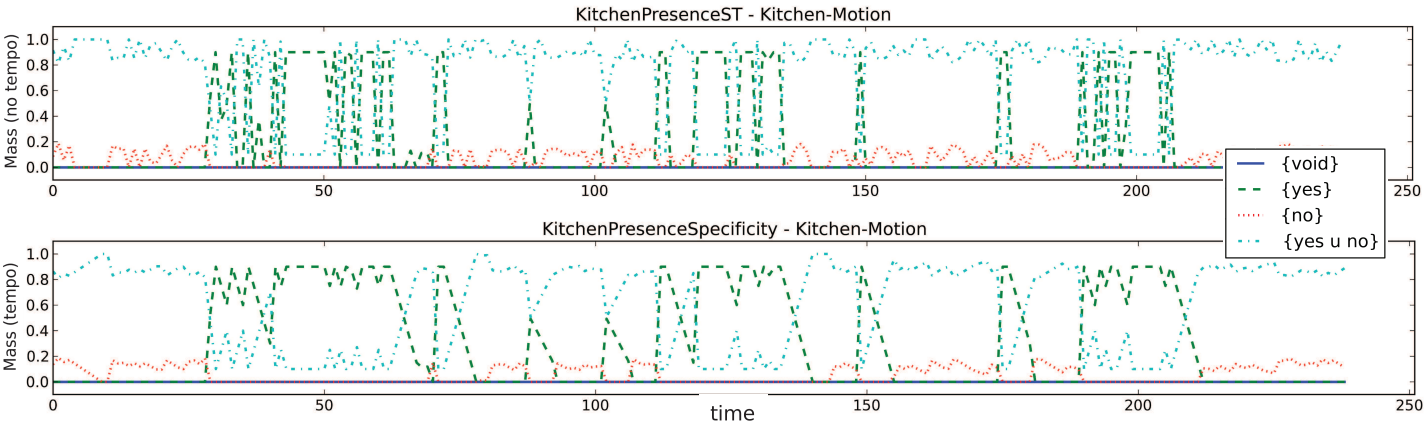

FIGURE 4.5 – Croyances induites par un capteur de mouvement pour la détection de la présence dans une pièce. La personne se déplace dans les intervalles [30-62], [120-140] et [190-215]. Une temporisation avec $T_{max} = 6s$ est appliquée.

capteur à l'autre en fonction de l'évènement physique mesuré.

Nous avons appliqué une temporisation de six secondes à notre capteur de mouvement. Les résultats sont donnés figure 4.5. On constate facilement que la croyance induite lorsqu'une temporisation est appliquée est beaucoup plus stable que lorsqu'il n'y en a pas. De la même manière, la figure 4.6 montre la croyance induite par une personne assise sur une chaise avec et sans temporisation. On constate ici que la temporisation permet de compenser les pertes de données de façon acceptable. Cela est d'autant plus important qu'un autre moyen couramment utilisé pour compenser les pertes de données dans les communications sans fil n'est autre que la multiplication des paquets envoyés. Ceci est bien entendu inacceptable dans notre cas où des nœuds de capteurs peuvent fonctionner sur batteries. Cet algorithme de temporisation est donc une meilleure solution avec un coût en calcul moindre et aucun recours à la multiplication des communications.

4.2 Temporisation avec fusion

Nous avons vu un premier algorithme permettant de propager les croyances dans le temps et d'effectuer une discrimination entre une croyance ancienne et une plus récente. Cela nous permet d'améliorer la stabilité des croyances du système sans augmenter la fréquence d'envoi des données des capteurs ni augmenter la quantité de fusions effectuées.

Cependant, cette discrimination peut être dans certains cas un peu trop restrictive. En effet, suite à la discrimination, une seule des deux fonctions de masse persiste et aucune trace n'est gardée de l'autre. Cela peut poser problème comme on l'a vu notamment lorsqu'un pic de bruit vient perturber la croyance de notre système. De plus, le fait d'observer plusieurs fois d'affilée la même chose, et donc avoir plusieurs fois de suite la même croyance, devrait renforcer cette croyance. Considérons par exemple un capteur de mouvement pour la détection de présence. L'absence de mouvement n'est pas en soi une bonne preuve de l'absence de toute personne. Par contre, l'absence de mouvement prolongée, elle, peut devenir une preuve plus recevable. Il

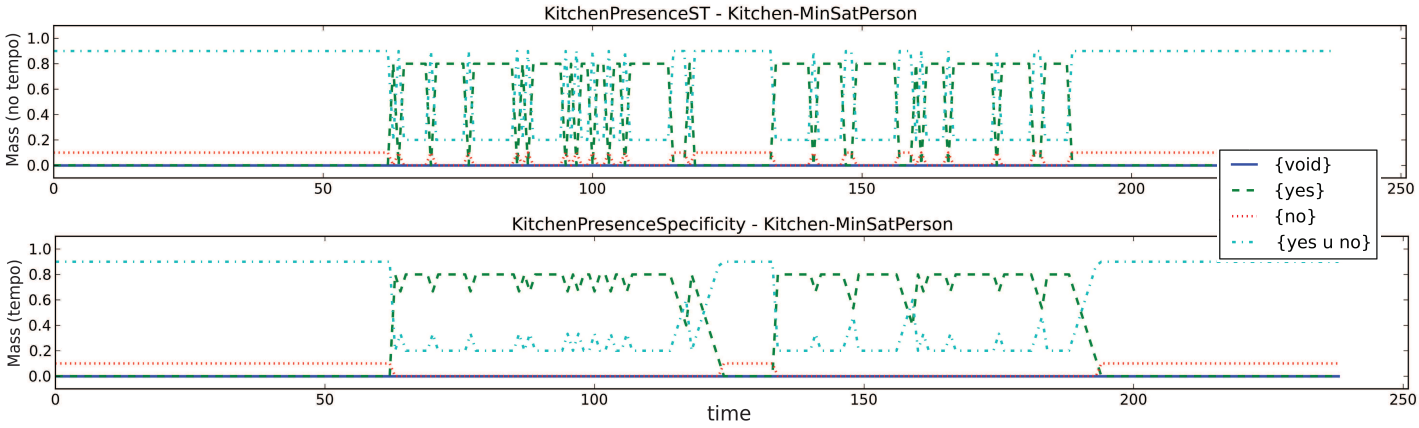

FIGURE 4.6 – Croyances induites par un capteur de pression sur une chaise pour la détection de la présence dans une pièce. La personne est assise dans les intervalles [62-120] et [135-190]. Une temporisation avec $T_{max} = 6s$ est appliquée.

faudrait donc que ce capteur puisse induire une croyance en l'absence de plus en plus prononcée au fur et à mesure qu'aucun mouvement n'est observé.

Ce type d'accumulation de preuve existe dans des méthodes de combinaison récursive [72, 73]. Bien que cela corresponde en partie à ce que nous cherchons à faire, ces méthodes reposent sur le fait que l'ensemble d'informations collectées dans le temps ne doit défendre qu'un seul des états possibles. Cela revient donc à accumuler des preuves sur une plus grande période de temps. Malheureusement, dans notre cas, nous ne savons pas si l'état réel du monde a changé ou pas, ni quand il change. De plus, il est évident qu'il doit changer à un moment donné. Si l'on cherche à détecter une présence, la pièce deviendra bien vide à un moment donné si l'on travaille sur une période suffisamment longue. Ce sera évidemment le cas dans un habitat intelligent connecté en permanence.

Dans cette section, nous proposons un autre algorithme de temporisation pour prendre en compte ces considérations.

4.2.1 Combinaison temporelle

Nous avons vu comment propager les fonctions de masse dans le temps. Nous gardons ici l'affaiblissement pour cette propagation. Ensuite, un des meilleurs moyens de conserver la trace d'un ensemble de fonctions de masse en une unique est de les combiner. Dans notre cas, on souhaite donc pouvoir combiner notre ancienne croyance affaiblie avec notre nouvelle croyance issue de la dernière mesure renvoyée par un capteur.

Nous avons vu que pour fusionner les fonctions de masse induites pas nos différents capteurs,

nous utilisons la règle de combinaison de Dempster donnée par :

$$m_1 \oplus m_2(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C) \quad (4.7)$$

$$avec\ K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \quad (4.8)$$

$$et\ m(\emptyset) = 0 \quad (4.9)$$

Le fait que cette règle ne soit pas idempotente (*i.e.* $m \oplus m \neq m$) est la propriété que l'on recherche pour que la croyance induite par notre capteur soit renforcée lorsque celle-ci se répète dans le temps. Malheureusement, cette règle pose problème en cas de conflit. Et le conflit arrivera forcément lors d'un changement d'état. Il nous faut donc une règle de combinaison offrant une bonne convergence mais capable aussi de gérer les conflits. Pour cela, nous avons choisi la règle de combinaison de Dubois et Prade [29]. Elle est donnée pour deux fonctions de masse m_1 et m_2 par :

$$m_{DP}(A) = \sum_{B \cap C = A} m_1(B) m_2(C) + \sum_{B \cup C = A,\ B \cap C = \emptyset} m_1(B) m_2(C)\ ,\ \forall A \subseteq \Omega \quad (4.10)$$

Cette règle de combinaison est à la fois conjonctive et disjonctive. Ainsi, elle convergence lorsqu'il n'y a pas de conflit et diverge sinon. Cette règle de combinaison a donc les propriétés recherchées pour la combinaison temporelle. Pour effectuer cette combinaison, nous appliquons alors l'algorithme 2. On fusionne donc les fonctions de masse issues d'un capteur au cours du temps. La temporisation permet d'introduire un facteur d'oubli aux croyances les plus anciennes. Nous évitons ainsi de conserver la trace de toutes les croyances précédentes. Ces dernières finissent par être oubliées avec le temps.

Algorithme 2 Temporisation avec fusion

m_{new} : la dernière fonction de masse obtenue
m_{old} : la dernière fonction de masse sauvegardée
t : le temps présent
t_{ref} : le temps de référence associé à m_{old}
T_{max} : le temps maximum de persistance de la croyance

$m_\alpha \leftarrow$ affaiblissement(m_{old}, t, t_{ref}, T_{max})
$m_{fused} \leftarrow$ CombinaisonDuboisPrade(m_α, m_{new})
$m_{old} \leftarrow m_{fused}$
$t_{ref} \leftarrow t$
return m_{fused}

4.2.2 Analyse du comportement de l'algorithme

Cet algorithme de temporisation avec fusion doit nous permettre d'augmenter l'engagement des fonctions de masse induites par les mesures de capteurs au cours du temps lorsqu'une stabilité est mesurée. De même, elle doit permettre d'éviter les effets négatifs des pics de bruit. Regardons en détail l'effet de cet algorithme sur les fonctions de masse.

Convergence

La première propriété qui nous intéresse est la convergence. Cela correspond au renforcement dans le temps d'une croyance qui sans temporisation serait stable. La figure 4.7 illustre cette convergence. Elle dépend de deux facteurs : T_{max} et le nombre de données reçues en T_{max} secondes. On peut regrouper ces deux facteurs en un seul comme suit :

$$R = T_{max}.F \tag{4.11}$$

Où F est la fréquence de mesures de nos capteurs. Ainsi, si l'on applique une temporisation de 8 secondes et que nos capteurs envoient leurs mesures à 2 Hz, on obtient $R = 16$. Plus ce rapport est grand, plus la convergence sera importante. Cela s'explique simplement par l'oubli plus lent des anciennes croyances.

Dans la temporisation avec discrimination, il n'est pas nécessaire de faire la différence entre une perte de données induisant une fonction de masse vide et une mesure de capteur induisant une fonction de masse vide. Dans le cas de la temporisation avec fusion, une fonction de masse vide induite par un capteur doit être considérée comme une réelle preuve et doit donc participer à la réduction de la croyance mais pas aussi rapidement qu'une perte de données. On peut remarquer dans l'algorithme 2 que l'affaiblissement se fait par rapport au temps écoulé entre deux mesures. Une perte de données ne doit donc pas amener le même résultat qu'une mesure n'apportant qu'une fonction de masse vide.

Le comportement de la temporisation avec fusion lors de la perte de données est illustré figure 4.8 et lors de l'absence de preuve réelle (*i.e.* une fonction de masse vide induite par la mesure d'un capteur) figure 4.9. Ainsi, on peut voir une décroissance linéaire lors de la perte de données correspondant à la fonction du temps linéaire utilisée pour l'affaiblissement. Dans le cas d'absence de preuve, on obtient une ligne polygonale due à l'affaiblissement linéaire appliqué entre deux mesures reçues.

Changement d'état

Le changement d'état est important et c'est d'ailleurs ce qui souvent permet de déclencher des services comme la lumière par exemple. Comme nous l'avons vu au chapitre 1, la détection d'une présence doit permettre d'allumer la lumière si nécessaire. Cela doit être suffisamment réactif pour ne pas déranger les habitants. Le changement d'état ne doit donc pas être empêché par la temporisation. Il est donc important de mesurer l'impact de la temporisation sur la possibilité qu'a le système à changer d'état en même temps que le monde réel.

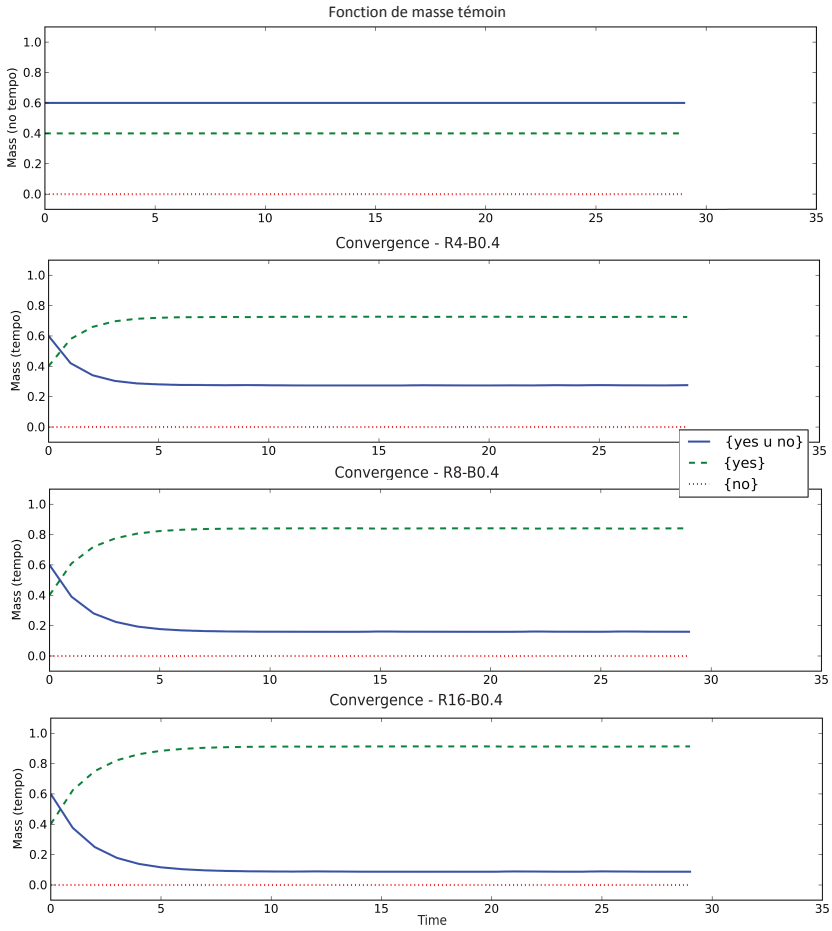

FIGURE 4.7 – Convergence de la fonction de masse associée à un capteur au cours du temps avec des rapports R de 4, 8 et 16.

La figure 4.10 illustre le changement d'état dominant au sein des croyances lorsque celui-ci change réellement. On constate qu'une fois de plus, le rapport R est déterminant pour la réactivité du système. En effet, plus R est grand, plus le système mettra de temps à changer d'état. Cela peut s'expliquer par le fait que plus R est grand, plus la convergence de la croyance est grande elle aussi. De fait, l'oubli étant plus lent, l'ancien état persiste plus longtemps dans les croyances.

Ce manque de réactivité peut sembler problématique mais on peut tout de même constater que grâce à la règle de combinaison utilisée, celle de Dubois et Prade, un changement d'état brusque entraîne une croyance moins spécifique pendant quelques secondes (ou quelques unités de temps, le rapport R dépend en effet à la fois de T_{max} mais aussi de la fréquence de réception des données issues des capteurs). On peut voir ainsi que le système accorde d'abord une croyance forte à l'union des deux états avant de changer complètement d'opinion. Cela correspond finalement à ce que font la plupart des humains quand ils ne sont pas certains de ce qu'ils observent. C'est avec le temps que le changement d'état se confirme.

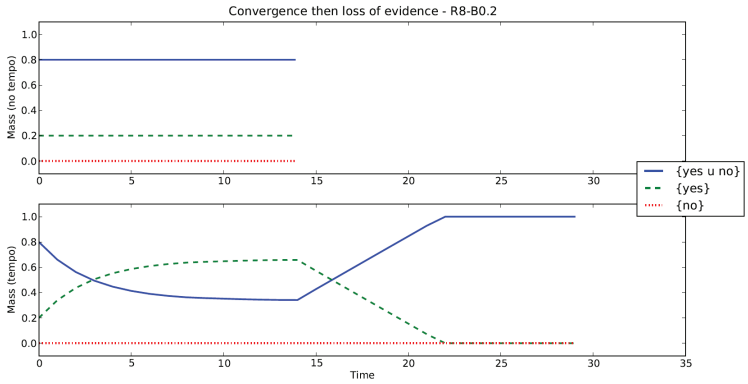

FIGURE 4.8 – Convergence de la fonction de masse associée à un capteur au cours du temps avec un rapport R de 8 puis plus aucune mesure n'est reçue.

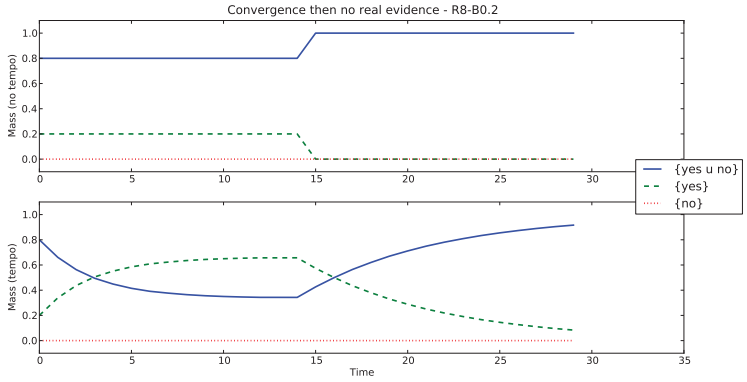

FIGURE 4.9 – Convergence de la fonction de masse associée à un capteur au cours du temps avec un rapport R de 8 puis plus aucune réelle preuve n'est reçue.

La figure 4.10 illustre le changement d'état lorsque les deux niveaux de croyance induits par les mesures de capteurs sont identiques. Cependant, il est possible d'introduire une asymétrie dans les niveaux de croyances pour modifier la vitesse de convergence du nouvel état. Les figures 4.11 et 4.12 illustrent respectivement un changement d'état lent et un changement d'état rapide.

La temporisation avec fusion permet donc de jouer sur la vitesse de changement d'état de deux manières différentes : soit avec le rapport R défini précédemment, soit en créant une asymétrie dans les niveaux de croyance des états. Une transition d'un état à croyance forte vers un état à croyance faible sera plus lente que l'inverse. En effet, la croyance reste dominée nettement plus longtemps par l'union des deux états dans le premier cas.

