République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle
en vue d'obtention du diplôme de master recherche en informatique
spécialité : Réseaux et systèmes distribués

Thème

La modélisation hybride des connaissances contextuelles dans un environnement ubiquitaire.

Présenté par

M^{r} ***AROUA*** **Adel**

M^{elle} ***BOUARAR*** **Rahima**

Présidente	M^{me} ***ALOUI*** **Soria**
Examinatrice	M^{me} ***KHALED*** **Hayet**
Examinateur	M^{r} ***SEBAA*** **Abd arrezak**
Encadreur	M^{me} **BOUTRID Samia**

Promotion 2012/2013

Remerceiments

Nous remercions avant tous ALLAH le tout puissant
de nous avoir donné la force, la
foie, le courage et la volonté pour réaliser cet
humble travail.
Nos remerciments vont également à nos parents pour
tous les sacrifices qu'ils ont consentis pour nous permettre de suivre nos études
dans les meilleurs conditions
possibles et n'avoir jamais cessez de nous encouragé
tout au long de nos années d'études.
Qu'il nous soit permis de rendre un vibrant hommage à
notre Encadreur Mme BOUTRID Samia
pour sa disponibilité, son savoir faire et son soutien.
Nous remercions également tous ceux qui ont contribués
à la réalisation de ce travail.
Enfin, nous tenons à remercier les membres du jury
qui ont acceptés de juger notre travail.

Dédicace

Je dédie ce modeste travail à :
Mes très chers parents sans qui je ne serai pas
l'homme que je suis.
Mes frères, mes cousins ainsi que toute la famille pour qui
je souhaite une immense réussite.
Notre Promotrice Madame " BOUTRID Samia " pour nous
avoir prodigués d'utiles conseils et orientations dans la réalisation
de ce travail et pour tout l'aide et la compréhension dont elle a fait
preuve à notre égard.
Mon binôme RAHIMA avec qui j'ai partagé le bon et le
pire pendant la réalisation de ce travail.
Mes amis qui m'ont beaucoup encouragé pour mettre
au point ce travail, et tous les autres avec qui j'ai vécu une
année très agréable.
Et finalement à toutes personnes qui me connaissent
de prés ou de loin.

Adel

Dédicace

Je dédie ce modeste travail à :

A ma très précieuse et ma bien aimée, ma mère

qui a partagé mes joies et mes soucis, et qui a tout sacrifié pour ma réussite

A la mémoire de mes grands parents

A mon très cher oncle Dada Brahim

A Notre Promotrice Madame " BOUTRID Samia "

A Mon binôme Adel et sa famille

A mes trés chers frères Idir, Aziz, Karim, Hmoumou

A mes très chères soeurs Fahima, Karima, Khalida

A mes beaux-frères Idir et Hassene

A ma belle-soeur Lylia

A mon neveu Yani et ma nièce Dylia

A tous mes cousins et cousines

A mon meilleur ami Salah qui a vraiment été à mes cotés.

A mes meilleures amis :

Alima,Ouarda,Kenza,Ghania,Dihia,Dihou, Nabila ,Katia,Kahina,Nassima,

Sabrina,Zahra,Dida, Amel,Karima,Kimouche,Nassim,Lamine,Saadi,Sofiane

Et finalement à toutes personnes qui me connaissent de prés ou de loin.

Rahima

Résumé

L'évolution technologique des dispositifs mobiles a donné naissance à de nouveaux besoins applicatifs pour assurer l'exécution des applications dans des environnements dynamiques. Ces applications appelées applications sensibles au contexte doivent détecter les variations de l'environnement et s'adapter en conséquence. Le développement de ce type d'applications est difficile à mettre en oeuvre et nécessite de grands efforts de programmation. La majorité des travaux existants, concernant le contexte et l'adaptation, focalisent leurs efforts à proposer des modèles pour décrire le contexte.