La vitesse de transition d'un état à l'autre peut être déterminante dans certains cas. Par exemple, si l'on souhaite avoir un service de lumière reposant en grande partie sur la présence, il est nécessaire d'avoir une réactivité importante pour la détection de la présence. L'absence peut

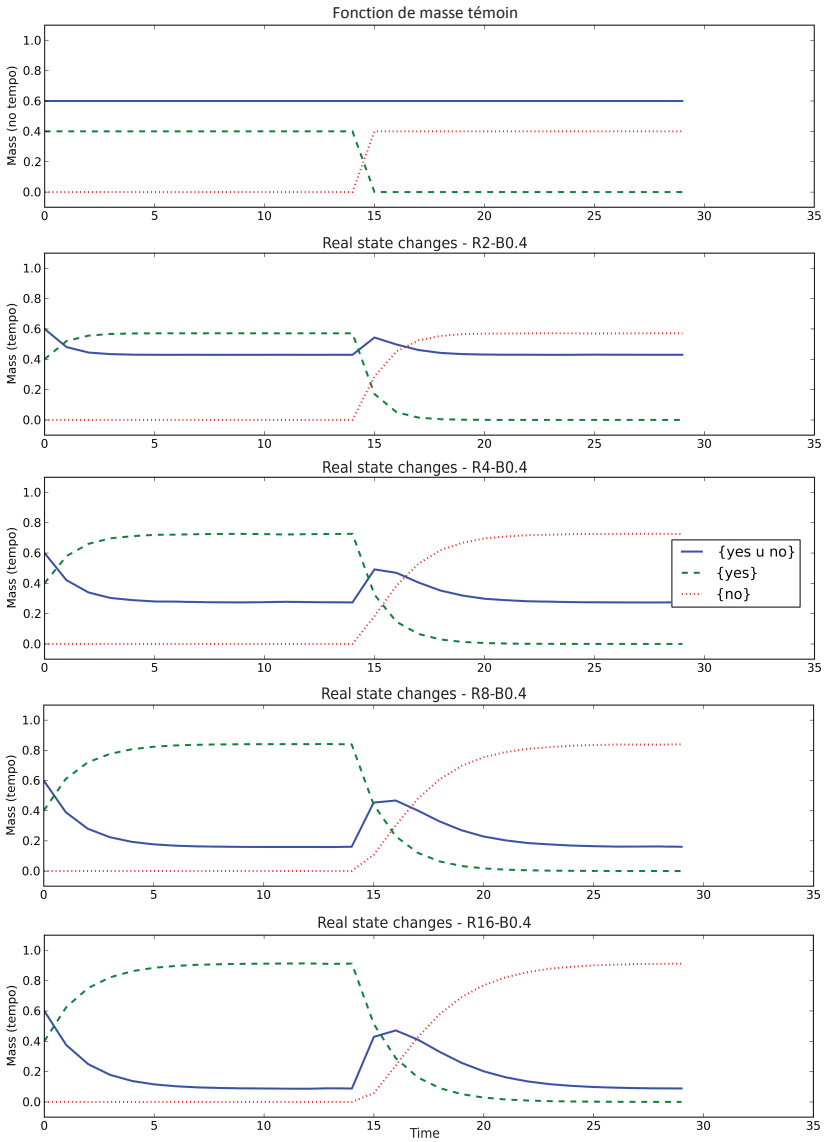

FIGURE 4.10 – Changement d'état lorsqu'une temporisation avec fusion est appliquée. Les rapports R sont de 2, 4, 8 et 16.

mettre un peu plus de temps à s'installer sans réduire drastiquement la qualité du service. En effet, il est plus important d'allumer instantanément la lumière quand une personne entre dans une pièce que de l'éteindre instantanément quand la personne quitte la pièce.

Effet du bruit

Nous avons vu que, contrairement à la fusion avec discrimination, la temporisation avec fusion ne permet pas un changement d'état immédiat. Ce comportement est efficace lorsqu'il s'agit de contrecarrer les effets des pics de bruit. La figure 4.13 illustre une convergence, suivie

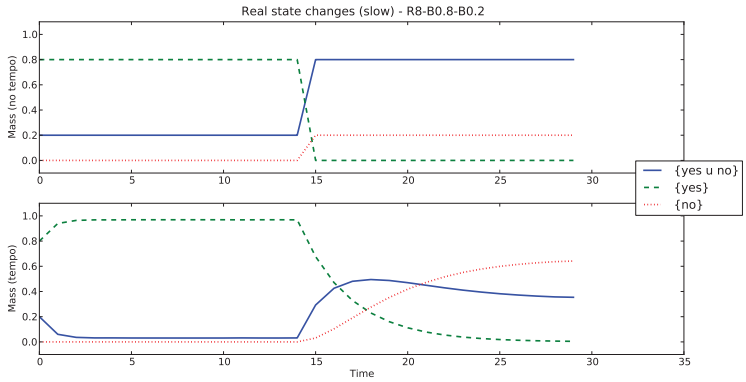

FIGURE 4.11 – Effet de la temporisation avec fusion sur la transition d'un état à croyance faible vers un état à croyance forte (R = 8).

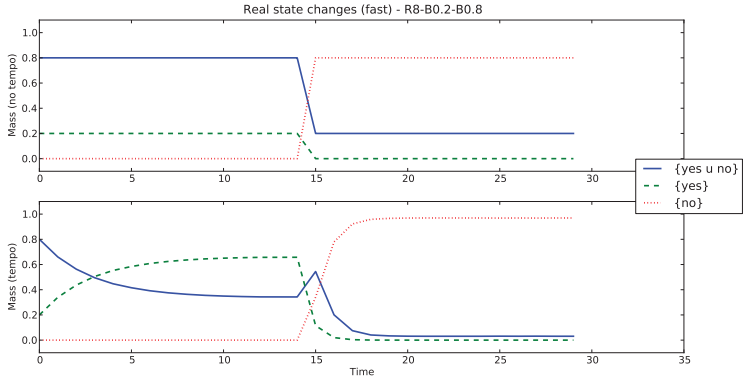

FIGURE 4.12 – Effet de la temporisation avec fusion sur la transition d'un état à croyance forte vers un état à croyance faible (R = 8).

d'un changement d'état lui-même suivi de deux pics de bruit : l'un du même niveau de croyance que la croyance reçue sans temporisation, l'autre d'un niveau de croyance nettement supérieur. Là où la temporisation avec discrimination changeait l'état reconnu comme réel par le système de façon immédiate et le conservait pour quelques temps, la temporisation avec fusion, elle, introduit un doute temporaire avant de permettre une nouvelle convergence de l'état dominant.

La sensibilité aux pics de bruit est d'autant plus faible que le rapport R est élevé. Cela est cohérent avec la vitesse de changement d'état en fonction de R.

En résumé, avec la temporisation avec fusion, nous avons réussi à offrir une convergence des croyances lorsque celles-ci se répètent dans le temps mais aussi à résoudre le problème de l'influence des pics de bruit sur la croyance générale de notre système.

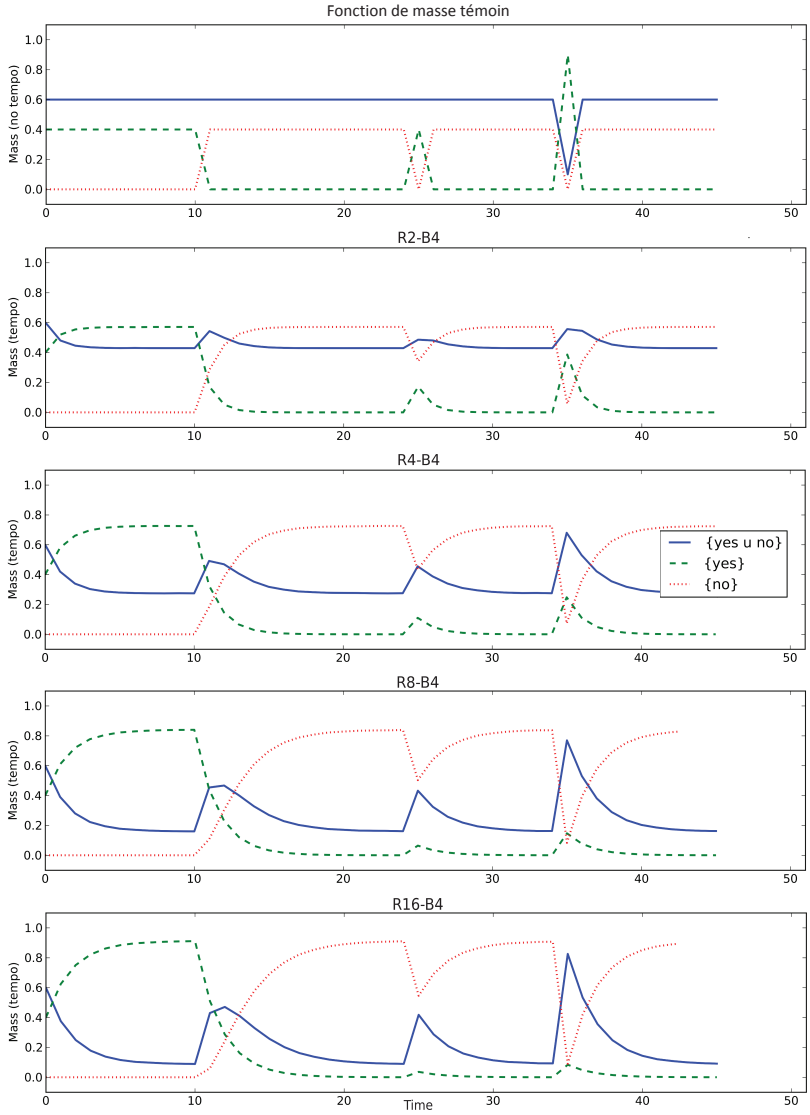

FIGURE 4.13 – Effet du bruit sur la fonction de masse résultant de la temporisation avec fusion. Les rapports R sont de 2, 4, 8 et 16.

4.2.3 Analyse d'un exemple : mouvement et présence

Nous avons vu que la détection de la présence peut être difficile et instable suivant les capteurs que l'on utilise. Notre premier algorithme de temporisation a permis d'apporter un peu plus de stabilité aux preuves reccueillies.

Malheureusement, si l'on cherche à détecter la présence, on cherche aussi à détecter l'absence. Cela est beaucoup plus délicat car cela correspond finalement à une absence de preuve pour la plupart des capteurs usuellement utilisés pour la détection de présence. Si l'on considère par exemple un capteur de mouvement, l'absence de mouvement ne permet malheureusement pas

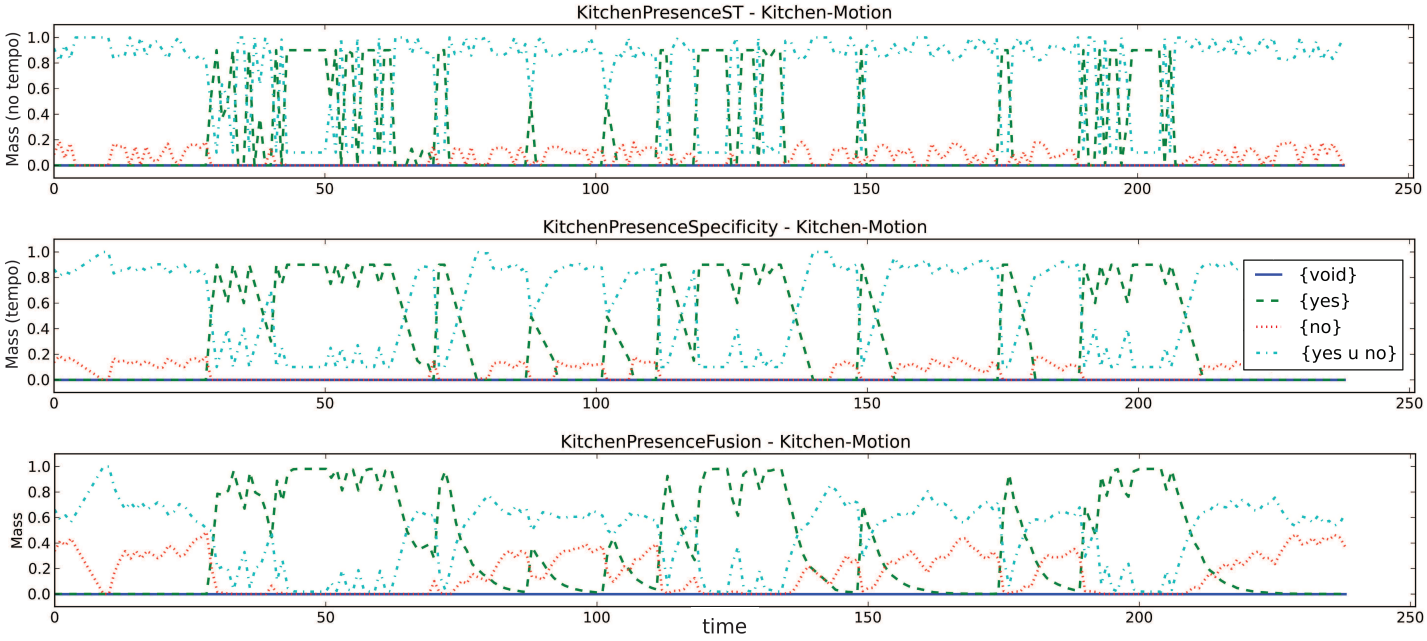

FIGURE 4.14 – Croyances induites par un capteur de mouvement pour la détection de la présence dans une pièce. La personne se déplace dans les intervalles [30-62], [120-140] et [190-215]. La courbe du haut représente la croyance induite par le capteur sans temporisation, celle du milieu avec une temporisation avec discrimination, celle du bas avec une temporisation avec fusion (dans les deux cas, $T_{max} = 6s$).

d'affirmer avec une grande confiance l'absence de quelqu'un dans la pièce. En effet, une personne peut être présente sans pour autant se déplacer.

Cependant, l'absence de mouvement à long terme peut être une meilleure preuve de l'absence. La convergence offerte par la temporisation avec fusion nous permet donc d'apporter une meilleure preuve de l'absence avec un capteur de mouvement. Le résultat de l'application de la temporisation avec fusion est illustré figure 4.14. Grâce au modèle de croyance utilisé pour le capteur du mouvement, on profite de l'asymétrie des niveaux de croyance entre la présence et l'absence pour permettre une détection rapide de la présence et une diminution plus lente de la croyance concernant la présence lorsqu'aucun mouvement n'est observé. Sur cet exemple, on observe donc une meilleure convergence de la détection de la présence, une meilleure stabilité mais aussi la possibilité de mieux détecter l'absence. Nous essayons donc d'utiliser la temporisation avec fusion en priorité sur la temporisation avec discrimination si cela a du sens pour le capteur auquel on l'applique. L'accumulation de preuves dans le temps a du sens pour un capteur de mouvement mais n'en aurait pas pour un capteur de contact sur une fenêtre par exemple.

4.3 Mise à l'épreuve des algorithmes : détection de la présence

Nous avons vu comment temporiser les croyances de notre système de deux manières différentes. Nous avons pu constater une nette amélioration de la qualité des croyances du système grâce à ces mécanismes notamment lors de l'utilisation de preuves instables mais aussi lors de la perte de données. Dans cette section, nous souhaitons mettre à l'épreuve nos algorithmes sur un cas réel que nous empirons encore un peu plus en y ajoutant des pertes de données supplémentaires.

La première courbe des figures 4.15 et 4.16 illustre la détection de la présence dans une cuisine en utilisant les capteurs décrits au chapitre précédent. Le modèle associé à chaque capteur est le même dans les deux cas. Une temporisation a été appliquée dans le second cas en fonction de chaque capteur. On constate sans problème que la temporisation :

- stabilise les croyances et permet de compenser l'effet des pertes et l'instabilité des capteurs ;
- permet d'avoir un niveau de croyance acceptable sur l'absence sans pour autant le faire converger trop vite.

Maintenant que nous savons que les deux algorithmes de temporisation proposés permettent de réduire l'effet des pertes de données, nous avons voulu tester jusqu'à quel point cela le permet. Les figures 4.15 et 4.16 illustre la croyance de notre système concernant la présence dans la pièce respectivement sans et avec temporisation. Afin d'évaluer nos algorithmes, nous avons rejoué les données réelles issues des capteurs en rajoutant artificiellement des pertes de données en plus des pertes réelles.

Sans temporisation, on constate que la croyance résultant de l'accumulation de preuves devient très vite trop instable pour être exploitable. En appliquant l'algorithme de prise de décision, le système changerait de réponse sans cesse. Avec la temporisation, l'état dominant la croyance reste suffisamment stable pour permettre une prise de décision stable dans le temps y compris avec 50% de pertes. En effet, on constate une nette amélioration de la réponse du système. Les absences et présences restent nettement dessinées.

Bien que 50% de pertes puisse paraître élevé, cela n'est pas un cas absurde et peut être obtenu en condition réelle en fonction des environnements plus ou moins favorables aux communications radio. En effet, les technologies radio courte portée peuvent facilement être perturbées par des équipements divers, des objets en métal (armoires, placards, etc) ou encore par les personnes elles-mêmes. Si nos algorithmes semblent répondre correctement à nos problèmes d'instabilité des croyances, leur impact reste cependant limité lors de la perte d'un grand nombre de données d'affilée. Ainsi, on observe quand même quelque "trous" dans les croyances de notre système lors de la perte d'une grand nombre de données.

4.4 Conclusion

Dans ce chapitre, nous avons présenté deux algorithmes permettant de propager dans le temps des fonctions de croyance. Nous avons montré qu'il est possible de conserver efficacement une trace d'anciennes croyances quand de nouvelles sont disponibles.

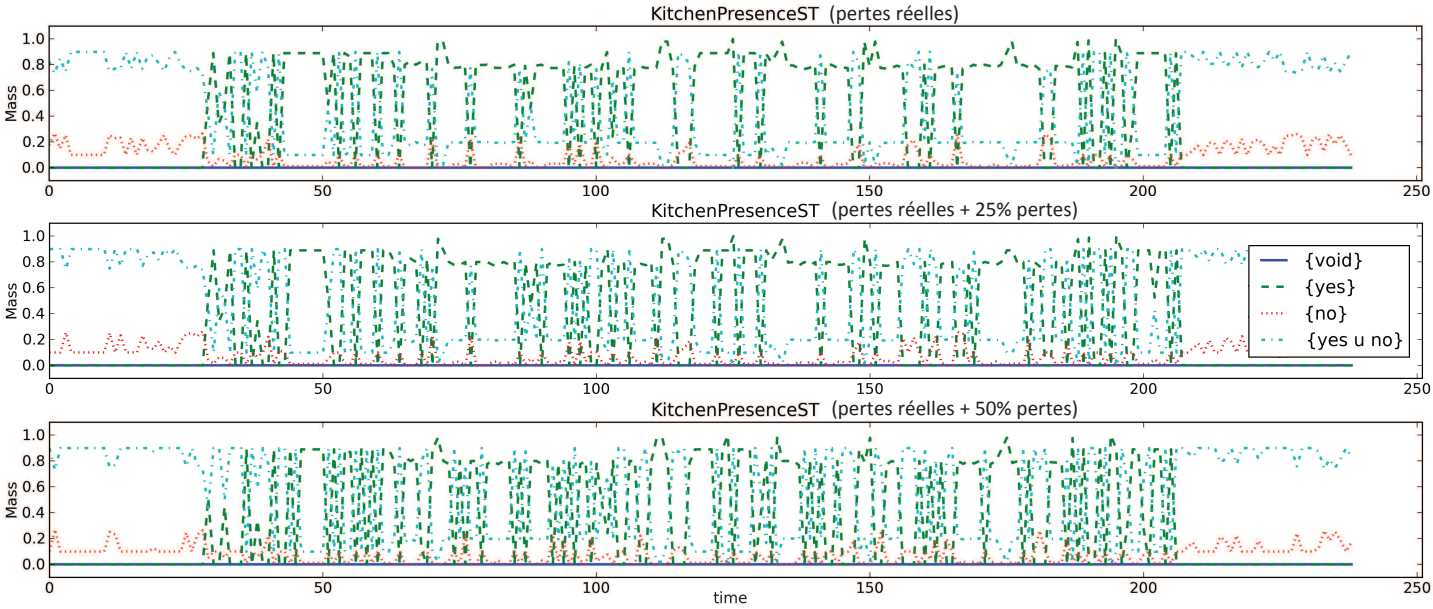

FIGURE 4.15 – Croyance associée à la détection de présence avec les pertes réelles puis avec 25% et 50% de pertes simulées. Aucune temporisation n'est appliquée.

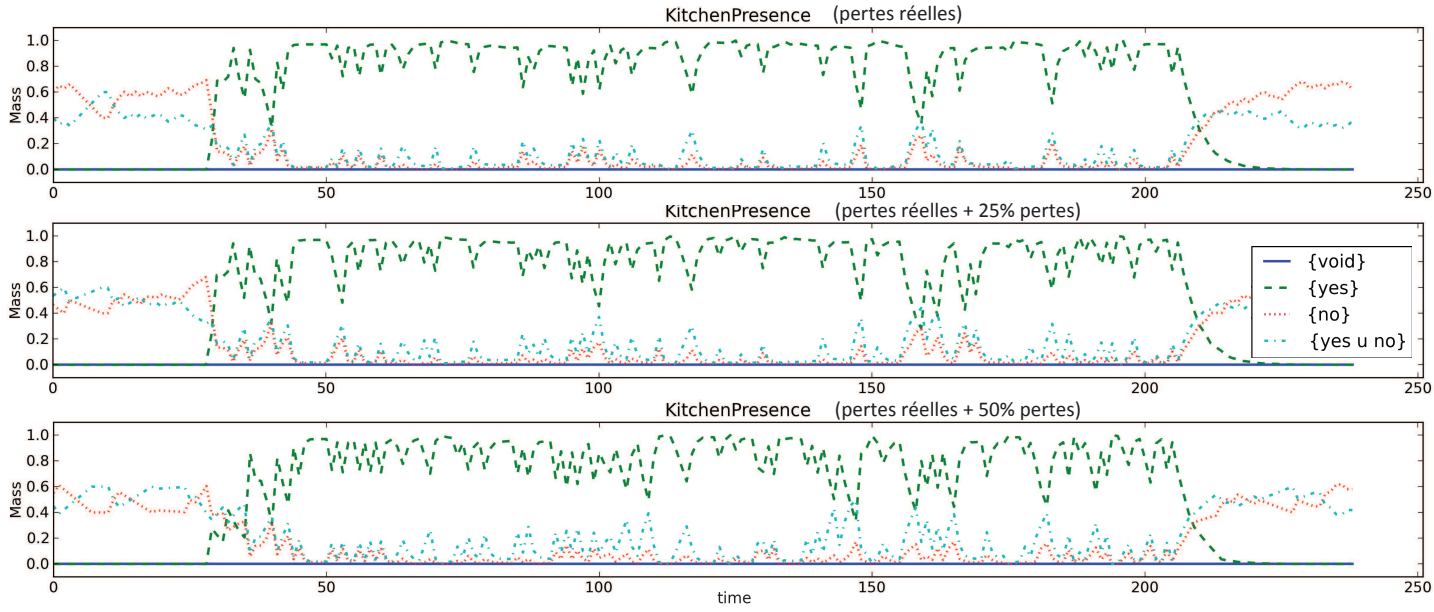

FIGURE 4.16 – Croyance associée à la détection de présence avec les pertes réelles puis avec 25% et 50% de pertes simulées. Une temporisation avec fusion ($T_{max} = 6s$) a été appliquée aux capteurs de mouvement et du nombre de personnes assises. Une temporisation avec discrimination ($T_{max} = 4s$) a été appliquée au capteur de contact sur la fenêtre et aux capteurs de vibration sur les chaises.