Ce mémoire s'intéresse à faciliter la description et le raisonnement sur le contexte en utilisant les approches de modélisation hybrides, pour cela, nous proposons l'approche de modélisation hybride basé sur les ontologies et l'Event Calculs.

Afin de valider cette proposition, nous avons présenté l'architecture globale de notre plateforme et nous avons proposé un petit scénario de supervision pour montrer et illustrer le genre de scénario qui peut être exécuté sur la plate-forme et nous avons décrit ce scénario avec les ontologies et effectué le raisonnement sur les informations avec les prédicat et les règles de Event Calculus.

Mots-clés : Informatique ubiquitaire, Sensibilité au contexte, Modélisation du contexte, Modélisation hybride, Ontologies, Event Calculs, Raisonnement.

Abstract

The technological change of the mobile devices gave rise to new applicatifs needs to ensure the execution of the applications in dynamic environments. These applications called applications sensitive to the context must detect the variations of the environment and adapt consequently. The development of this type of applications is difficult to implement and requires main efforts of programming. The majority of existing work, concerning the context and the adaptation, focus their efforts to propose models to describe the context.

This memoirs is interested to facilitate description and the reasoning on the context by using the hybrid approaches of modeling, for that, we propose the approach of hybrid modeling based on ontology and the Event Calculus.

In order to validate this proposal, we presented the total architecture of our platform and we proposed a small scenario of supervision to show and to illustrate the kind of scenario which can be carried out on the platform and we described this scenario with ontologies and carried out the reasoning on information with the predicate and the rules of Event Calculus.

Keywords : Ubiquitous computing, Sensitivity to the context, Modeling of the context, Hybrid modeling, Ontology, Event Calculus, Reasoning.

LISTE DES ABRÉVIATIONS

WLAN : Wireless Local Area Network
WMAN : Wireless Metropolitan Area Network
WWAN : Wireless Wide Area Network
WPAN : Wireless Personal Area Network
GSM : Global System For Mobile Communication
GPRS : General Packet Radio System
UMTS : Universal Mobile Télécommunication System
PSDN : Packet Switched Data Network
DSL : Digital Suscriber Line
PDA : Le Personal Digital Assistants
RFID : Radio Frequency Identification
UML : Unified Modeling Language
RDF : Resource Description Framework
RDFS : Resource Description Framework Schema
W3C : World Wide Web Consortium
URI : Uniform Resource Identifier
XML : eXtended Markup Language
SWRL : Semantic Web Rule Language
DTD : Document Type Definition
CC/PP : Composit Capability/Prefrence Profile
CML : Context Modeling Langage

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

INTRODUCTION GÉNÉRALE

L'évolution technologique réalisée dans le domaine de l'informatique ne cesse de croître et marque profondément notre société actuelle. Dans une société où l'accès au savoir et à l'information est primordial, l'intégration et le passage rapide à l'exploitation des dernières technologies de communication et d'accès à l'information ont contribué à l'émergence de l'informatique dans différents domaines. Le récent développement des terminaux, et plus généralement des équipements informatiques, et les technologies des systèmes embarqués rendent aujourd'hui réalisable la vision du 21ème siècle de Mark Weiser (Weiser, 1991) [1] où les systèmes informatiques sont omniprésents dans la vie de tous les jours. Cette vision où la technologie devient invisible à l'utilisateur est connue sous le nom de l'informatique diffuse (en anglais " pervasive computing "). En effet ces systèmes sont au carrefour de plusieurs domaines technologiques (exemple multimédia, réseau,systèmes distribués et embarqués) et préoccupations sociales (exemple confidentialité,fiabilité, sécurité), ce qui les rend particulièrement difficiles à construire et à vérifier.

Les systèmes informatiques ubiquitaires reposent sur une notion centrale : le contexte à partir du quel ils réagissent et s'adaptent aux différents changements de l'environnement.

En effet, ces systèmes, pour être utiles et utilisables doivent faciliter la tâche de l'utilisateur. Ils leurs sont ainsi nécessaire de posséder la capacité de s'adapter à leur environnement, à leur contexte, afin de faire intervenir l'utilisateur le moins possible dans la reconfiguration du système pour qu'il tire au mieux parti de son nouvel environnement.