Nous avons explicité l'effet de ces algorithmes sur la construction des fonctions de masse à partir des données des capteurs. Ces mécanismes de temporisations permettent de :

- stabiliser les croyances issues de données de capteurs instables ;
- conserver une trace des croyances anciennes lors de la perte de données ;
- prendre en compte la répétition d'une preuve dans le temps et ainsi faire converger la croyance associée.

Enfin, nous avons montré la robustesse des algorithmes proposés, y compris avec des pertes de données importantes, et la pertinence d'une telle approche pour la construction de croyances au cours du temps notamment pour la détection de l'absence. Cependant, bien que ces algorithmes offrent de belles performances, en termes de compensation des pertes de données notamment, ils ne sont pas toujours applicables. En effet, tous les capteurs ne permettent pas de déduire une croyance pour plus longtemps que l'instant auquel une mesure a été effectuée. C'est le cas notamment du capteur de passage dans la porte. De même, si le second algorithme présenté semble être meilleur que le premier, en termes de convergence et de réponse aux pics de bruit, il ne doit être appliqué qu'aux capteurs dont la répétition d'une preuve a un sens réel. C'est le cas comme on l'a vu pour nos capteurs de mouvement, ce n'est pas le cas pour un capteur de contact sur une fenêtre dont seule la variation nous intéresse. En effet, il est peu probable que quelqu'un s'amuse à ouvrir et fermer une fenêtre plusieurs fois d'affilé rapidement. Le premier algorithme est en effet plus léger en calcul que le second et doit donc être privilégié lorsque la temporisation avec fusion n'apporterait rien.

Dans la partie suivante, nous décrivons comment nous avons mis en place l'ensemble des mécanismes précédemment présentés dans un prototype. Nous y présontons les outils technologiques et techniques que nous avons utilisées ainsi que la manière dont nous avons pu construire un prototype en respectant l'ensemble des contraintes présentées dans les deux premiers chapitres.

Troisième partie

Construction d'un prototype

Chapitre 5

Prototype : matériels et outils

Every building is a prototype. No two are alike.
Helmut Jahn

Men have become the tools of their tools.
Henry David Thoreau

Dans les chapitres précedents, nous avons étudié la mise en place de méthodes de fusion de données afin d'obtenir des données contextuelles abstraites facilement exploitables par des humains ou des services. Pour obtenir ces abstractions, nous avons mis en avant la nécessité de s'appuyer sur différents niveaux de fusion de données comme suggéré dans l'architecture de Coutaz *et al.*. Nous nous sommes ensuite focalisés sur la fusion de données de "bas niveau" exploitant des données brutes issues de capteurs et exploitant la théorie des fonctions de croyance. Nous avons ensuite introduit des algorithmes permettant d'améliorer la stabilité de nos attributs de contexte.

Afin de valider notre approche, nous avons déployé un système dans un environnement réel, avec de vrais capteurs et des personnes se déplaçant dans l'environnement, tout en essayant au maximum de respecter les contraintes que nous nous sommes fixées, c'est-à-dire en nous limitant à une instrumentation légère composée de capteurs élémentaires imparfaits, distribués dans l'espace physique. Pour cela, nous nous sommes reposés sur un ensemble de technologies et d'outils déjà existants.

L'objectif de ce chapitre est donc de présenter les outils technologiques mais aussi logiciels que nous avons utilisés et qui ont permis et facilité le déploiement de notre prototype pour un habitat intelligent. Nous commençons avec les divers matériels que nous avons utilisés et présentons ensuite les principaux outils logiciels que nous avons déployés. Le chapitre suivant sera, lui, consacré à l'implémentation de ce prototype et aux apports de notre équipe au niveau logiciel. Comme dans les chapitres précédents, la suite du document se concentre sur les couches

basses de l'architecture (capteurs et premières abstractions) même si aborderons les services déployés à titre d'exemple.

5.1 Technologies, matériels et équipements

Dans cette section, nous présentons les matériels et équipements utilisés pour mettre en place notre prototype d'habitat intelligent. Le but n'est pas ici de faire un état de l'art en matière de technologie mais uniquement de présenter de manière synthétique les composants matériels sur lesquels nous nous sommes appuyés.

5.1.1 Capteurs

Comme nous l'avons dit de nombreuses fois dans les chapitres précédents, notre système repose, au plus bas niveau, sur un ensemble de capteurs. Nous souhaitions utiliser des capteurs bas coût, facilement déployables et exploitables. Pour cela, nous avons utilisé la plupart du temps des Phidgets [39] comme par exemple des capteurs de mouvement, de niveau sonore, de pression sur les chaises, de vibration, de distance (utilisés pour le passage dans les portes), etc. Ces capteurs analogiques ont l'avantage d'être peu coûteux et ils sont très simples à exploiter. La figure 5.1 présente quelques uns des capteurs que nous avons déployés.

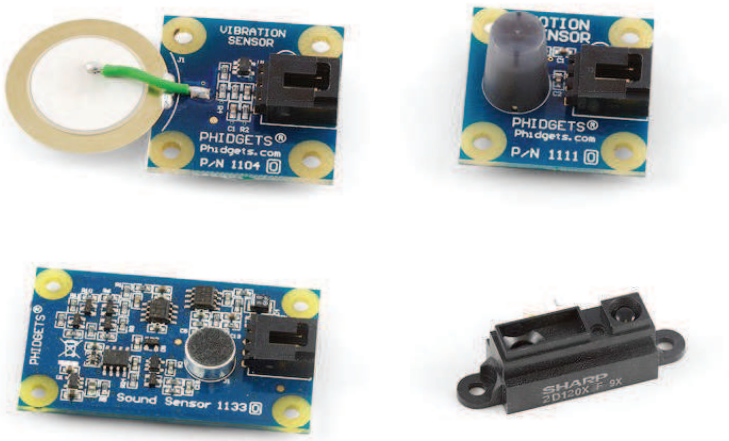

FIGURE 5.1 – Capteurs phidgets. En haut à gauche, un capteur de vibration, en haut à droite, un capteur de mouvement, en bas à gauche, un capteur de niveau sonore, en bas à droite, un capteur de distance 10-80cm.

Nous avons aussi utilisé quelques capteurs binaires comme les contacteurs sur les portes et les fenêtres, certains capteurs de mouvement ou encore des capteurs de vibration. Nous nous sommes donc limités à des capteurs élémentaires pour la mise au point de notre prototype.

5.1.2 Nœuds communicants

Afin de permettre à tous les capteurs déployés de communiquer leurs mesures, nous avons connecté ces derniers à des nœuds communicants. Le but est aussi ici de regrouper les capteurs afin de réduire l'instrumentation au maximum. La figure 5.2 montre un exemple de nœuds que nous avons déployés. On y voit deux types de nœuds, un Wismote [1] et un Z1 [2]. Ces nœuds, fonctionnant à l'aide du système d'exploitation Conkiti [31], communiquent via le protocole IEEE 802.15.4, un protocole radio courte portée et bas débit qui ne consomme que peu d'énergie. Ils forment un réseau plus connu sous le nom de 6LoWPAN [79] (*IPv6 Low power Wireless Personal Area Networks*).

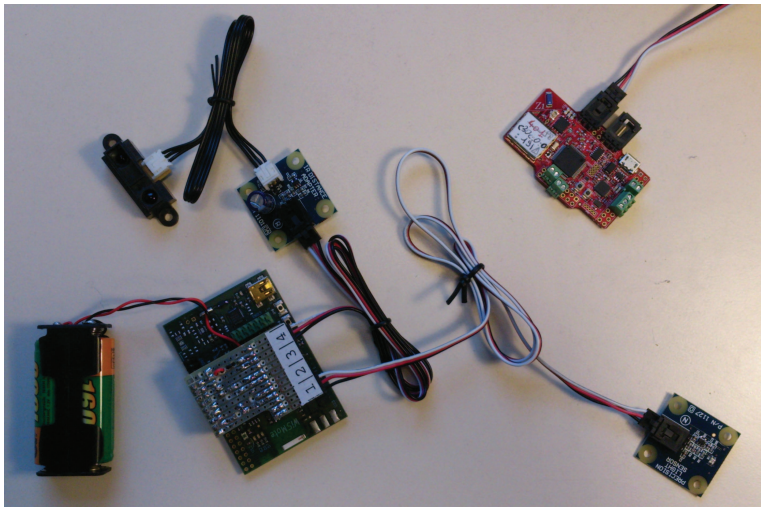

FIGURE 5.2 – Nœuds communicants. A gauche, un Wismote sur batteries et muni d'un capteur de distance et un capteur de luminosité. A droite, un nœud Z1.

Dans notre cas, nous utilisons un réseau maillé de nœuds de capteurs. Le routage se fait à l'aide du protocole RPL [96] (*Routing Protocol for Low power and Lossy Networks*) spécialement conçu pour ce type de réseaux et permettant de répondre aux besoins des réseaux domotiques [3]. Ce protocole construit un graphe orienté acyclique (*Directed Acyclic Graph* ou DAG) afin de définir les routes. Ce type de réseau n'étant pas compatible directement avec les réseaux plus classiques tels qu'Ethernet et Wifi, un routeur de bordure (*border router* en anglais) est nécessaire pour créer le lien entre ce réseau et le reste des équipements n'exploitant pas la technologie 6LoWPAN. C'est le routeur de bordure qui constitue la racine du graphe représentant les routes. La figure 5.3 illustre un exemple de routes que nous avons obtenues avec notre plate-forme. Chaque point représente un nœud communicant et les flèches représentent les routes empruntées par les paquets.

1. http ://www.aragosystems.com/fr/wisnet-item/wisnet-wismote-item.html
2. http ://www.zolertia.com/ti
3. http ://tools.ietf.org/html/rfc5826

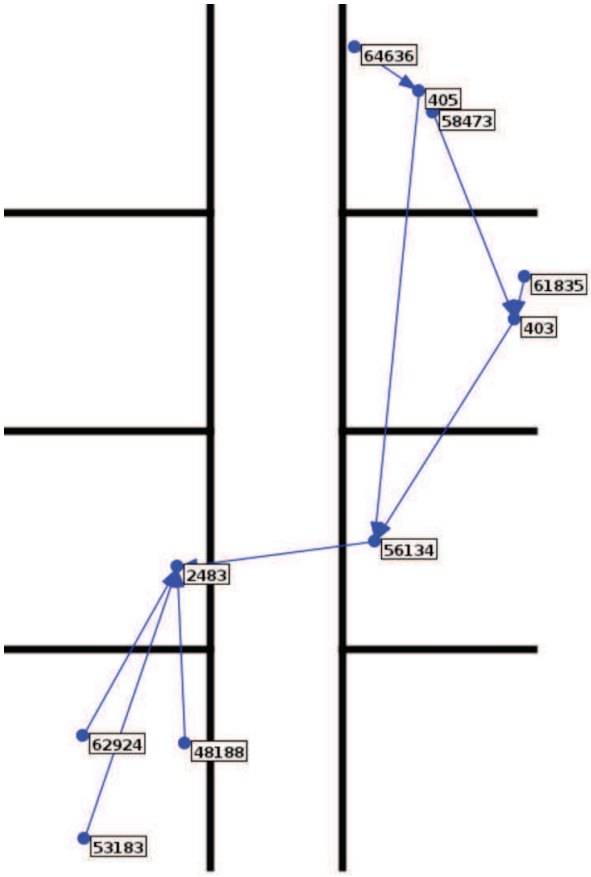

FIGURE 5.3 – Exemple de routes obtenues lors du déploiement d'un réseau 6LoWPAN dans nos bureaux. Le nœud 2483 est le routeur de bordure.

5.1.3 Calculateurs

Les nœuds communicants utilisés étant très limités en capacités (CPU et mémoire), il n'était pas concevable de leur faire supporter l'ensemble des calculs nécessaires aux différentes couches de fusion de données. Tout en restant dans le cadre des contraintes de notre architecture (*cf* chapitre 1 et 2), nous avons donc opté pour de petits ordinateurs peu coûteux. Nous avons donc utilisé des GuruPlug [4] (ARM 1.2GHz), des PandaBoard [5] (ARM 1.2GHz) et enfin des Raspberry Pi [6] (ARM 700MHz). Ce dernier est particulièrement prometteur. En plus d'avoir des capacités largement suffisantes pour nos applications, son prix est dérisoire ($\approx$ 25\$). Ce prix pourrait être encore réduit en enlevant les fonctionnalités qui nous sont pour le moment inutiles comme la gestion de la vidéo HD et la sortie HDMI. Ces petits calculateurs doivent nous permettre de tester notre approche sur une architecture limitée et de valider le fait que les performances de

4. http ://www.globalscaletechnologies.com/t-guruplugdetails.aspx
5. http ://pandaboard.org/
6. http ://www.raspberrypi.org/

nos outils logiciels sont suffisantes pour fonctionner sur ce type d'équipements peu coûteux et ne consommant que très peu d'énergie.

5.1.4 Equipements augmentés

Pour le développement de notre prototype, nous ne disposions malheureusement pas de véritables équipements augmentés capables de fournir des attributs de contexte. Nous les avons donc simulé soit avec des applications Android comme cela a été le cas pour la plaque de cuisson et la radio (figure 5.4) soit avec de petits boîtiers à LED comme cela a été le cas des radiateurs (figure 5.5).

FIGURE 5.4 – Capture d'écran de l'application Android permettant de simuler une plaque de cuisson et une radio.

Nous avons opté pour cette solution qui nous permettait de démontrer les concepts de notre prototype sans avoir de problème de compatibilité de matériels.

5.2 Outils logiciels

Nous utilisons un ensemble de technologies et matériels "sur étagère" pour construire notre prototype. Nous avons aussi utilisé quelques outils logiciels déjà disponibles. Dans cette section, nous présentons rapidement l'implémentation utilisée pour la reconnaissance de situations mais aussi le paradigme de communication que nous avons utilisé pour permettre à l'ensemble de nos composants d'échanger des données.

5.2.1 Implémentation des Context Spaces : ECSTRA

Pour les couches supérieures, nous nous sommes appuyés sur la théorie des *Context Spaces*. Cette dernière est présentée au chapitre 2. Elle permet de représenter des contextes et des si-

FIGURE 5.5 – Boîtier à LED permettant de simuler les états d'un radiateur augmenté.

tuations sous forme de solide dans un espace dont les dimensions peuvent être des données de capteurs des attributs de contexte ou encore des situations. Une implémentation appelée ECSTRA est disponible [11]. Elle offre tous les outils développés dans la théorie [69, 70, 68] et a même fait l'objet de quelques extensions en partenariat avec l'auteur de l'implémentation [27].

Cette implémentation permet une distribution complète des fonctionnalités du moteur de raisonnement. Cela est particulièrement intéressant dans notre cas afin de pouvoir répartir sur différents calculateurs peu puissants le raisonnement sur les situations mais aussi afin d'éviter la perte complète d'une couche de données contextuelles en cas de panne d'un équipement. Avec cette implémentation, il est ainsi possible de répartir la reconnaissance de situation sur des équipements augmentés, de petits calculateurs mais aussi un *smartphone*.

5.2.2 Exploitation d'un moteur publish/subscribe

Afin que chaque partie logicielle de notre prototype soit capable de communiquer avec toutes les autres, nous avons choisi d'utiliser le paradigme *publish/subscribe* [32]. Ce dernier repose sur l'idée que des émetteurs (les *publishers*) publient des données auxquelles d'autres programmes peuvent s'abonner (les *subscribers*).

Choix du paradigme

La figure 5.6 illustre le fonctionnement du mécanisme de communication publish/subscribe. Les émetteurs envoient des données dans un espace global. Les abonnés reçoivent les données auxquelles ils se sont abonnés sous forme de notifications. Ce modèle de communication offre

un avantage majeur : les émetteurs n'envoient pas leurs données en fonction des abonnés. Ils n'ont même pas besoin de savoir s'ils ont effectivement des abonnés. Ainsi, chaque émetteur de données peut être développé sans avoir à se soucier de quels programmes recevront ces données. Ce sont les abonnés qui s'adaptent aux données reçues.

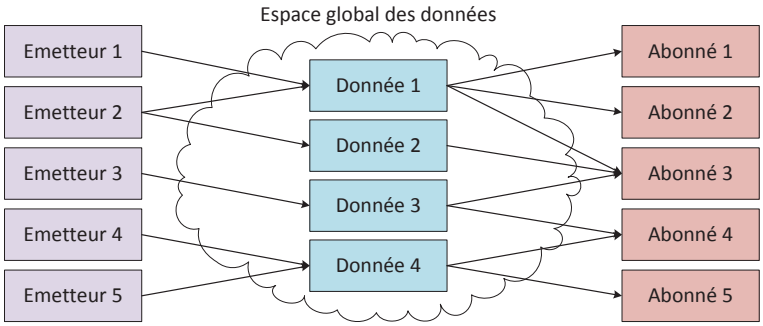

FIGURE 5.6 – Principe de fonctionnement du mécanisme *publish/subscribe*.

Dans notre cas, la plupart de nos programmes seront à la fois émetteurs et abonnés afin de créer une chaîne de fusion de données. Chaque programme pourra s'abonner seulement à ce dont il a besoin. Par exemple, chaque service ne s'abonnera qu'aux données contextuelles pertinentes pour ce service. On peut par exemple imaginer qu'un service permettant à un téléviseur d'éteindre uniquement l'écran quand plus personne n'est présent dans la pièce s'abonnerait certainement à l'attribut de contexte présence de la pièce dans laquelle il se situe ainsi qu'à son propre état. Certaines situations pourront aussi certainement l'intéresser pour se couper entièrement, par exemple, lorsque que le repas a lieu dans une autre pièce.

Implémentation Avis/Elvin

Pour mettre en place le paradigme de communication *publish/subscribe*, nous avons utilisé l'implémentation Avis [7] de Elvin [8]. Cette implémentation est décomposée en deux parties : la partie serveur et la partie cliente.

Le serveur Elvin se charge de recevoir l'ensemble des données et effectue le filtrage des messages ainsi reçus avant de les renvoyer vers les clients concernés. Les clients, de leur côté souscrivent aux données qui les intéressent à l'aide d'un langage de souscription spécifique [9].

Chaque message est constitué d'un ensemble de champs chacun composé d'un nom (sous forme de chaîne de caractères) et d'une valeur qui peut être un entier (int32 ou int64), un réel (real64) ou une chaîne de caractères (string). Chaque souscription se fait alors sous forme d'une expression logique permettant de filtrer les messages en fonction des champs et des valeurs voulues. Par exemple, pour souscrire aux attributs de contexte *PrésenceCuisine* et *PrésenceSalon*,

7. http ://avis.sourceforge.net/index.html
8. http ://elvin.org/specs/index.html
9. http ://avis.sourceforge.net/subscription_language.html

on pourrait avoir une souscription de la forme suivante :

```
NomAttributContexte == "PrésenceSalon" ||NomAttributContexte == "PrésenceCuisine"
```

Avec cette souscription, le client ne recevra que les messages ayant un champ appelé "NomAttributContexte" et ayant pour valeur "PrésenceSalon" ou "PrésenceCuisine". Dans notre application, chaque message des nœuds de capteurs a la forme suivante :

```
                    inria.aces.type : data.raw
                  inria.aces.format : sensor.node.data.raw
               inria.aces.sensor.node.ID : 174324
         inria.aces.sensor.node.nbSensors : 2
inria.aces.sensor.node.sensorType.0 : DistancePhidget
    inria.aces.sensor.node.data.raw.0 : 219.000000
inria.aces.sensor.node.sensorType.1 : MotionPhidget
    inria.aces.sensor.node.data.raw.1 : 1601.000000
```

Les principaux avantages d'Elvin sont le fait que le filtrage se fasse par le serveur de façon transparente pour les clients (sous réserve d'avoir effectué les bonnes souscriptions) et la possibilité d'avoir une fédération de serveurs qui permettent de répartir la charge mais aussi d'éviter les pannes totales lors de l'arrêt d'un équipement sur lequel fonctionne le serveur. La souscription sous forme d'expression régulière sur les données permet aussi aux clients de souscrire aux données qui les intéressent sans avoir la moindre connaissance du système dans son ensemble ou du réseau. Cette implémentation répond donc parfaitement à nos besoins de distribution et de flexibilité.

5.3 Conclusion

Dans ce chapitre, nous avons présenté l'ensemble des équipements et technologies à notre disposition pour la construction de notre prototype. Les équipements et matériels choisis répondent aux contraintes que nous nous sommes fixés aussi bien en termes de coût que d'acceptabilité. Les solutions logicielles choisies nous permettent quant à elles de répondre à nos besoins de flexibilité et de distribution.

Dans le prochain chapitre, nous allons donc voir comment nous avons utilisé l'ensemble de ces technologies et ce que nous avons apporté pour arriver à construire un prototype fonctionnel.

Chapitre 6

Prototype : implémentation et expérimentations

The Game is a mental game where the objective is to avoid thinking about The Game itself. Thinking about The Game constitutes a loss, which, according to the rules of The Game, must be announced each time it occurs. It is impossible to win most versions of The Game ; players can only attempt to avoid losing for as long as they possibly can.
Wikipedia, The Game

1. Everyone in the world is playing The Game.
2. Whenever one thinks about The Game, one loses.
3. Losses must be announced to at least one person.
Rules of The Game

Nous avons vu dans le chapitre précédent les équipements et les technologies existants sur lesquels nous nous reposons pour construire notre prototype. Dans ce chapitre, nous nous intéressons à l'approche retenue pour réaliser ce prototype ainsi qu'aux outils développés pour le déploiement d'un démonstrateur.

Nous commençons par présenter l'implémentation de la théorie des fonctions de croyance que nous avons réalisée et nous en évaluons les performances via une série de tests. Considérant l'exploitation de cette implémentation, nous posons ensuite le problème de la répartition des

calculs entre les différents équipements composant notre prototype. Nous explicitons donc ensuite l'architecture en composants qui a été mise en place et son intérêt en termes de répartition de la charge de calcul, de flexibilité, de résilience mais aussi de mise à l'échelle.

Comme nous l'avons montré dans les chapitres 3 et 4, appliquer la théorie des fonctions de croyance à des mesures de capteurs n'est pas trivial. L'indépendance des sources est un problème majeur si l'on souhaite ne pas accorder trop d'importance à certaines preuves vis-à-vis des autres. Nous exposons donc ce problème et introduisons un nouveau type de composant : les capteurs virtuels. Nous présentons notre contribution qui conduit à la création de ces "capteurs virtuels" puis les principales applications de ces derniers.

Enfin, dans le cadre de la collaboration avec EDF R&D dans le domaine du contrôle énergétique au sein de l'habitat intelligent, nous avons pu déployer un démonstrateur afin d'illustrer certaines possibilités offertes par un habitat intelligent et plus particulièrement lorsque celui-ci repose sur une analyse des contextes et situations en cours. Ces travaux s'intègrent parfaitement aux travaux présentés dans ce document. Ce chapitre se termine par la présentation du démonstrateur que nous avons déployé.

6.1 THE GAME : exploiter les fonctions de croyance dans un contexte embarqué

L'implémentation de la théorie des fonctions de croyance a fait l'objet de nombreux travaux [6, 41, 54, 65]. Malheureusement, les implémentations disponibles se font peu nombreuses. Afin de pouvoir utiliser cette théorie sur des équipements aux capacités limitées, une implémentation optimisée et compatible avec un maximum d'équipements nous est nécessaire. Nous avons donc créé une implémentation appelée THE GAME (*THeory of Evidence in a lanGuage Adapted for Many Embedded systems*). Dans cette section, nous présentons les principales caractéristiques et optimisations de cette implémentation. Nous présentons ensuite quelques tests de performance.

6.1.1 Principales caractéristiques

THE GAME est une implémentation des bases de la théorie des fonctions de croyance écrite en langage C [47]. Ce langage a l'avantage d'être disponible pour un très grand nombre d'architectures et de systèmes d'exploitation. Le système d'exploitation Contiki [31] des nœuds que nous utilisons est ainsi programmé en C. Il permet une optimisation fine de la gestion de la mémoire mais aussi des calculs. Ce langage répondait donc parfaitement à nos contraintes matérielles.

L'implémentation THE GAME est disponible sous licence Apache v2.0 [1] et est donc libre et gratuite. Elle est disponible à l'adresse suivante :

https ://github.com/bpietropaoli/THEGAME

1. http ://www.apache.org/licenses/LICENSE-2.0.html

L'implémentation THE GAME fournit les opérations les plus couramment rencontrées dans les travaux utilisant la théorie des fonctions de croyance. On y retrouve notamment plusieurs règles de combinaison telles que celles de Dempster [21], de Smets [81], de Chen et al. [16], Dubois et Prade [29], de Murphy [61] et enfin de Yager [97]. Sont implémentées aussi des fonctions usuelles telles que l'affaiblissement et le conditionnement [81] ou encore de quoi caractériser les fonctions de masse avec des opérations telles que la spécificité [98], l'auto-conflit [66], le désaccord [90], la distance [45] et bien d'autres.

THE GAME offre bien entendu les outils précédemment décrits chapitre 3 et 4 pour la construction des fonctions de masse à partir de mesures de capteurs. Cette bibliothèque offre également de quoi construire des croyances à partir d'autres croyances en permettant de propager des fonctions de masse d'un cadre de discernement à un autre [PDW13a]. Enfin, les transformations usuelles pour la prise de décision sont elles aussi disponibles. En résumé, THE GAME implémente toutes les opérations nécessaires dans de nombreuses applications de la théorie des fonctions de croyance.

6.1.2 Optimisations

Afin de pouvoir faire fonctionner la bibliothèque THE GAME sur des équipements aux capacités de calcul limitées, nous avons procédé à deux principales optimisations dans l'implémentation des fonctions de croyance.

Représentation binaire des ensembles

La première optimisation concerne la réprésentation des ensembles. L'ensemble de la théorie des fonctions de croyance repose sur la manipulation d'ensembles. Une optimisation commune consiste à utiliser une représentation binaire des ensembles [41].

Nous avons vu qu'il est nécessaire de définir un cadre de discernement Ω contenant l'ensemble des mondes/états possibles. Les fonctions de masse sont ensuites définies sur l'ensemble des sous-ensembles 2^{Ω} de ce cadre de discernement. Chaque élément focal de nos fonctions de masse sera donc un sous-ensemble de Ω. Si notre cadre de discernement contient n états possibles, il est alors possible de représenter chaque sous-ensemble sous forme d'un nombre écrit en binaire à l'aide de n bits. Par exemple, pour le cadre de discernement $\Omega = \{A, B, C, D\}$, chaque sous-ensemble sera représenté par un nombre en binaire à 4 bits. Ainsi, on a :

$$\begin{aligned} \emptyset &= 0000 \\ \{A \cup B\} &= 1100 \\ \{B \cup D\} &= 0101 \\ \Omega &= 1111 \end{aligned}$$

On peut donc faire correspondre chaque sous-ensemble à un nombre (c'est particulièrement intéressant pour communiquer des fonctions de croyance) mais surtout, les opérations de conjonction et disjonction d'ensembles sont largement simplifiées. En effet, elles correspondent respectivement

à un ET logique (usuellement représenté par &) et un OU logique (usuellement représenté par |) bit à bit. Par exemple :

$$\{A \cup B\} \cap \{B \cup D\} = 1100 \ \& \ 0101 = 0100 = \{B\}$$
$$\{A \cup B\} \cup \{B \cup D\} = 1100 \mid 0101 = 1101 = \{A \cup B \cup D\}$$

Cette représentation binaire des ensembles permet donc de gagner un temps considérable sur les opérations de combinaison notamment. Ces opérations étant les plus lourdes, cette optimisation est la bienvenue.

Réduction du nombre d'éléments

La deuxième optimisation effectuée concerne le fait de ne considérer dans les fonctions de masse que les éléments focaux plutôt que l'ensemble des sous-ensembles 2^{Ω}. Cela est possible du fait que les éléments non-focaux induisent toujours des termes nuls dans toutes les opérations, que ce soit les combinaisons ou les caractérisations. Il est donc parfaitement possible de s'en débarrasser.

En procédant ainsi, on réduit drastiquement la complexité des règles de combinaison normalement en $O(2^n)$ (où n est le nombre d'états possibles définis dans le cadre de discernement) en ne considérant que les éléments focaux des fonctions de masse à combiner. Le plus souvent, les fonctions de masse ont un nombre d'éléments focaux beaucoup plus petit que le nombre de sous-ensembles d'états possibles. Il est d'ailleurs possible de réduire le nombre d'éléments focaux via des approximations de fonctions de masse [25].

Grâce à ces deux principales optimisations, les opérations les plus lourdes de la théorie des fonctions de croyance peuvent être effectuées sur des équipements aux capacités de calculs limitées dans des temps raisonnables pour des applications en temps réel comme le nécessitent certains services de l'habitat intelligent.

6.1.3 Performances

Nous devons valider la possibilité de faire fonctionner un raisonnement reposant sur la théorie des fonctions de croyance sur des équipements limités. Nous avons donc exécuté une série tests de performances sur un Raspberry Pi [2] muni d'un processeur ARM1176 700MHz. Nous souhaitions nous assurer qu'il était possible d'effectuer les calculs nécessaires à nos attributs de contexte en un temps raisonnable permettant aux services des couches supérieures de notre architecture de fonctionner.

Nous avons d'abord testé le temps d'exécution de la combinaison de fonctions de masse qui est l'opération la plus lourde. Les deux facteurs importants qui influencent le temps nécessaire à la combinaison de fonctions de masse sont le nombre d'états possibles dans le cadre de discernement (qui donne la taille de la forme binaire des ensembles) et le nombre d'éléments focaux. Les tableaux 6.1, 6.2, 6.3 et 6.4 donnent les temps d'exécution moyens des règles de combinaison de Dempster, de Dubois et Prade [29], la moyenne des fonctions de masse et la règle de combinaison

2. http ://www.raspberrypi.org/

de Murphy [61] en fonction de ces deux paramètres lors de la combinaison de deux fonctions de masse générées aléatoirement. Les règles de Smets et de Yager ont des temps d'exécution comparables à ceux de la règle de Dempster, c'est pourquoi ils ne sont pas donnés ici. Cela peut s'expliquer par leur proximité algorithmique.

États = $\lvert\Omega\rvert$ \ # d'élem. focaux	2	4	8	16	32	64	128
8	19µs	56µs	340µs	1.91ms	7ms	46ms	180ms
32	27µs	132µs	780µs	8.39ms	119ms	1.97s	35.15s
128	81µs	330µs	2.03ms	19.90ms	259ms	4.34s	69.16s
512	225µs	1.07ms	6.82ms	64.62ms	822ms	13.88s	209.3s

Tableau 6.1 – Temps moyens d'exécution de la règle de combinaison de Dempster en fonction du nombre d'états possibles et du nombre d'éléments focaux.

États = $\lvert\Omega\rvert$ \ # d'élem. focaux	2	4	8	16	32	64	128
8	13µs	80µs	390µs	2.23ms	12.0ms	60.0ms	283ms
32	43µs	133µs	840µs	8.33ms	118ms	1.98s	35.0s
128	74µs	320µs	2.03ms	19.82ms	258ms	4.33s	69.06s
512	253µs	1.08ms	6.77ms	64.56ms	823ms	13.87s	209.22s

Tableau 6.2 – Temps moyens d'exécution de la règle de combinaison de Dubois et Prade en fonction du nombre d'états possibles et du nombre d'éléments focaux.

États = $\lvert\Omega\rvert$ \ # d'élem. focaux	2	4	8	16	32	64	128
8	11µs	16µs	60µs	200µs	490µs	2ms	4ms
32	11µs	30µs	80µs	170µs	650µs	2ms	7ms
128	37µs	68µs	140µs	450µs	1.4ms	4ms	17ms
512	95µs	213µs	510µs	1.4ms	4.2ms	16ms	47ms

Tableau 6.3 – Temps moyens d'exécution de la moyenne de fonctions de masse en fonction du nombre d'états possibles et du nombre d'éléments focaux.

On remarque aisément que le temps d'exécution augmente de façon exponentielle avec le nombre d'éléments focaux et ce quelque soit la règle de combinaison utilisée. Les temps d'exécution explosent rapidement pour la règle de combinaison de Murphy. Cette règle effectue une moyenne des fonctions de masse pour ensuite combiner le résultat avec lui-même en utilisant la règle de Dempster. En combinant deux fonctions de masse aléatoires, il est alors possible d'obte-

États = $\lvert\Omega\rvert$ \ # d'élem. focaux	2	4	8	16	32	64	128
8	68µs	297µs	1.4ms	8.0ms	36.4ms	131ms	433ms
32	124µs	601µs	4.8ms	59.6ms	943.9ms	16.4s	62.3s
128	313µs	1.5ms	11.7ms	138.6ms	2.1s	34.1s	156.7s
512	1.1ms	5.2ms	38.8ms	449.3ms	6.6s	108.9s	500s

TABLEAU 6.4 – Temps moyens d'exécution de la règle de combinaison de Murphy en fonction du nombre d'états possibles et du nombre d'éléments focaux.

nir un très grand nombre d'éléments focaux. Cela peut expliquer pourquoi les temps d'exécution ne correspondent pas en moyenne à une simple somme des temps de la règle qui fait la moyenne et de la règle de Dempster. Les temps d'exécution de la règle de combinaison de Dubois et Prade sont relativement proches de ceux de la règle de Dempster du fait de la proximité algorithmique des deux règles.

Les temps d'exécution présentés ne sont pas tous représentatifs du type de problème que nous souhaitons aborder. Ainsi, il ne sera que rarement nécessaire d'avoir plus de quatre ou cinq états possibles dans notre cadre de discernement. De même, les fonctions de masse induites par les capteurs ne comporteront jamais un très grand nombre d'éléments focaux. Cela nécessiterait des modèles très complexes mais ne reflèterait que l'utilisation de preuves faibles incapables d'apporter de réelles informations sur ce qui se produit dans l'environnement. Afin de compléter ces tests de perfomances, nous avons donc évalué les temps d'exécution de cas usuels dans notre habitat intelligent. Le tableau 6.5 résume les temps d'exécution obtenus. Ces derniers correspondent à un processus complet autour des fonctions de croyance pour obtenir à résultat à partir de données de capteurs. Ainsi, ces temps d'exécution comprennent :

- La construction des fonctions de masse à partir de données de capteurs (ici, de fausses données) ;
- La combinaison des fonctions de masse à deux reprises afin de prendre en compte la possibilité d'utiliser une combinaison pour l'analyse du conflit ;
- La caractérisation via divers outils (auto-conflit, désaccord, distance, spécificité et non-spécificité) ;
- Le conditionnement de toutes les fonctions de masse, utile par exemple pour réviser des croyances *a posteriori*) ;
- Le calcul des critères de décision (crédibilité, pari sur la probabilité, plausibilité et commonalité) ;
- La transformation en chaînes de caractères et l'enregistrement dans des fichiers textes de toutes les fonctions de masse, des transformations diverses et des caractérisations (pire cas d'historique, peut être nécessaire au déboggage).

Dans ces résultats, le nombre d'états possibles détermine le nombre moyen d'éléments focaux créés lors de la construction des fonctions de masse à partir des données de capteurs. On observe également une nette augmentation du temps d'exécution avec l'augmentation du nombre

Etats \ Capteurs	5	10	15	20
3	10.2ms	24.3ms	43.0ms	67.5ms
5	11.3ms	26.7ms	47.7ms	75.0ms
7	12.2ms	29.6ms	52.6ms	82.6ms
9	16.5ms	38.3ms	66.6ms	102.2ms

TABLEAU 6.5 – Temps moyen d'exécution de problèmes usuels en fonction du nombre de capteurs utilisés et du nombre d'états possibles.

de capteurs utilisés et le nombre d'états possibles. Les temps d'exécution restent cependant raisonnables, d'autant que nous verrons dans la suite de ce chapitre qu'il est possible de répartir une bonne partie de ces calculs sur plusieurs équipements. De plus, l'utilisation de quinze ou vingt capteurs pour le calcul d'un attribut de contexte semble peu probable compte-tenu de la faible instrumentation visée. De plus, cela induirait certainement trop de conflit et n'améliorait pas la qualité des contextes inférés.

L'ensemble des résultats de performance présentés ici sont disponibles dans le module de tests de la bibliothèque THE GAME. Il est donc possible de les reproduire sur divers équipements.

6.2 Définition d'une architecture modulaire

Dans la section précédente, nous avons vu qu'il est possible, à l'aide de notre implémentation de la théorie des fonctions de croyance THE GAME, de calculer des attributs de contexte en un temps raisonnable sur des équipements tels que les Raspberry Pi. Dans cette section, nous présentons comment nous avons utilisé THE GAME afin d'implémenter un ensemble de composants, chacun effectuant une des tâches présentées au chapitre 3. Nous décrivons aussi les principaux avantages d'une telle approche.

6.2.1 Partage des tâches

Le principal but de l'architecture en composants est de répartir les tâches. Dans notre cas, on peut les décomposer en trois niveaux : construire les croyances, les fusionner et enfin prendre une décision. Il est parfaitement possible d'effectuer l'ensemble de ces tâches à l'aide d'un unique programme. Malheureusement, cette approche ne permet pas de répartir la charge de calcul requise sur un ensemble d'équipements. Dans une maison réelle, le nombre d'attributs de contexte nécessaires à la compréhension de situations différentes peut être très élevé. Il est donc peu probable qu'un seul calculateur soit déployé. En plus du risque de surcharge, ne déployer qu'un seul équipement rend le système particulièrement sensible aux pannes. Nous essayons donc de répartir l'ensemble des calculs à effectuer pour la couche *Perception* sur plusieurs calculateurs.

Dans le chapitre précédent, nous avons proposé l'utilisation du paradigme de communication *publish/subscribe* pour assurer les échanges entre les éléments de notre architecture. Celui-ci va

nous permettre de répartir les tâches facilement sans qu'aucun programme n'ait la moindre idée de la topologie de notre système ni comment les tâches sont réparties.

La figure 6.1 illustre les composants qui ont été implémentés : le *believer*, le *fuser* et le *decision maker* (littéralement le croyant, le fusionneur et le preneur de décision). Dans cette architecture, chaque composant n'effectue qu'une seule tâche. Plusieurs composants de chaque type peuvent être déployés afin de partager la charge de calcul entre divers équipements. Chacun de ces composants s'abonne, par l'intermédiaire d'un client Elvin (*i.e.* un client de l'implémentation du paradigme *publish/subscribe* que nous avons choisie), à toutes les données qui l'intéressent. Par exemple, les *believers* s'abonnent aux données de capteurs dont ils ont besoin pour les attributs de contexte dont ils ont la charge. Il est possible de déployer de nouveaux composants à la volée et donc une partie du système peut-être éteinte/allumée en fonction des besoins de la plateforme. On peut penser par exemple que de nombreux attributs de contexte ne sont pas nécessaires tant qu'une présence n'a pas été détectée. Il est alors possible de réduire la consommation générale du système en mettant en veille une partie de ses fonctionnalités quand elles ne sont pas *a priori* nécessaires.

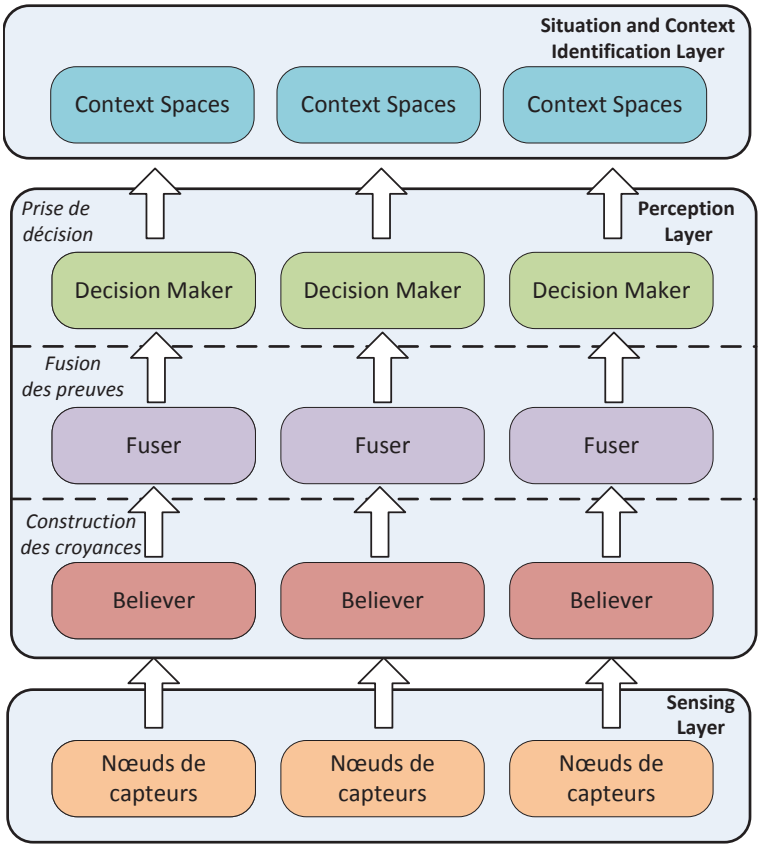

FIGURE 6.1 – Architecture en composants.

Chacun de ces composants peut fonctionner sur un équipement quelconque du moment qu'il est capable de communiquer avec les autres. Pour le moment, il est possible d'avoir plusieurs *belie-*

vers pour un même attribut de contexte. Cela permet donc de réellement répartir la construction des fonctions de masse. Malheureusement, chaque *fuser* récolte l'ensemble des preuves disponibles pour un ou plusieurs attributs de contexte et les fusionne toutes. Les *fuser* servent donc de "puits" de données et rendent le système sensible aux pannes. En procédant ainsi, on ne distribue donc pas la fusion des preuves bien que cela constitue la majeure partie des calculs à effectuer. Il faut donc réfléchir à une manière de répartir cette fusion.

6.2.2 Répartition de la fusion

Afin de répartir la charge de calcul requise pour la fusion des preuves reccueillies, il faut que chaque *fuser* ne s'abonne qu'à une partie des preuves. Il faudrait ensuite plusieurs couches de *fusers* afin de combiner les sous-combinaisons de preuves. Cela est possible lorsque l'on utilise une règle de combinaison à la fois associative et commutative comme c'est le cas avec la règle de combinaison de Dempster. Cela ne serait malheureusement pas possible avec la règle de combinaison de Dubois et Prade puisque cette règle demande de combiner l'ensemble des preuves en une seule fois sous peine de voir le résultat dépendre de l'ordre dans lequel les fonctions de masse sont combinées.

Pour faire cette répartition des *fusers*, il est possible d'utiliser des identifiants uniques composés de nombre premiers ou de facteurs de nombres premiers. Chaque *believer* se voit attribuer un identifiant qui est un nombre premier. Chaque *fuser*, en s'abonnant à une partie des *believers* peut se créer un identifiant unique qui est un produit de nombres entiers. Ensuite, il est possible de rajouter des couches de *fusers* qui s'abonnent aux autres précédemment décrits afin de terminer la combinaison de la totalité des preuves.

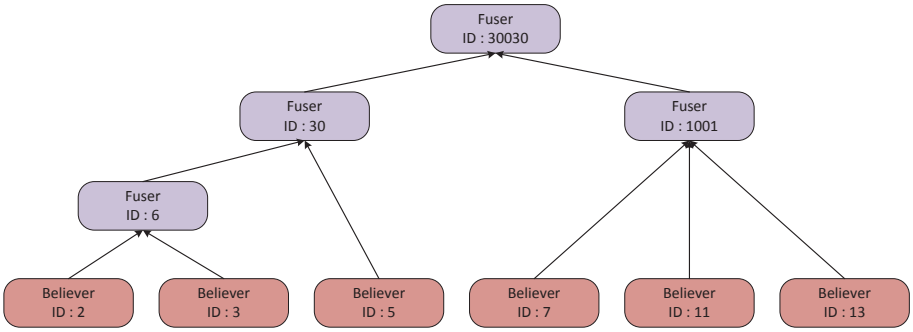

FIGURE 6.2 – Répartition de le combinaison des fonctions de masse.

La figure 6.2 illustre ce principe. Le *fuser* 6 est chargé de fusionner les preuves reccueillies par les *believers* 2 et 3 et ainsi de suite. En procédant ainsi, il est donc possible de répartir ces *fuser* sur divers calculateurs et le paradigme de communication *publish/subscribe* nous permet d'acheminer les données sans que les composants n'aient besoin de connaître la topologie du réseau ou la répartition exacte.

Bien que cette approche puisse paraître adaptée à la répartition de la charge de calcul, elle a ses défauts. Comme cela a déjà été dit, cette répartition ne permet pas l'utilisation de

n'importe quelle règle de combinaison. En effet, l'utilisation d'une règle non communitative et non associative entrainerait une dépendance vis-à-vis de l'ordre dans lequel les combinaisons sont effectuées et donc de la manière dont la fusion est distribuée. Enfin, si l'on a bien réparti notre charge de calcul, un *fuser* particulier doit être désigné comme racine de l'arbre de fusion afin de servir de source de données à le ou les *decision maker(s)* concerné(s). De fait, les données transitent toujours par un goulot d'étranglement. Ce système est donc malheureusement lui aussi sensible aux pannes.

Pour ce qui est de la perte de données, jusqu'à maintenant, nos *fusers* étaient agnostiques vis-à-vis des preuves qu'ils fusionnaient. Ce principe n'est pas violé ici. En effet, la seule différence ici pour les *fusers* sont les données auxquelles ils sont abonnés. Ainsi, si des preuves sont perdues, à cause des communications par exemple, seules les preuves recueillies seront effectivement fusionnées. Avec ce système, on ne perd donc pas en résilience lors de la perte de données.

6.2.3 Ajout de composants

Comme nous avons vu, l'architecture en composants nous permet de déployer plusieurs instances d'un même composant. Cela nous permet notamment de répartir la charge de calcul nécessaire à l'obtention de nos attributs de contexte. Autre avantage d'une grande importance est l'indépendance des composants en termes de développement. En effet, chaque composant n'a besoin de connaître que les données dont il a besoin en entrée. A quoi et à qui servent ses données de sortie n'a aucune importance. Il est donc possible ici, de développer les composants de façon plus ou moins indépendante.

Il est donc possible de rajouter des composants sans avoir à modifier toute l'architecture déjà en place. C'est ce que nous avons fait par exemple avec un nouveau composant permettant de construire des croyances à partir d'autres croyances. Ce composant repose sur le principe de propagation des fonctions de masse d'un cadre de discernement à un autre que nous avons proposé dans [PDW13a]. Cela peut nous permettre de déduire des attributs de contexte à partir d'autres attributs de contexte. La figure 6.3 illustre ce principe. On peut par exemple déduire des croyances sur le fait que quelqu'un est endormi dans une pièce à partir des attributs de contexte "Présence" et "Posture".

Nous avons pu rajouter un composant permettant de propager de fonctions de masse d'un cadre de discernement à un autre comme illustré figure 6.4. Ce composant est vu par les *fusers* de la même manière que les *believers*. Nous n'avons donc modifié aucun autre composant pour rajouter celui-ci. Cette architecture en composants offre donc une bonne flexibilité pour l'ajout de nouvelles fonctionnalités.

6.2.4 Analyse des limites de l'architecture

Le prototype construit ne traite pas de tous les aspects nécessaires au développement d'un habitat intelligent réel. En effet, nous ne nous sommes pas concentrés sur les problématiques liées au conflit par exemple. Il serait alors possible de rajouter un composant permettant d'analyser le conflit pour détecter les éventuels capteurs défaillants et les éteindre si nécessaire. De même, il serait possible d'implémenter un superviseur, déployé en un ou plusieurs exemplaires, capable

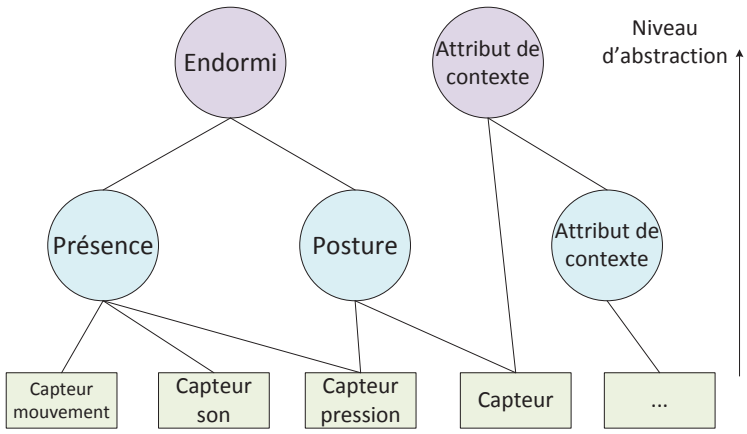

FIGURE 6.3 – Réseau d'attributs de contexte.

de répartir la fusion en fonction des capacités de calcul disponibles sur chaque équipement, des niveaux de batteries, du surcoût en communication, etc. Le superviseur pourrait aussi redéployer les composants à la volée lors d'une panne d'un équipement. L'architecture en composants offre donc de nombreuses possibilités d'extension des services fournies par la couche *Perception*. Cette approche est d'ailleurs généralisée à toutes les couches pour des raisons de mise à l'échelle et de flexibilité.

6.3 Mise en œuvre des capteurs virtuels

L'architecture en composants nous permet de rajouter des composants facilement sans avoir à modifier les composants déjà en place. En mettant en place notre prototype, nous avons observé un ensemble de problèmes liés aux capteurs et à l'application de la théorie des fonctions de croyances aux mesures reccueillis. Pour résoudre ces problèmes, nous avons introduit dans l'architecture des capteurs virtuels. Ces capteurs sont donc un nouveau type de composant. Nous en présentons les principes dans cette section.

6.3.1 Instabilité des modèles

Dans le chapitre 3, nous avons vu que nous construisons un modèle de croyance pour chaque attribut de contexte et chaque capteur pouvant servir de preuve pour cet attribut de contexte (*cf* figure 6.5). Malheureusement, ces modèles ont un défaut majeur : ils dépendent directement des mesures retournées par les capteurs. Nos capteurs peuvent être instables comme nous l'avons déjà vu, mais ils peuvent aussi être bruités et biaisés. Le bruit et le biais peuvent dépendre de nombreux paramètres comme le capteur lui-même, l'environnement dans lequel il est placé, l'électronique du nœud sur lequel il est branché ou bien encore le niveau de batterie du nœud en question. Tous ces paramètres font que d'un déploiement à un autre, les modèles doivent être réajustés afin de prendre en compte le niveau de bruit moyen et les biais des divers capteurs

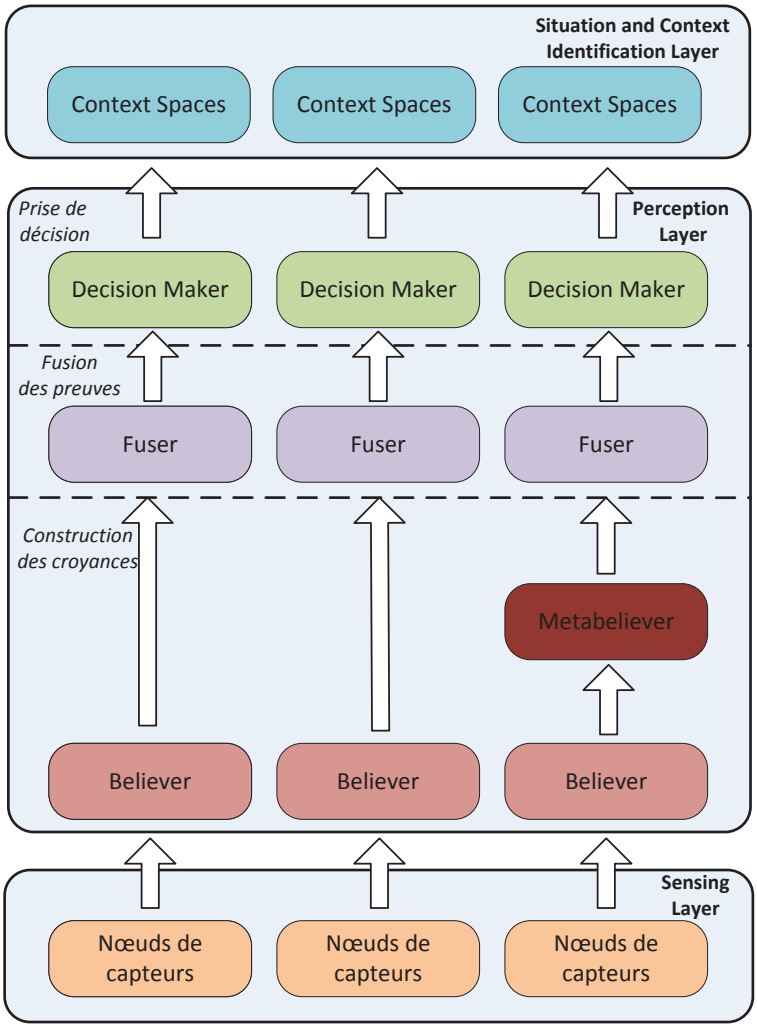

FIGURE 6.4 – Architecture en composants modifiée. Le *metabeliever* est vu exactement de la même manière que n'importe quel *believer* par les *fusers*.

placés dans l'habitat.

Pour résoudre ce problème, nous avons voulu rajouter un composant entre les nœuds de capteurs et les modèles de croyance gérés par les *believers*. Nous avons donc créé un composant permettant de corriger les valeurs renvoyées par nos capteurs. Nous avons donc d'abord construit des capteurs virtuels n'étant que des versions stables des capteurs réels. Notre architecture est alors modifiée comme illustré figure 6.6.

L'utilisation de capteurs virtuels permet de faciliter le déploiement en permettant de réexploiter des modèles de croyance déjà existants. De même, il est possible de standardiser le comportement de capteurs de différents fabricants par l'intermédiaire d'un capteur virtuel. Les capteurs virtuels peuvent aussi servir à la fabrication de méta-données à partir des mesures comme par exemple la variation d'une mesure, la moyenne sur un certain temps, etc. Les utilisations des

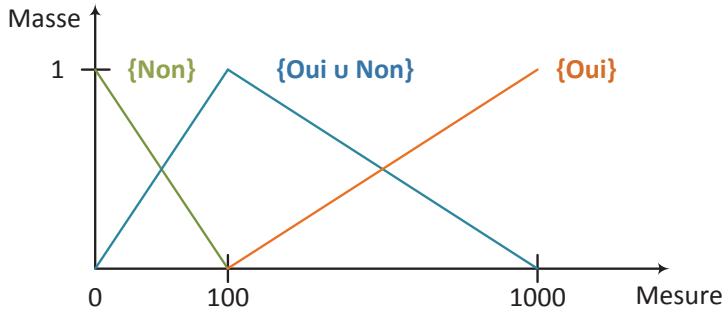

FIGURE 6.5 – Ensemble de fonctions de masse pour un capteur de mouvement utilisé dans la détection de présence.

capteurs virtuels peuvent être multiples et c'est le sujet de la suite de cette section.

Ce nouveau composant, comme les précédents, n'a demandé aucune modification de l'existant. Il s'abonne aux capteurs dont il a besoin et est vu comme un nœud de capteurs réels. Une fois de plus, rien n'a été changé dans les composants exploitant les données créées par ce nouveau composant. Chaque instance est personnalisée à l'aide d'un code à réimplémenter à chaque déploiement. Si cela peut paraître lourd en terme d'ingénierie, cela l'est toujours moins que de devoir modifier des modèles complexes. Cela évite également d'avoir à réexpérimenter la pertinence des modèles de croyance utilisés. Ces derniers peuvent donc être testés dans divers environnement et redéployés tels quels dans n'importe quel environnement. Enfin, n'a besoin d'être recodée que la partie qui permet de transformer l'entrée en sortie. Ainsi, toute la partie permettant aux capteurs virtuels de s'abonner aux capteurs et d'envoyer leurs résultats restent constantes d'un déploiement à un autre. On ne développera donc que de petites fonctions rapides de stabilisation de données brutes.

6.3.2 Indépendance des capteurs

Dans les chapitres précédents, nous avons vu que la règle de combinaison de Dempster n'est pas idempotente (*i.e.* $m \oplus m \neq m$). Cette propriété suggère que les preuves combinées doivent provenir de sources indépendantes. Par exemple, si l'on cherche à obtenir l'attribut de contexte "Présence" dans une pièce et que l'on dispose de trois capteurs de mouvement et d'un capteur de son, les trois capteurs de mouvement ne doivent pas être considérés comme des sources indépendantes. En effet, ces derniers peuvent mesurer le même évènement physique et leurs mesures ne représentent pas trois preuves distinctes mais une seule et même preuve.

Ce problème d'indépendance peut être résolu en construisant des macro-capteurs virtuels. Ainsi, si l'on dispose de plusieurs capteurs de mouvements dans une pièce, leurs champs de vision peuvent se superposer (*cf* figure 6.7) et il vaut donc mieux considérer les deux capteurs comme n'en étant qu'un seul. Il n'est en effet pas possible de savoir si les deux capteurs n'ont pas observé un seul et même évènement physique.

En disposant de deux capteurs de mouvement, on peut alors décrire le comportement du

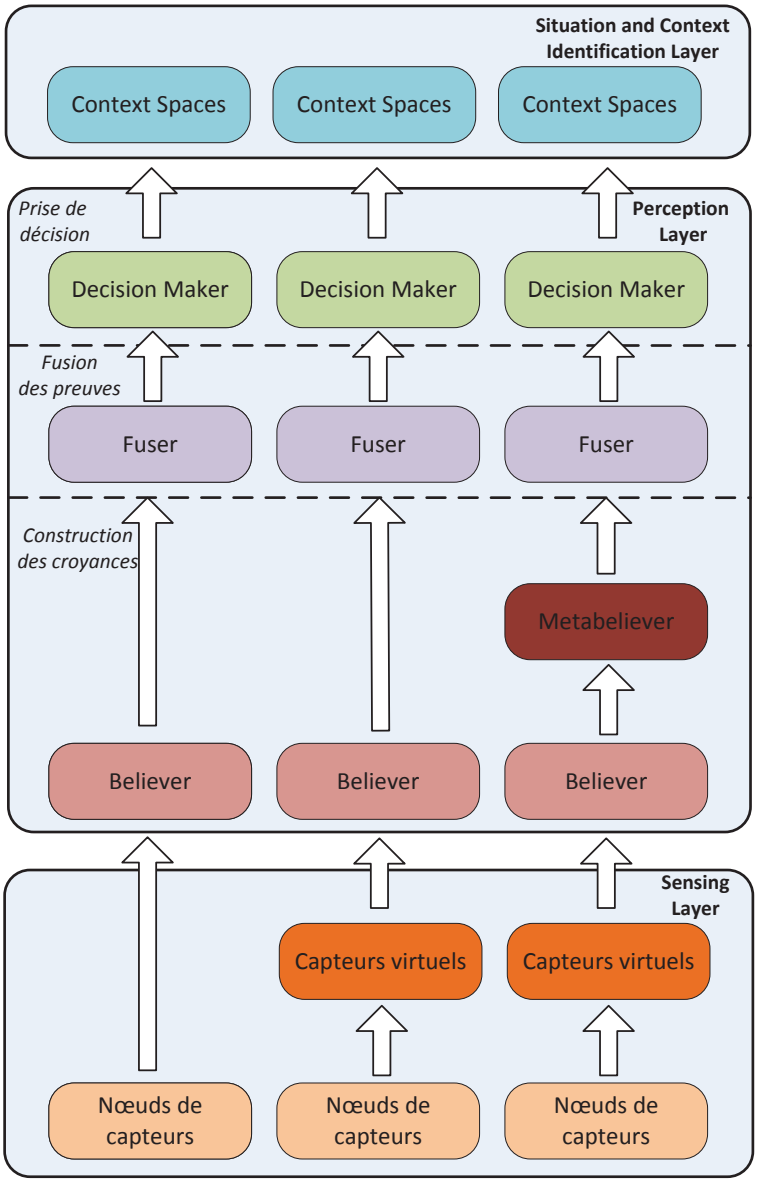

FIGURE 6.6 – Architecture en composants modifiée. Des capteurs virtuels permettent de stabiliser les mesures des capteurs réels.

macro-capteur de mouvement les regroupant comme suit :

- Si les deux mesures sont reçues alors le plus grand mouvement détecté est retourné ;
- Si une seule des deux mesures est reçue, la mesure est renvoyée si elle correspond à un minimum de mouvement observé (on ne peut pas savoir si l'autre capteur n'a pas observé un mouvement), aucune mesure n'est retournée sinon ;
- Si aucune mesure n'est reçue, aucune mesure n'est retournée.

Il faut donc développer un petit algorithme pour chaque macro-capteur que l'on souhaite im-

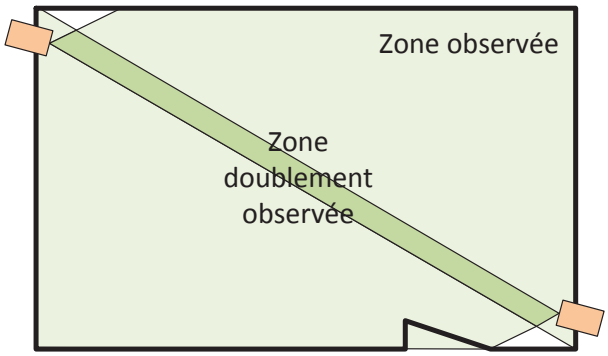

FIGURE 6.7 – Deux capteurs de mouvement dans une pièce. Leurs champs de vision se superposent.

plémenter. Dans notre travaux, nous nous limitons à des macro-capteurs relativement simples dont le comportement est rapide à implémenter. C'est grâce à de simples macro-capteurs que nous obtenons les capteurs de mouvement et les capteurs du nombre de personnes assises dans les résultats décrits chapitre 3 et 4.

6.3.3 Rétroaction

Dans tout automatisme, il est nécessaire de connaître l'impact du système sur ce qu'il mesure. Dans notre habitat intelligent, cela peut se traduire avec un exemple tout simple : l'éclairage automatique et le capteur de luminosité. L'éclairage automatique repose en partie sur un capteur de luminosité. Lorsque la luminosité est faible le système allume la lumière. En allumant la lumière, on augmente la luminosité. Elle devient donc plus importante. Le système décide alors d'éteindre la lumière. La luminosité devient alors trop faible et il rallume la lumière et ainsi de suite.

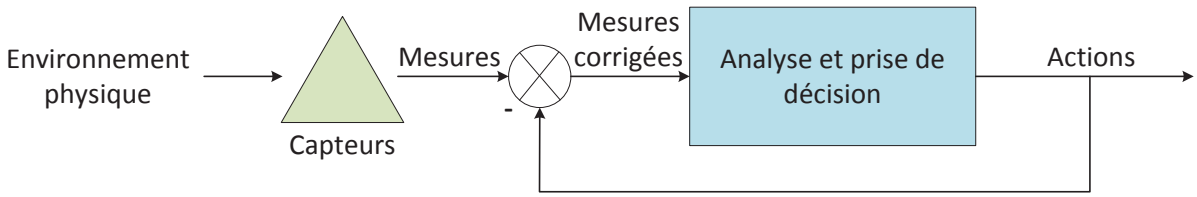

FIGURE 6.8 – Principe de fonctionnement de la rétroaction. Les mesures retournées par les capteurs sont corrigées afin de prendre en compte l'effet des actions du système.

Pour éviter ce genre de boucles infinies, il suffit de prendre en compte une rétro-action (*cf.* figure 6.8). Avec l'aide de capteurs virtuels, il est alors possible de prendre en compte l'impact du système dans les mesures retournées par nos capteurs. Ainsi, dans le cas de l'éclairage automatique, on considèrera le fait que le système a allumé ou non la lumière.

Si l'impact exact des actions du système sur les mesures de nos capteurs n'est pas connu, il

peut au moins être estimé. On évitera au moins les comportements inadaptés dues aux asservissements en boucle ouverte.

6.3.4 Capteurs contextuels

Jusqu'à maintenant, nous nous sommes limités à des cas simples d'utilisation des capteurs virtuels avec la stabilisation de données brutes, la construction de méta-données et la construction de macro-capteurs. Cependant, certains capteurs sont plus complexes que d'autres à utiliser. C'est le cas des capteurs de niveau de CO_2. La figure 6.9 illustre les mesures de niveau de CO_2 enregistrées sur une journée. Si les tendances sont faciles à observer à l'oeil nu sur l'ensemble d'une journée, l'exploitation en temps réel de ces mesures n'est pas triviale.

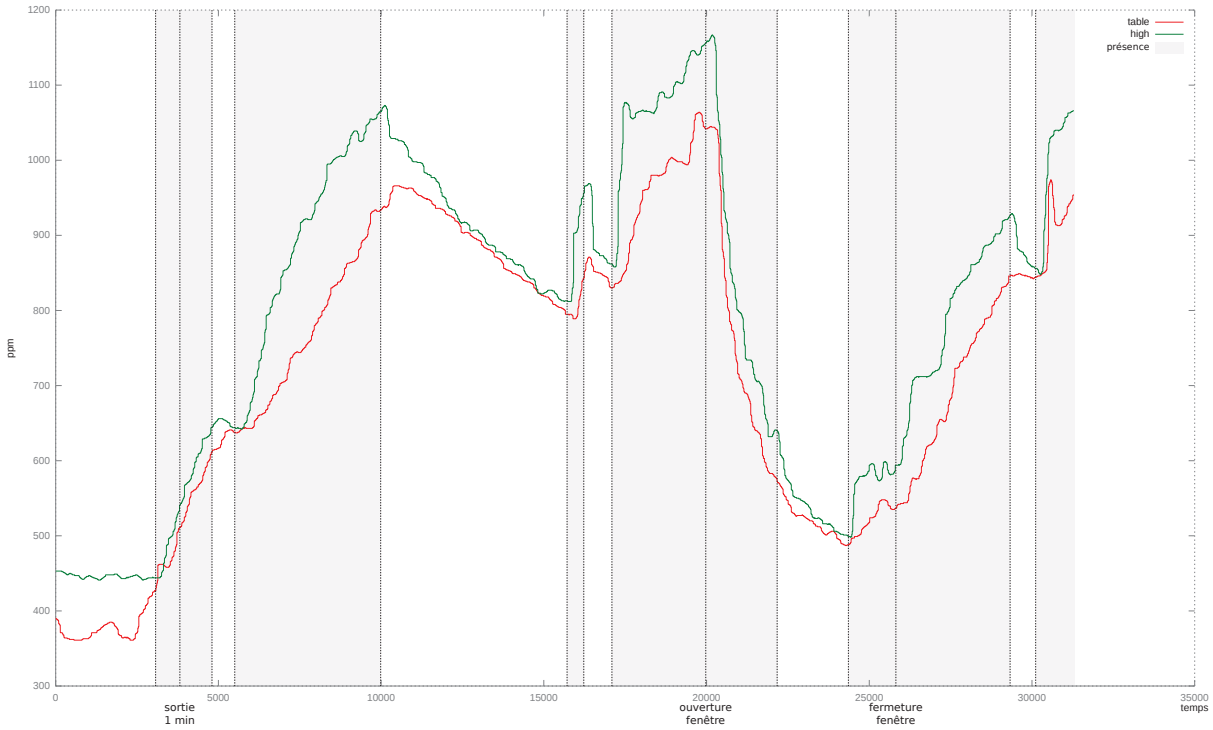

FIGURE 6.9 – Mesures du niveau de CO_2 dans un bureau au cours de la journée. La courbe rouge (au-dessous) correspond aux mesures d'un capteur placé sur une table au centre de la pièce et la courbe verte (au-dessus) celle d'un capteur placé en hauteur sur une armoire. Les intervals grisés correspondent à la présence d'une personne.

Premier fait intéressant, l'inertie du niveau de CO_2 dans la pièce est importante. En effet, lorsque la personne quitte la pièce, le niveau de CO_2 continue de monter pendant un certain temps et ne diminue donc pas immédiatement. Un autre fait important est la sensibilité à l'aération de la pièce. L'ouverture de la porte ou de la fenêtre peut fortement gêner l'interprétation de la mesure de CO_2. Il est donc nécessaire pour pouvoir exploiter ces mesures de connaître le contexte dans lequel on doit les interpréter. Ainsi, on pourra s'aider ici de contacteurs sur les fenêtres et

les portes mais aussi d'un capteur de passage pour déterminer si une personne est entrée ou sortie de la pièce.

En créant un capteur virtuel contextuel, qui exploite donc des capteurs sur les portes et les fenêtres, nous avons réussi à construire des croyances reposant sur les capteurs de niveau de CO_2 pour la détection de présence. Les résultats sont donnés figure 6.10. Le capteur de CO_2 nous permet de stabiliser la détection de la présence sur de plus longues périodes. La croyance peut-être instable avec les entrées/sorties des personnes dans la pièce, générant ainsi les "trous" que l'on observe dans la contribution du niveau de CO_2 dans la détection de la présence. En effet, pour déduire une croyance sur la présence, on observe la tendance du niveau de CO_2 sur un certain temps. Ce temps est remis à zéro à chaque passage dans la porte puisque cela peut signifier aussi bien une entrée qu'une sortie.

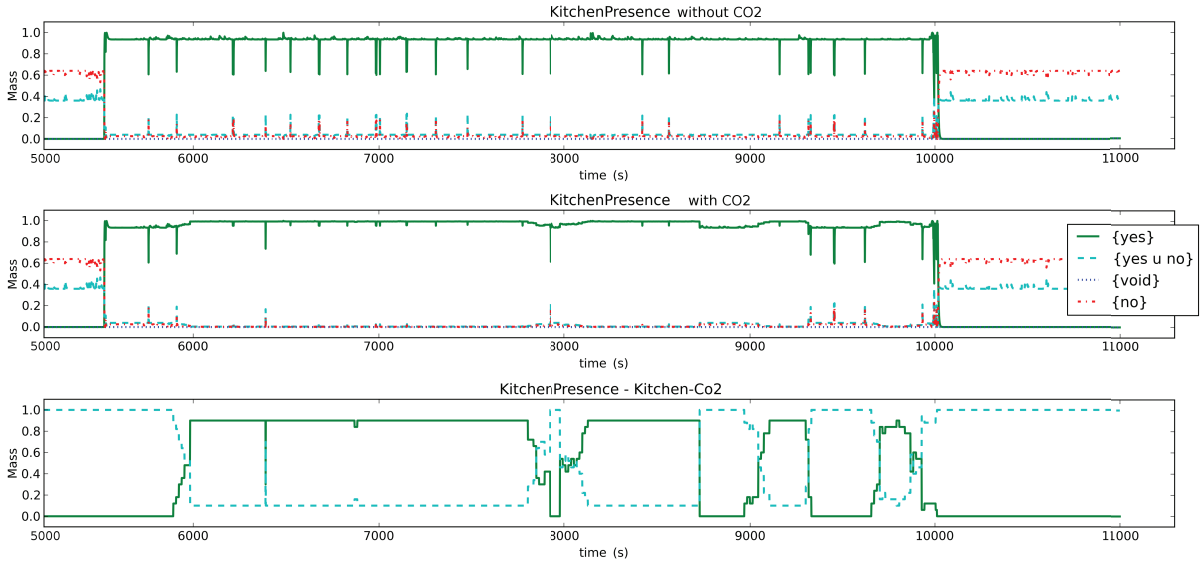

FIGURE 6.10 – Croyances à propos de la présence dans une pièce avec et sans CO_2. En haut, la croyance sans CO_2, au centre, avec CO_2 et en bas la croyance directement induite par le niveau de CO_2.

Les capteurs virtuels ont donc de nombreux usages dans notre prototype que ce soit pour stabiliser nos modèles de croyance, créer de nouvelles données, prendre en compte l'action du système sur les mesures ou encore exploiter de nouveaux capteurs à l'aide du contexte dans lequel ils doivent être interprétés. Chaque instance ne demande de réimplémenter qu'une petite fonction de transformation des données d'entrées en données de sorties. Nous avons donc grandement gagné en temps de déploiement et en simplicité d'utilisation de notre plateforme tout en rajoutant de nouvelles possibilités.

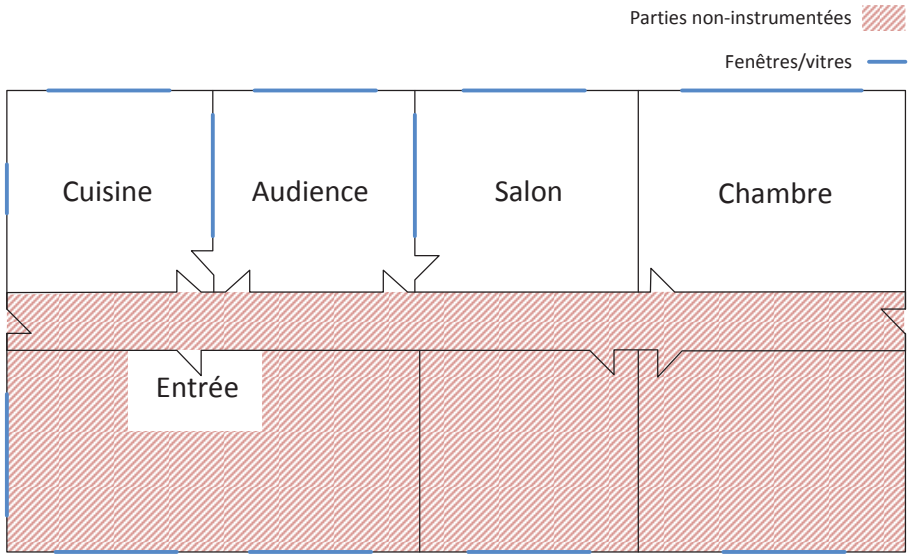

FIGURE 6.11 – Plan des bureaux utilisés pour le dernier démonstrateur.

6.4 Démonstration du prototype

Nous avons eu l'occasion, à trois reprises, de présenter l'état d'avancement de nos travaux à travers un prototype déployé dans plusieurs pièces. Dans cette section, nous présentons le dernier démonstrateur que nous avons déployé.

6.4.1 Installation

Nous avons déployé notre dernier démonstrateur dans un ensemble de bureaux. Le but était de pouvoir montrer à un groupe de personnes, sur un temps assez court, toutes les possibilités de notre plateforme. Le plan des bureaux utilisés est donné figure 6.11.

Nous avons déployé les capteurs suivants :

- Capteurs de mouvement dans toutes les pièces instrumentées ;
- Capteurs de pression sur les chaises dans la cuisine et le salon ;
- Capteurs de vibration sur les chaises de la cuisine ;
- Capteurs de passage dans les portes ;
- Capteurs de contact sur les fenêtres et les portes.

Tous ces capteurs étaient connectés à 12 nœuds communicants. Un interrupteur sans fil EnOcean [3] a aussi été utilisé pour simuler la sonnette de la maison. L'ensemble des calculs nécessaires à la plateforme toute entière étaient répartis entre deux Raspberry Pi [4] et une PandaBoard [5]. Nous avons aussi déployé les équipements augmentés simulés présentés au chapitre précédent. Nous disposions également de smartphones et de tablettes permettant de montrer l'évolution

3. http ://www.enocean.com/en/home/
4. http ://www.raspberrypi.org/
5. http ://pandaboard.org/

FIGURE 6.12 – Capture d'écran d'une interface permettant de suivre ce qui se passe dans la maison.

des contextes en temps réel pendant l'exécution des scénarii. La figure 6.12 montre une capture d'écran d'une des applications permettant de suivre ce qui se passe dans la maison. Ces interfaces permettaient notamment de contrôler quelques équipements augmentés simulés.

6.4.2 Contextes et scénarii

Afin de montrer les possibilités de notre prototype, nous avons joué quelques scénarii types de la vie courante. Les scénarii sont les suivants :

- Le réveil (sur un téléphone Android) va bientôt sonner ;
- La famille se lève et prends son petit déjeuner, certains vont regarder la télévision dans le salon ;
- La famille quitte la maison pour le travail et l'école ;
- Certains membres de la famille rentrent (détecté en avance avec le smartphone) ;
- Quelqu'un prépare le repas ;
- La famille dîne ;
- Le dîner est terminé.

Aux plus bas niveau d'abstraction, nous nous sommes donc intéressés principalement à la présence dans l'ensemble des pièces. En effet, cet attribut de contexte est essentiel pour la détection de la plupart des situations. Nous étions aussi intéressés par l'état des divers équipements et des fenêtres (pour gérer les oublis de fenêtres ouvertes). A plus haut niveau, les situations repas et absence totale (personne à la maison) étaient les clés pour ces scénarii.

6.4.3 Services déployés

Grâce à ce démonstrateur, nous avons pu déployer toute une panoplie de services. Les travaux présentés dans ce document sont centrés sur les couches basses de notre architecture, le but ici

n'est donc pas de faire une liste exhaustive de tous les services qu'il est possible de déployer dans un habitat intelligent réel. Nous décrivons donc uniquement les grandes familles de services qui ont été testés pour illustrer simplement les capacités du démonstrateur.

Eclairage automatique

Au début de ce document, nous décrivons les difficultés de mettre en place un service d'éclairage automatique qui soit effectif. Dans le démonstrateur, nous avons mis en place un service d'éclairage automatique en utilisant à la fois des données de capteurs et des abstractions. Ainsi, notre service d'éclairage automatique repose sur le passage devant la porte pour l'allumage rapide à l'entrée d'une personne dans la pièce mais utilise la présence pour savoir s'il doit s'éteindre ou pas. De même, une présence dans la pièce, y compris sans passage dans la porte, entraîne automatiquement l'allumage de la lumière.

Nous n'avons malheureusement pas pu expérimenter plus en profondeur ce service avec un plus grand nombre de situations et de degrés de complexité avec par exemple des éclairages différents, etc. Ces expérimentations feront l'objet de travaux futurs.

Chauffage automatique

Un des objectifs du démonstrateur était de montrer le concept d'occupabilité des pièces. Ce concept simple permet de définir des pièces occupables (qui seront peut-être occupées dans un futur proche) en fonction des pièces occupées. Typiquement, les pièces adjacentes à une pièce occupée sont occupables. Notre service de chauffage automatique permet trois niveaux de chauffage : économique, confort réduit et confort. Ce niveau de chauffage peut être différent dans chaque pièce. Nous avons donc une gestion multi-zone du chauffage.

Ce service repose lui aussi sur différents niveaux d'abstraction. Tout d'abord une présence prolongée déclenchait l'allumage du chauffage en position confort. Le signal tarifaire quant à lui permettait de définir le niveau de chauffage requis pour les pièces occupables : confort réduit en cas d'heure creuse et économique en cas d'heure pleine. La situation repas permettait aussi de définir qu'une pièce n'était pas réellement occupable malgré la présence de personnes dans la pièce proche. Ainsi, lorsque la famille dîne, le chauffage est en confort dans la cuisine mais en économique dans le salon.

Le signal tarifaire peut permettre aussi de jouer sur les temps de réaction du chauffage automatique. Si une présence prolongée doit déclencher l'allumage du chauffage, la notion de "présence prolongée" peut être modifiée en fonction du signal tarifaire.

Oublis

Il peut arriver d'oublier d'éteindre des équipements ou encore de fermer des fenêtres. La détection d'absence et d'absence totale permettait de signaler les oublis. Si la famille quitte la maison en laissant une ou plusieurs fenêtres ouvertes, le système pouvait le signaler par l'intermédiaire du smartphone avant que la famille ne soit trop éloignée de la maison.

De même, la situation repas suggère qu'il n'est pas nécessaire d'avoir du chauffage ou de l'éclairage dans les pièces inoccupées. Il est donc possible de gérer les oublis avec des données contextuelles.

Equipements intelligents

Certains équipements peuvent consommer beaucoup d'énergie inutilement. C'est le cas par exemple d'un téléviseur qui garde son écran allumé lorsque plus personne ne se trouve dans la pièce. Ainsi, à l'aide d'un téléviseur augmenté simulé sur une tablette, nous avons pu illustrer un téléviseur intelligent capable de couper l'écran lorsque plus personne ne peut le voir *a priori*. Le son est lui aussi coupé lorsque les pièces proches du salon ne sont plus occupées non plus.

Certains équipements électroménagers consomment aussi beaucoup d'énergie. Il est donc préférable de les utiliser pendant les heures creuses. La simulation du signal tarifaire nous a permis d'illustrer les possibilités pour certains électroménagers comme le lave-vaisselle ou la machine à laver d'attendre une heure creuse pour se lancer.

Interface contextuelle

Les interfaces contextuelles sont des interfaces qui s'adaptent en fonction des situations en cours. Nous avons pu illustrer cela avec une interface contextuelle sur un smartphone lorsqu'un appel est reçu alors qu'un appareil peut gêner cet appel. La figure 6.13 montre une capture d'écran d'un smartphone Android présentant une interface modifiée lorsque une radio a proximité est allumée. Ainsi, l'interface propose de prendre l'appel ou de raccrocher mais aussi de prendre l'appel tout en éteignant la radio pour ne pas être dérangé.

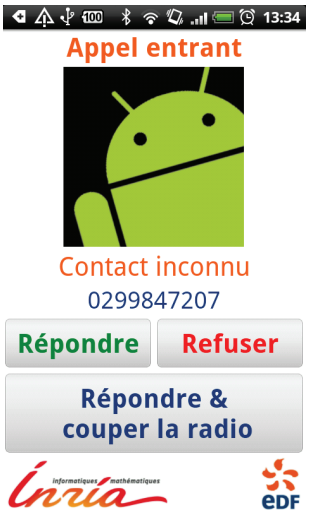

FIGURE 6.13 – Interface Android contextuelle lors d'un appel entrant.

Avec ce démonstrateur, nous avons donc pu démontrer qu'un très grand nombre de services peuvent profiter d'informations contextuelles de divers niveaux d'abstraction. Nous n'avons

malheureusement pas eu le temps de déployer plus de services ni d'exploiter plus de données contextuelles différentes. Cependant, la variété de services déployés prouve la valeur d'un habitat intelligent sensible au contexte et le réel intérêt des différents niveaux d'abstraction.

6.5 Conclusion

Dans ce chapitre, nous avons décrit notre implémentation de la théorie des fonctions de croyance THE GAME. Nous avons présenté les résultats des tests de performance réalisés sur un équipement aux capacités de calcul limitées, un Raspberry Pi. Les temps d'exécution étant raisonnables et suffisamment courts pour des applications temps réel, une répartition des calculs directement sur les nœuds de capteurs est envisageable.

Nous avons aussi présenté notre architecture en composants exploitant la bibliothèque THE GAME. Cette architecture permet un partage des tâches entre différents programmes pouvant fonctionner sur des équipements séparés. Le paradigme de communication *publish/subscribe* et ce type d'architecture créent une excellente synergie et nous donnent la possibilité d'étendre les fonctionnalités du système avec l'ajout de nouveaux composants sans avoir à modifier l'existant. Les composants peuvent être répartis sur l'ensemble des calculateurs et déployés à la volée offrant une grande flexibilité et une certaine résilience à notre système.

Nous avons mis au point un système de reconnaissance de contexte de bas niveau stable et performant, tout cela en respectant les contraintes technologiques et techniques que avons fixées dans les chapitre 1 et 2. En intégrant nos travaux dans un démonstrateur complet, nous avons pu montrer la pertinence de notre approche à plusieurs niveaux d'abstraction. De grandes familles de services ont ainsi pu être mises en place sur la base de quelques contextes et situations élémentaires.

Ce démonstrateur, en tant que preuve de concept, illustre les capacités d'un habitat intelligent sensible au contexte et l'importance de la reconnaissance de contexte à différents niveaux d'abstraction. Un travail de recherche supplémentaire sur l'identification de nouveaux contextes et de nouvelles situations couplé à un travail d'ingénierie donnera certainement naissance à un habitat intelligent peu coûteux, aux services pertinents, acceptables et non frustrants pour ses habitants.

Conclusion et discussion

L'habitat intelligent est un sujet de recherche de grand intérêt aussi bien pour l'assistance aux personnes âgées ou handicapées que pour l'économie d'énergie et le confort de façon générale. L'utilisation de l'informatique ubiquitaire pour créer un habitat intelligent capable de s'adapter dynamiquement aux activités de ses habitants est une approche pertinente. Cette adaptation passe par de la reconnaissance de contexte qui peut être complexe et difficile à mettre en place avec une instrumentation limitée en capacités d'observation et de calcul.

L'objectif de cette thèse est de démontrer qu'une telle approche est possible. Pour cela, nous nous sommes intéressés aux difficultés de mettre en place un service aussi simple que l'éclairage automatique. La description de quelques scénarii de la vie courante nous a suggéré que plusieurs niveaux d'abstraction de données contextuelles sont nécessaires. Ces abstractions peuvent être obtenues via le déploiement de capteurs dans l'habitat mais aussi et surtout via la mise en place de méthodes de fusion de données. Nous avons décidé d'utiliser plusieurs théories et méthodes à différents niveaux afin d'exploiter au mieux les avantages de chacune d'entre elles. Le prototype que nous avons construit nous a ensuite permis de valider notre approche et sa pertinence avec le déploiement d'un grand nombre de services.

Contributions

Cette thèse a été à l'origine d'un certain nombre de propositions et d'implémentations. Ainsi, nous avons atteint les objectifs regroupant la validation d'une approche et le respect de contraintes technologiques et techniques. Nous résumons ici nos contributions.

THE GAME

Pour notre premier niveau d'abstraction, nous avons décidé d'utiliser la théorie des fonctions de croyance. En plus de valider la possibilité d'exploiter facilement des mesures de capteurs pour le calcul d'attributs de contexte simples, nous avons proposé une implémentation des bases de la théorie. Cette implémentation, appelée *THeory of Evidence in a lanGuage Adapted for Many Embedded systems* (THE GAME), utilise des optimisations classiques mais efficaces et permet la fusion des preuves données par nos capteurs dans des temps raisonnables, y compris sur des équipements aux capacités de calcul limitées tels que les Raspberry Pi.

THE GAME étant libre et gratuite, nous souhaitons que son développement se poursuive. De nouvelles règles de combinaison ou encore des méthodes de construction des fonctions de masse

peuvent être implémentées.

Temporisation des croyances

L'application de la théorie des fonctions de croyance à des données de capteurs nous a permis de mettre en évidence un ensemble de difficultés : l'asynchronicité des preuves, l'instabilité des mesures et la nécessité d'accumuler des preuves dans le temps. Pour résoudre ces problèmes, nous avons proposé deux algorithmes de temporisation des fonctions de masse.

Le comportement des deux algorithmes proposés a été décrit en détail. Une mise à l'épreuve de nos algorithmes a permis de valider leur pertinence ainsi que leur efficacité en terme compensation des pertes de données.

Architecture en composants

Si l'architecture en composants n'est pas une idée nouvelle, elle est particulièrement utile dans notre prototype. Elle permet en effet de répartir la charge de calcul de façon simple entre différents équipements. De plus, elle permet l'ajout de nouvelles fonctionnalités sans avoir à modifier l'existant. Elle offre à notre système flexibilité, résilience et possibilité de mise à l'échelle.

L'architecture en composants reposant sur le paradigme de communication *publish/subscribe* est la clé de voûte de notre couche *Perception*. C'est en effet cet intergiciel qui permet de répondre à une grande partie de nos besoins pour le développement d'un habitat intelligent réel.

Capteurs virtuels

Le déploiement de capteurs et de modèles de croyance dans divers environnements nous a amené au problème de la stabilité et donc de la réutilisabilité des modèles. Pour pallier cette difficulté, nous avons mis au point un nouveau composant dans notre architecture : le capteur virtuel.

Ses utilisations sont diverses. Il sert d'abord à stabiliser le comportement des capteurs afin de stabiliser nos modèles de croyance. Il peut aussi servir à produire de nouvelles données, en quelques sortes de nouveaux capteurs. Nous avons vu qu'il nous permet aussi de prendre en compte l'impact du système sur les mesures en insérant une boucle de rétroaction. Enfin, il nous permet d'exploiter de nouveaux types de capteurs en tenant compte du contexte dans lequel ceux-ci sont déployés.

Prototype

Afin de valider notre approche, nous avons développé un prototype complet composé de l'ensemble des couches présentées. Nous avons ainsi pu valider la pertinence des différents niveaux d'abtraction de données contextuelles pour le déploiement de service dans l'habitat intelligent. Ce prototype a été construit en respectant autant que faire se peut les contraintes technologiques nécessaires au coût et à l'acceptabilité, la flexibilité et la mise à l'échelle de l'habitat intelligent.

Le démonstrateur a montré la possibilité de déployer un grand nombres de services pertinents et efficaces avec une instrumentation limitée. Nous avons aussi pu tester nos algorithmes en conditions réelles et prouver leur efficacité.

Discussions

Dans ce document, nous décrivons de nombreux choix que nous avons faits, aussi bien au niveau de l'architecture que des théories utilisées et des technologies déployées. Certains peuvent être discutés et c'est donc pour cela que nous souhaitons dans cette conclusion justifier de manière approfondie certains de nos choix.

Des théories utilisées

Afin d'adapter l'architecture en couches suggérée par Coutaz *et al.*, nous avons décidé d'utiliser plusieurs théories et méthodes de fusion. Comme expliqué au chapitre 2, nous avons choisi les théories qui nous semblaient les plus pertinentes en fonction du type d'abstraction que nous cherchions à obtenir. Ainsi, à bas niveau, pour obtenir des attributs de contexte, nous avons décidé d'utiliser la théorie des fonctions de croyance. Cette théorie a l'avantage d'être agnostique du matériel. Ainsi, il est possible, une fois les données brutes transformées en croyance, de fusionner n'importe quelle preuve avec les autres quelque soit sa provenance. L'indépendance vis-à-vis du matériel nous semblait importante pour faciliter les déploiements et surtout permettre une extension et une flexibilité du système.

A plus haut niveau, la théorie des *Context Spaces* offre des outils de modélisation et de raisonnement intelligibles et surtout adaptés à la fusion de données abstraites. Les besoins en modélisation se font de plus en plus importants au fur et à mesure que l'on monte en abstraction car les situations deviennent de plus en plus complexes. La reconnaissance de plan avec PHATT nous semblait correspondre à nos besoins de prédictions. L'utilisation de cette méthode à haut niveau présente l'avantage de réduire nettement la complexité de l'algorithme en offrant en données d'entrée des situations plutôt que des mesures de capteurs. De plus, la prédiction de situations et d'activités nous semble plus pertinente que la prédiction de petites actions détaillées qui ne seraient pas forcément exploitables par des services.

Bien que les théories et méthodes que nous avons intégrées dans notre prototype nous semblent pertinentes, il n'est pas exclu d'utiliser d'autres théories et méthodes si cela s'avère nécessaire. De plus, la limite entre les différentes couches n'est pas parfaitement définie. Ainsi, il serait parfaitement possible de n'utiliser que la théorie des fonctions de croyance avec des réseaux crédibilistes (dans lesquels des croyances sur des contextes peuvent être déduites de croyances sur d'autres contextes [PDW13a]) pour calculer à la fois les attributs de contexte mais aussi les situations et éventuellement les plans. De même, la théorie des *Context Spaces* offre des outils pour déduire des situations directement à partir de mesures de capteurs. Il est aussi possible de déduire des sous-situations qui pourrait correspondre à nos attributs de contexte. Nous avons donc essayé de définir les limites des couches en termes de temps et d'espace. Ainsi, la couche "Perception" s'applique plutôt localement à l'échelle d'une pièce et pour une période allant de

quelques secondes à quelques minutes. Quant à la couche supérieure, elle couvre l'ensemble de l'habitat sur des échelles de quelques minutes à quelques heures.

L'architecture déployée n'est donc pas figée et pourra évoluer au cours des déploiements et des besoins.

De la règle de combinaison de Dempster

La théorie des fonctions de croyance n'est pas sans polémique. De nombreuses questions se posent encore sur l'interprétation que l'on doit faire des fonctions de masse et de la règle de combinaison de Dempster. Cela a donné et donne toujours lieu à de longues discussions sur la validité de cette règle de combinaison et plus généralement de la théorie [26, 40, 91, 102].

Les exemples de Zadeh [102] et Dezert *et al.* [26], donnés respectivement tableau 6.6 et 6.7, remettent en cause la validité de la règle de combinaison de Dempster en explicitant des cas dits contre-intuitifs. Il faut donc se poser la question de l'interprétation à la fois des fonctions de masse et de la règle de combinaison. Les deux exemples sont donnés pour un cadre de discernement générique $\Omega = \{A, B, C\}$.

m	A	B	C	$\emptyset$
m_1	0.99	0	0.01	0
m_2	0	0.99	0.01	0
$m_1 \textcircled{\cap} m_2$	0	0	0.0001	0.9999
$m_1 \oplus m_2$	0	0	1	0

TABLEAU 6.6 – Exemple de Zadeh : combinaison de deux fonctions de masse fortement en conflit à l'aide de la règle de combinaison de Dempster.

m	A	$A \cup B$	C	$A \cup B \cup C$
m_1	a	$1-a$	0	0
m_2	0	b_1	$1-b_1-b_2$	b_2
$m_1 \oplus m_2$	a	$1-a$	0	0

TABLEAU 6.7 – Exemple de Dezert *et al.* : m_2 est élément neutre de la règle de Dempster pour m_1.

L'exemple de Zadeh peut paraître contre-intuitif car le résultat de la combinaison des deux fonctions de masse ne reflète absolument pas les deux croyances combinées *a priori*. En effet, ni A ni B ne persiste après la combinaison alors que ces deux mondes avaient chacun une preuve forte l'appuyant. Intuitivement, on pourrait penser que la combinaison des deux croyances devraient résulter en une croyance forte partagée entre A et B. Malheureusement, ce n'est pas le cas car la règle de combinaison de Dempster est conjonctive.

En terme d'interprétation, si l'on considère les fonctions de masse comme des preuves (la théorie des fonctions de croyance est aussi appelée *theory of evidence*, littéralement "théorie des

preuves") et la règle de combinaison de Dempster comme une combinaison neutre de preuves (cette règle n'accorde pas plus d'importance à une fonction de masse plutôt qu'à une autre), alors le résultat n'est pas si contre-intuitif. En effet, dans l'exemple de Zadeh, il est possible d'interpréter m_1 comme "il est presque certain que c'est A, peut-être C mais de façon certaine, ce n'est pas B". Ainsi, m_1 peut-être interprété comme une preuve formelle que ce n'est pas B. De même, on peut alors interpréter m_2 comme étant une preuve formelle que ce n'est pas A. Si l'on considère le cadre de discernement comme étant exhaustif, alors la combinaison des deux preuves donne que l'état du monde réel est nécessairement C. Cela est donc cohérent avec cette interprétation des fonctions de masse.

De la même manière, Dezert *et al.* interprètent leur exemple comme allant à l'encontre du bon sens (*"This result goes against common sense." sic*). Ils remettent en cause notamment l'impossibilité par la règle de Dempster de réviser une croyance trop spécifique. Comme pour l'exemple de Zadeh, il est possible d'interpréter les fonctions de masse données. Ainsi, dans l'exemple donné tableau 6.7, m_1 peut être interprétée comme étant une preuve formelle que C n'est pas possible. La fonction de masse m_2, quant à elle, défend un peu les trois hypothèses notamment en accordant de la masse à l'ignorance totale. Il n'est pas choquant que C disparaisse lors de la combinaison de ces deux preuves avec la règle de Dempster.

Finalement, ces deux exemples semblent illustrer plus des problèmes d'interprétation des fonctions de masse et de la règle de Dempster que des problèmes intrinsèques à la théorie des fonctions de croyance. Les deux exemples ont d'ailleurs comme cadre illustratif des diagnostics médicaux, des données subjectives donc (*soft data* en anglais), qui ne devraient en aucun cas être considérées comme preuves formelles. Un affaiblissement devrait être appliqué de façon automatique si l'on souhaite utiliser la règle de combinaison de Dempster afin de permettre la révision des opinions.

Dans cette thèse, nous avons donc considéré les fonctions de masse comme des preuves et la règle de combinaison de Dempster comme une accumulation de ces preuves. Avec cette interprétation en tête, nous nous sommes bien gardés de construire des fonctions de masse trop spécifiques et surtout pour lesquelles il n'y a pas de révision possible. C'est pourquoi, dans la plupart de nos modèles, une part d'ignorance totale est toujours conservée, bien que nos croyances reposent sur des données objectives (*hard data* en anglais). Il est d'ailleurs très peu probable qu'il soit possible d'obtenir une preuve formelle de quoi que ce soit. En effet, nos sources ne sont pas infaillibles (existe-t-il des sources infaillibles ?) et la transformation de nos mesures en croyances ne peut donc jamais être totalement sûre. Nous conseillons donc de bien suivre le principe de moindre engagement et surtout de toujours considérer les sources comme étant faillibles, quelles qu'elles soient.

De la récolte des mesures

Tout au long de ce document, des résultats d'applications avec de vrais capteurs ont été présentés. Nous avons décidé de récolter les mesures de nos capteurs périodiquement, en l'occurrence toutes les secondes dans la plupart des cas. En procédant ainsi, on fixe la réactivité de nos services reposant sur ce système. Ce n'est pas la seule manière de collecter les données issues

des capteurs. En effet, il est possible de récolter les données brutes soit tous les Δt, soit tous les Δv où v est la valeur retournée par notre capteur. Chaque approche a ses avantages et ses inconvénients.

Nous ne sommes *a priori* intéressés que par les changements de valeurs des mesures de nos capteurs. Ainsi, si un capteur mesure toujours la même chose, il ne devrait pas être nécessaire de renvoyer cette mesure encore et encore. Il est donc possible de renvoyer une nouvelle valeur uniquement lorsque celle-ci a suffisamment changé. Pour cela, on définit un seuil Δv correspondant à une variation au-delà de laquelle la nouvelle mesure sera renvoyée (*cf.* figure 6.14).

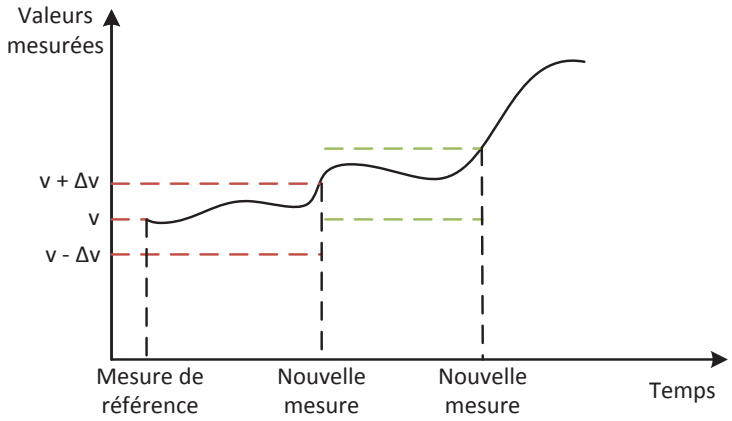

FIGURE 6.14 – Envoi de valeurs mesurées par un capteur tous les Δv.

Cette approche peut paraître séduisante. En effet, de cette manière, on peut observer toutes les variations significatives dans les mesures des capteurs. De plus, cela évite d'envoyer des données lorsque la valeur est constante.

Malheureusement, cette approche présente deux défauts majeurs :

- Perte de données : le principal défaut de cette approche est l'impact des pertes de données. En effet, lorsqu'on ne reçoit pas de nouvelle mesure, on ne peut pas savoir si c'est parce que la valeur mesurée est constante ou si les données sont perdues. Quand bien même on pourrait savoir si une donnée a été perdue, chaque perte ne ferait qu'augmenter le Δv. Plus les pertes sont importantes, plus on perd en sensibilité donc.
- Définition du Δv : la variation requise pour l'envoi d'une nouvelle donnée dépend de chaque type de capteur. Malheureusement, il n'est pas évident de définir la valeur de variation pertinente pour ce qui l'utilise. Un Δv trop grand conduit à une sensibilité trop faible, un Δv trop faible conduit à un envoi quasi continu des mesures. Cela peut notamment impacter le système de fusion de données soit en induisant une réactivité trop faible soit, au contraire, en le saturant d'informations. De plus, définir ce Δv peut être particulièrement complexe pour les capteurs dont la mesure n'est pas linéaire comme c'est le cas des capteurs de niveau sonore ou encore certains capteurs de distance.

C'est donc à cause de ces principaux défauts que nous avons opté pour l'approche Δt. Ainsi, en définissant une fréquence de mesure, on ne peut certes pas voir toutes les variations des

mesures mais les pertes de données ont un impact moindre comparé à l'approche Δv. De plus, il est tout à fait possible de modifier la fréquence à la volée en fonction des variations observées [68].

Perspectives

Des aspects n'ont pas pu être traités au cours de cette thèse. Les perspectives sont donc nombreuses. De plus, la construction d'un prototype complet a ouvert de nouveaux problèmes notamment concernant la gestion des équipements et le déploiement des services.

Gestion de conflit

Nous savons qu'il est possible dans la théorie des fonctions de croyance de caractériser le conflit et d'en trouver les sources [77, 56, 55, 76, 83]. Cela pourrait nous permettre de détecter les pannes de capteurs. Pour faire cela, il serait alors possible d'implémenter un nouveau composant qui analyserait le conflit pour un ou plusieurs attributs de contexte et qui pourrait modifier la configuration de capteurs exploités pour ce ou ces attributs de contexte par exemple en éteignant les capteurs considérés comme défaillants.

La gestion du conflit est généralisable à toutes les couches. Ainsi, dans notre architecture, la couche "Perception" nous permet de détecter les défaillances de la couche *Sensing*. De même, la couche supérieure pourrait permettre de détecter des incohérences dans les attributs de contexte reçus et déterminer des défaillances dans la couche "Perception". Chaque couche pourrait alors renvoyer un *feedback* à la couche inférieure lui indiquant un problème.

Distribution de la fusion

Nous avons vu qu'il serait possible de distribuer la combinaison des fonctions de masse sur un ensemble de calculateurs à condition d'utiliser une règle de combinaison associative et commutative. Les calculs nécessaires pour nos attributs de contexte étant assez limités, il est envisageable de répartir la fusion directement sur les nœuds de capteurs.

Cela demanderait bien entendu un vrai travail d'optimisation afin de ne pas surcharger les nœuds en calcul mais aussi de ne pas saturer les communications ni vider les batteries trop vite. De même, la perte de données devra être gérée afin de pouvoir appliquer les algorithmes de temporisation que nous avons présenté au chapitre 4. Des solutions à ces types de problèmes d'optimisation existent déjà [18, 60].

Construction automatique des modèles

Au chapitre 2, nous avons présenté les ensembles de fonctions de masse comme modèles permettant de transformer des données brutes issues de capteurs en fonctions de croyance. Ces modèles sont pour le moment construits par expertise à partir des mesures obtenues lors de quelques expérimentations. Ce processus, si l'on souhaite identifier un grand nombre d'attributs de contexte avec une variété de capteurs différents, demande un long travail d'ingénierie.

Un peu d'apprentissage pourrait permettre de réduire le temps nécessaire à la construction des modèles. En l'occurrence, il serait possible d'automatiser la construction de ces modèles par exemple en procédant à de la classification supervisée ou semi-supervisée. Ainsi, après installation du matériel, le système pourrait entrer dans une phase de calibrage qui demande d'effectuer certaines actions ou classes d'actions. A partir des données obtenues pendant cette phase, il peut alors déterminer les plages de mesures correspondant à tel ou tel état à observer. La mise en place d'un tel processus demandera certainement un travail de recherche.

Déploiement automatique des services

Aujourd'hui, les services dans notre plateforme sont déployés à la main en fonction de ce que l'on souhaite. Il pourrait être bénéfique pour un vrai système d'être capable, à partir des équipements déployés, de déterminer les données contextuelles qu'il est capable de calculer et de ces données contextuelles de déduire les services qu'il est possible de déployer.

En procédant ainsi, le système pourrait même suggérer l'installation de nouveaux équipements pour rendre accessibles de nouveaux services pour les habitants.

Standardisation des équipements

La standardisation des équipements aiderait certainement l'habitat intelligent. En effet, si l'ensemble des équipements blancs et bruns augmentés qui seront disponibles dans les années à venir disposaient tous des mêmes formats de données et des mêmes protocoles de communication, alors n'importe quel système pourrait facilement exploiter l'ensemble de ces données pour déduire de nouvelles données contextuelles et donc fournir tout un ensemble de services sensibles au contexte.

Marché aux services

Une standardisation des équipements augmentés, couplée à un système composé des algorithmes de fusion de données offrant plusieurs niveaux d'abstraction de données contextuelles, pourrait favoriser le développement d'une API permettant à quiconque de développer de nouveaux services exploitant des données contextuelles dans un habitat intelligent. Les services développés pourraient être regroupés sur une plateforme de distribution.

En procédant ainsi, chacun serait libre de développer les services dont il a besoin. Chacun pourrait donc aussi personnaliser la totalité des services déployés chez lui. Le développement des services serait alors déportés sur une main d'œuvre volontaire répondant à ses propres besoins. Chaque service pourrait alors spécifier les abstractions dont il a besoin. Ces abstractions pourraient elle-même être développées et partagées. On retrouve ce type de plateforme pour les applications pour smartphones. Les avantages en termes de démocratisation et de développement sont bien entendu importants.

En resumé, les perspectives pour le développement de l'habitat intelligent sont nombreuses. Le prototype que nous avons développé a permis de montrer les possibilités offertes par notre approche. Il reste cependant encore de nombreux aspects de recherche et d'ingénierie qui n'ont pas

pu être traités tels que l'acceptabilité réelle qui reste à expérimenter avec de véritables usagers, la mise au point de nouveaux attributs de contexte et de nouvelles situations à reconnaitre, la sécurité et la vie privée, le déploiement à grande échelle dans de nombreux habitats, l'entretien d'un tel système, et bien d'autres aspects encore.

Publications personnelles

[DFG+11] Michele Dominici, Myriam Fréjus, Julien Guibourdenche, Bastien Pietropaoli et Frédéric Weis, « Towards a System Architecture for Recognizing Domestic Activity by Leveraging a Naturalistic Human Activity Model », *in GAPREC 2011 : Workshop on Goal, Activity and Plan Recognition (Gaprec) at the International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

[DPW11a] Michele Dominici, Bastien Pietropaoli et Frédéric Weis, « Towards a feasibility-driven uncertainty-aware layered architecture for recognizing complex domestic activity », *in SAGAWARE 2011 : International Workshop on Situation, Activity and Goal Awareness at the 13th ACM International Conference on Ubiquitous Computing (UbiComp2011)*, p. 89–94, 2011.

[DPW11b] Michele Dominici, Bastien Pietropaoli et Frédéric Weis, « Towards a feasibility-driven uncertainty-aware layered architecture for recognizing complex domestic activity », *in Proceedings of the 2011 international workshop on Situation activity*, SAGAware '11, p. 89–94, 2011.

[DPW12] Michele Dominici, Bastien Pietropaoli et Frédéric Weis, « Experiences in managing uncertainty and ignorance in a lightly instrumented smart home », *International Journal of Pervasive Computing and Communications*, vol. 8, n° 3, p. 225–249, 2012.

[PDW11] Bastien Pietropaoli, Michele Dominici et Frédéric Weis, « Multi-sensor data fusion within the belief functions framework - application to smart home services », *in RUSMART 2011 : 4th International Conference on Smart Space and next generation wired/wireless networking s*, p. 123–134, 2011.

[PDW12] Bastien Pietropaoli, Michele Dominici et Frédéric Weis, « Belief inference with timed evidence : methodology and application using sensors in smart home », *in Belief 2012 : The 2nd International Conference on Belief Functions*, vol. 164 *in Advances in Intelligent and Soft Computing*, p. 409–416, 2012.

[PDW13a] Bastien Pietropaoli, Michele Dominici et Frédéric Weis, « Propagation of belief functions through frames of discernment : Application to context computing », *in FLAIRS Conference*, 2013.

[PDW13b] Bastien Pietropaoli, Michele Dominici et Frédéric Weis, « Virtual sensors and data fusion in a multi-level context computing architecture », *in Information Fusion (FUSION), 2013 16th International Conference on*, 2013.

Bibliographie

[1] M. Raisul Alam, M. Bin Ibne Reaz, and M. Alauddin Mohd. Ali. A review of smart homes - past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6) :1190–1203, 2012.

[2] F. Aldrich. Smart homes : Past, present and future. In *Inside the Smart Home*, pages 17–39. 2003.

[3] A. Appriou. Probabilités et incertitude en fusion de données multi-senseurs. *Revue scientifique de la Défence*, 11 :27–40, 1991.

[4] A. Aregui and T. Denœux. Constructing consonant belief functions from sample data using confidence sets of pignistic probabilities. *International Journal of Approximate Reasoning*, 49 :575–594, November 2008.

[5] L. Atzori, A. Iera, and G. Morabito. The internet of things : A survey. *Comput. Netw.*, 54(15) :2787–2805, October 2010.

[6] J. A. Barnett. Computational methods for a mathematical theory of evidence. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 868–875, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[7] I. Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd, 2008.

[8] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2) :161–180, April 2010.

[9] H. Boström, S. F. Andler, M. Brohede, R. Johansson, A. Karlsson, J. van_Laere, L. Niklasson, M. Nilsson, A. Persson, and T. Ziemke. On the Definition of Information Fusion as a Field of Research. Technical report, University of Skovde, School of Humanities and Informatics, Skovde, Sweden, 2007.

[10] A. Boytsov and A. Zaslavsky. Extending context spaces theory by proactive adaptation. In Sergey Balandin, Roman Dunaytsev, and Yevgeni Koucheryavy, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, volume 6294 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2010.

[11] A. Boytsov and A. Zaslavsky. Ecstra : Distributed context reasoning framework for pervasive computing systems. In Sergey Balandin, Yevgeni Koucheryavy, and Honglin Hu, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, volume 6869 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2011.

[12] A. Boytsov, A. Zaslavsky, and K. Synnes. Extending context spaces theory by predicting run-time context. In *NEW2AN '09 and ruSMART '09 : Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces*, pages 8–21, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] K. Caine, S. Sabanovic, and M. Carter. The effect of monitoring by cameras and robots on the privacy enhancing behaviors of older adults. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 343–350, 2012.

[14] S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2) :31–48, March 2001.

[15] P. Chahuara, M. Vacher, and F. Portet. Localisation d'habitant dans un environnement perceptif non visuel par propagation d'activation multisource. In *MAJECSTIC*, page 8pp., Bordeaux, France, 13-15 oct. 2010.

[16] L.-Z. Chen, W.-K.i Sh, Y. Deng, and Z.-F. Zhu. A new fusion approach based on distance of evidences. In *Journal of Zhejiang University Science 6A(5)*, pages 476,482. 2005.

[17] G. Cleuziou and J. Sublemontier. Etude comparative de deux approches de classification recouvrante : Moc vs. OKM. In *8èmes journées d'Extraction et de Gestion des Connaissances (EGC'2008)*, pages 667–678, France, 2008.

[18] P. Coucheney. *Auto-optimisation des réseaux sans fil. Une approche par la théorie des jeux.* PhD thesis, Université de Grenoble, 2011.

[19] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is key. *Commun. ACM*, 48 :49–53, March 2005.

[20] W. Dargie. The role of probabilistic schemes in multisensor context-awareness. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops 2007), 19-23 March 2007, White Plains, New York, USA*, pages 27–32. IEEE Computer Society, 2007.

[21] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2), 1967.

[22] T. Denœux. A k-nearest neighbor classification rule based on Dempster-Shafer Theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25 :804–813, 1995.

[23] T. Denœux. Conjunctive and disjunctive combination of belief functions induced by non distinct bodies of evidence. *Artificial Intelligence*, 2007.

[24] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1) :4–7, January 2001.

[25] J. Dezert, D. Han, Z. Liu, and J.-M. Tacnet. Hierarchical proportional redistribution for bba approximation. In *Belief Functions*, pages 275–283, 2012.

[26] J. Dezert, Pei Wang, and A. Tchamova. On the validity of dempster-shafer theory. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 655–660, 2012.

[27] M. Dominici. *Contributing to Energy-efficiency through a User-centered Smart Home.* PhD thesis, Université de Rennes 1, France, 2013.

[28] D. Dubois. Possibility theory and statistical reasoning. *Computational Statistics & Data Analysis*, 51(1) :47–69, November 2006.

[29] D. Dubois and H. Prade. Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence*, 4 :244–264, 1988.

[30] D. Dubois and H. Prade. La problématique scientifique du traitement de l'information. Rapport de recherche 02-08R, IRIT, Université Paul Sabatier, Toulouse, mars 2002. bb.

[31] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, 2004.

[32] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2) :114–131, June 2003.

[33] M. C. Florea, A.-L. Jousselme, É. Bossé, and D. Grenier. Robust combination rules for evidence theory. *Information Fusion*, 10(2) :183 – 197, 2009.

[34] M. Fréjus and J. Guibourdenche. Analysing domestic activity to reduce household energy consumption. *Work*, 41, 2012.

[35] C. W. Geib and R. P. Goldman. Plan recognition in intrusion detection systems. *DARPA Information Survivability Conference and Exposition*, 1 :0046, 2001.

[36] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11) :1101 – 1132, 2009.

[37] M. Ghallab, D. Nau, and P. Traverso. Hierarchical task network planning. pages 229 – 261, 2004.

[38] R. P. Goldman, C. W. Geib, and C. A. Miller. A New Model of Plan Recognition. *Artificial Intelligence*, 64 :53–79, 1999.

[39] S. Greenberg and C. Fitchett. Phidgets : easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 209–218, New York, NY, USA, 2001. ACM.

[40] R. Haenni. Are alternatives to dempster's rule of combination real alternatives ? : Comments on "about the belief function combination and the conflict management problem" - lefevre et al. *Information Fusion*, 3(3) :237–239, 2002.

[41] R. Haenni and N. Lehmann. Implementing belief function computations. *International Journal of Intelligent Systems*, 18 :31–49, 2003.

[42] D. Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996.

[43] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney, and S. Devlin. Evidential fusion of sensor data for activity recognition in smart homes. *Pervasive and Mobile Computing*, 5(3) :236 – 252, 2009. Pervasive Health and Wellness Management.

[44] A.-M. Jolly, B. Marhic, L. Delahoche, C. Solau, and D. Menga. Detection of abnormal sensor behaviour using TBM. In *BELIEF10*, pages 1–10, Brest, France, April 2010.

[45] A.-L. Jousselme and P. Maupin. Distances in evidence theory : Comprehensive survey and generalizations. *Int. J. Approx. Reasoning*, 53(2) :118–145, February 2012.

[46] L. Kalra, X. Zhao, A. J. Soto, and E. E. Milios. A two-stage corrective markov model for activities of daily living detection. In P. Novais, K. Hallenborg, D. I. Tapia, and J. M. Corchado Rodríguez, editors, *ISAmI*, volume 153 of *Advances in Intelligent and Soft Computing*, pages 171–179. Springer, 2012.

[47] B. W. Kernighan. *The C Programming Language*. Prentice Hall Professional Technical Reference, 2nd edition, 1988.

[48] J.M. Keynes. Chapter iv : The principle of indifference. *A Treatise on Probability*, 4 :41–64, 1921.

[49] B. Khaleghi, A. Khamis, and F. O. Karray. Multisensor data fusion : A review of the state-of-the-art. *Information Fusion*, 14(1) :28–44, August 2013.

[50] M. La Placa, H. Pigot, and F. Kabanza. Assistive planning for people with cognitive impairments. In *Proc. of Workshop on Intelligent Systems for Assisted Cognition hosted by Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[51] E. Lefevre, Olivier Colot, and Patrick Vannoorenberghe. Belief function combination and conflict management. *Information Fusion*, 3(2) :149–162, 2002.

[52] J. Liao, Y. Bi, and C. Nugent. Activity Recognition for Smart Homes Using Dempster-Shafer Theory of Evidence Based on a Revised Lattice Structure. In *Proceedings of the 2010 Sixth International Conference on Intelligent Environments*, IE '10, pages 46–51, Washington, DC, USA, 2010. IEEE Computer Society.

[53] S. W. Loke. On representing situations for context-aware pervasive computing : six ways to tell if you are in a meeting. In *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, PERCOMW '06, pages 35–39, Washington, DC, USA, 2006. IEEE Computer Society.

[54] A. Martin. Implementing general belief function framework with a practical codification for low complexity. In Florentin Smarandache & Jean Dezert, editor, *Advances and Applications of DSmT for Information Fusion*, page Pnd. American Research Press Rehoboth, December 2008.

[55] A. Martin. About conflict in the theory of belief functions. In T.Denœux and Masson [87], pages 161–168.

[56] A. Martin, A. L Jousselme, and C. Osswald. Conflict measure for the discounting operation on belief functions. In *Information Fusion, 2008 11th International Conference on*, pages 1–8, 2008.

[57] A. Martin, C. Osswald, J. Dezert, and F. Smarandache. General combination rules for qualitative and quantitative beliefs. *CoRR*, abs/0906.5119, 2009.

[58] S. Mckeever, J. Ye, L. Coyle, C. Bleakley, and S. Dobson. Activity recognition using temporal evidence theory. *J. Ambient Intell. Smart Environ.*, 2(3) :253–269, August 2010.

[59] S. McKeever, J. Ye, L. Coyle, and S. Dobson. Using Dempster-Shafer Theory of Evidence for Situation Inference. In Payam Barnaghi, Klaus Moessner, Mirko Presser, and Stefan Meissner, editors, *Smart Sensing and Context*, volume 5741 of *Lecture Notes in Computer Science*, pages 149–162. Springer Berlin / Heidelberg, 2009.

[60] L. Mottola and G. P. Picco. Programming wireless sensor networks with logical neighborhoods. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, InterSense '06, New York, NY, USA, 2006. ACM.

[61] C. K. Murphy. Combining belief functions when evidence conflicts. *Decision Support Systems*, 29 :1–9, 2000.

[62] K. Ni, N. Ramanathan, M. Nabil Hajj Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3) :25 :1–25 :29, June 2009.

[63] J.R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[64] Chris D. Nugent, Xin Hong, Josef Hallberg, Dewar D. Finlay, and Kåre Synnes. Assessing the impact of individual sensor reliability within smart living environments. In *CASE*, pages 685–690, 2008.

[65] C. Osswald. Controling the number of focal elements - some combinatorial considerations. In *Belief Functions*, pages 135–143, 2012.

[66] C. Osswald and A. Martin. Understanding the large family of Dempster-Shafer theory's fusion operators - a decision-based measure. In *Information Fusion, 2006 9th International Conference on*, pages 1–7, 2006.

[67] A. Oulasvirta, A. Pihlajamaa, J. Perkiö, D. Ray, T. Vähäkangas, T. Hasu, N. Vainio, and P. Myllymäki. Long-term effects of ubiquitous surveillance in the home. In *Proceedings of the 14th International Conference on Ubiquitous Computing (Ubicomp 2012)*, Pittsburgh, Pennsylvania, September 2012.

[68] A. Padovitz. *Context Management and Reasoning about Situations in Pervasive Computing*. PhD thesis, Monash University, Australia, 2006.

[69] A. Padovitz, S.W. Loke, A. Zaslavsky, and B. Burg. Verification of uncertain context based on a theory of context spaces. *International Journal of Pervasive Computing and Communications*, 3(1) :30–56, 2007.

[70] A. Padovitz, A. Zaslavsky, and S. W. Loke. A unifying model for representing and reasoning about context under uncertainty. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, July 2006.

[71] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4) :50–57, October 2004.

[72] G. Powell and M. Roberts. GRP1. A recursive fusion operator for the transferable belief model. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8, 2011.

[73] G. Powell, M. Roberts, and D. Stampouli. Improvements to the GRP1 Combination Rule. In T. Denœux and M.-H. Masson, editors, *Belief Functions : Theory and Applications*, volume 164 of *Advances in Intelligent and Soft Computing*, pages 293–300. Springer Berlin Heidelberg, 2012.

[74] D. Preuveneers and Y. Berbers. Internet of things : A context-awareness perspective. In Lu Yan, Yan Zhang, Laurence T. Yang, and Huanshen Ning, editors, *The Internet of Things : From RFID to the Next-Generation Pervasive Networked Systems*, pages 287–307. Auerbach Publications, Taylor and Francis Group, New York, USA, March 2008.

[75] S. Régis, A. Doncescu, M. Takizawa, and G. Cleuziou. Initialization of Masses by the OKM for the Belief Function Theory : Application to System Biology. In L. Barolli, T. Enokido, F. Xhafa, and M. Takizawa, editors, *AINA Workshops*, pages 1167–1171. IEEE, 2012.

[76] V. Ricquebourg, M. Delafosse, L. Delahoche, B. Marhic, AM Jolly-Desodt, and D. Menga. Fault Detection by Combining Redundant Sensors : a Conflict Approach Within the TBM Framework. In *COGIS 2007, COGnitive systems with Interactive Sensors*. Stanford University, 2007.

[77] S.Destercke and T. Burger. Revisiting the notion of conflicting belief functions. In T.Denœux and Masson [87], pages 153–160.

[78] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[79] Z. Shelby and C. Bormann. *6LoWPAN : The Wireless Embedded Internet*. John Wiley & Sons, Ltd, 2009.

[80] P. Smets. Theories of uncertainty. In IOS Press, editor, *Handbook of Fuzzy Computation*, chapter Section B.1.2. IOS Press, 1998.

[81] P. Smets. The transferable belief model for quantified belief representation. In Philippe Smets, editor, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 1, pages 267–301. 1998.

[82] P. Smets. Belief functions on real numbers. *Int. J. Approx. Reasoning*, 40(3) :181–223, November 2005.

[83] P. Smets. Analyzing the combination of conflicting belief functions. *Information Fusion*, 8(4) :387–412, 2007.

[84] P. Smets and R. Kruse. The transferable belief model for belief representation. In *Uncertainty Management in Information Systems*, pages 343–368. Kluwer Academic Publishers, Boston, 1996.

[85] G.J. Székely. *Paradoxes in probability theory and mathematical statistics*. Mathematics and its applications. East European series. D. Reidel, 1986.

[86] L. Takayama, C. Pantofaru, D. Robson, B. Soto, and M. Barry. Making technology homey : finding sources of satisfaction and meaning in home automation. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 511–520, New York, NY, USA, 2012. ACM.

[87] T.Denœux and M.-H. Masson, editors. *Belief Functions : Theory and Applications - Proceedings of the 2nd International Conference on Belief Functions, Compiègne, France, 9-11 May 2012*, volume 164 of *Advances in Soft Computing*. Springer, 2012.

[88] J. Theureau. *Handbook of cognitive task design*, chapter Course-of-action analysis and course-of-action-centered design, pages 55–81. Lawrence Erlbaum Associates, New Jersey, 2003.

[89] M. Valtonen, T. Vuorela, L. Kaila, and J. Vanhala. Capacitive indoor positioning and contact sensing for activity recognition in smart homes. *J. Ambient Intell. Smart Environ.*, 4(4) :305–334, December 2012.

[90] P Vannoorenberghe. Un état de l'art sur les fonctions de croyance appliquées au traitement de l'information. Rapport technique Revue 13 (20), CNRS, Université de Rouen, UFR des Sciences, 2004.

[91] P. Wang. A defect in dempster-shafer theory. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, UAI'94, pages 560–566, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[92] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7) :74–84, 1993.

[93] M. Weiser. The world is not a desktop. *Interactions*, 1(1) :7–8, 1994.

[94] M. Weiser. Human-computer interaction. chapter The computer for the 21st century, pages 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[95] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[96] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012.

[97] R. R. Yager. On the dempster-shafer framework and new combination rules. *Inf. Sci.*, 41 :93–137, March 1987.

[98] R. R. Yager. Entropy and specificity in a mathematical theory of evidence. In Roland Yager and Liping Liu, editors, *Classic Works of the Dempster-Shafer Theory of Belief Functions*, volume 219 of *Studies in Fuzziness and Soft Computing*, pages 291–310. Springer Berlin / Heidelberg, 2008.

[99] K. Yamada. A new combination of evidence based on compromise. *Fuzzy Sets and Systems*, 159(13) :1689–1708, 2008.

[100] L. A. Zadeh. Fuzzy logic and its application to approximate reasoning. In *IFIP Congress*, pages 591–594, 1974.

[101] L.A. Zadeh. Fuzzy sets. *Information Control*, 8 :338–353, 1965.

[102] L.A. Zadeh. *On the Validity of Dempster's Rule of Combination of Evidence*. Memorandum UCB/ERL-M. Electronics Research Laboratory, University of California, 1979.

[103] D. Zhang, M. Guo, J. Zhou, D. Kang, and J. Cao. Context reasoning using extended evidence theory in pervasive computing environments. *Future Gener. Comput. Syst.*, 26(2) :207–216, February 2010.

Résumé

L'habitat intelligent est l'objet de nombreux travaux de recherche. Il permet d'assister des personnes âgées ou handicapées, d'améliorer le confort, la sécurité ou encore d'économiser de l'énergie. Aujourd'hui, l'informatique ubiquitaire se développe et s'intègre dans l'habitat intelligent notamment en apportant la sensibilité au contexte. Malheureusement, comprendre ce qui se passe dans une maison n'est pas toujours facile.

Dans cette thèse, nous explicitons comment le contexte peut permettre de déployer des services adaptés aux activités et aux besoins des habitants. La compréhension du contexte passe par l'installation de capteurs mais aussi par l'abstraction des données brutes en données intelligibles facilement exploitables par des humains et des services. Nous mettons en avant une architecture multi-couches de fusion de données permettant d'obtenir des données contextuelles de niveaux d'abstraction différents. La mise en place des couches basses y est présentée en détail avec l'application de la théorie des fonctions de croyance pour l'abstraction de données brutes issues de capteurs. Enfin, sont présentés le déploiement d'un prototype nous ayant permis de valider notre approche, ainsi que les services déployés.

Abstract

Smart home is a major subject of interest. It helps to assist elderly or disabled people, improve comfort, safety, and also save energy. Today, ubiquitous computing is developed and integrated into the smart home providing context-awareness. Unfortunately, understanding what happens in a home is not always easy.

In this thesis, we explain how context can be used to deploy services tailored to the activities and needs of residents. Understanding context requires the installation of sensors but also the abstraction of raw data into easily understandable data usable by humans and services. We present a multi-layer architecture of data fusion used to obtain contextual information of different levels of abstraction. The implementation of the lower layers is presented in detail with the application of the theory of belief functions for the abstraction of raw sensor data. Finally, are presented the deployment of a prototype that allowed us to validate our approach and the deployed services.