Les applications qui détectent les changements dans l'environnement et qui tirent parti de ces changements pour adapter leur comportements en conséquences sont appelées" applications sensibles au contexte ".

Le développement "d'applications sensibles au contexte" est par nature complexe. Ces applications s'adaptent à l'évolution des informations venant du contexte : contexte physique, contexte informatique et contexte de la tâche de l'utilisateur.

La communauté informatique pervasive comprend de plus en plus les avantages de la mo-

délisation formelle de l'information du contexte. Tout d'abord, en raison de la complexité inhérente aux applications contextuelles compatibles, le développement doit être soutenu par des méthodes de génie logiciel adéquates. L'objectif global est de développer des applications contextuelles capables d'évolution de manière autonome. Un bon formalisme de modélisation d'information sur le contexte permet de réduire la complexité des applications contextuelles et améliore leur maintenabilité et évolutivité. En outre, la collecte, l'évaluation et la maintenance des informations sur le contexte sont coûteuses. La réutilisation et le partage des informations sur le contexte entre les applications contextuelles devraient être considérés dès le début de la modélisation. L'existence de modèles bien conçus des informations sur le contexte facilite le développement et le déploiement des applications.

Les approches actuelles de modélisation des informations sur le contexte ou modélisation contextuelle varient dans la facilité avec laquelle les objets du monde réel peuvent être capturés par les ingénieurs logiciels :

- dans le pouvoir expressif des modèles des informations sur le contexte ;
- dans l'aide qu'elles peuvent fournir pour le raisonnement sur des informations sur le contexte ;
- dans la performance du raisonnement ;
- dans l'évolutivité de la gestion des informations sur le contexte.

Un grand nombre d'applications sensibles au contexte basées sur des modèles de contexte différents ont été développées au fil des années dans divers domaines d'application. Ces expériences influencent l'ensemble des exigences définies pour la modélisation de contexte et du raisonnement, et donc également influencent la recherche sur les modèles d'informations sur le contexte.

Plusieurs travaux ont défini des approches de modélisation hybride de description contexte qui déterminent des avantages évidents en ce qui concerne les exigences de la modélisation mais elles ne sont pas généralement satisfaisantes en raison de certaines limitation.

Donc notre objectif était de proposer une approche de modélisation hybride, pour avoir un haut pouvoir expressif, pour supporter le raisonnement sur le contexte et pour avoir de bonne performance dans le raisonnement.

Après cette introduction générale, ce mémoire comporte trois parties principales : Un état de l'art ,notre proposition et la validation de cette proposition.

La partie état de l'art est composée de trois chapitres :

Le premier chapitre présente des généralités sur les systèmes ubiquitaires à savoir : la définition de l'informatique ubiquitaire, son environnement, ses caractéristiques et ses contraintes, les équipements de l'environnement ubiquitaire, et nous citons quelques domaines d'applications.

Dans le deuxième chapitre, nous y détaillons la notion du contexte, la notion de la sensibilité au contexte, les exigences de la modélisation, les approches de modélisation de contexte, la

problématique, et les différentes approches de modélisation hybride proposées.

Le troisième chapitre est dédié à la présentation des deux approches de modélisation Event Calculus et OWL_DL.

La deuxième partie, qui regroupe l'ensemble de nos contributions.

La troisième partie, illustre la validation de notre proposition.

Enfin, notre mémoire se clôture par une conclusion générale résumant les différents points qui ont été traités.

CHAPITRE 1

INTRODUCTION AUX SYSTÈMES UBIQUITAIRES.

1.1 Introduction

Face à la montée en puissance des nouvelles technologies de l'information et de la communication, les utilisateurs ont vu leurs outils de communication passer d'un ordinateur de bureau à un ensemble de moyens mobiles dotés de plusieurs terminaux interconnecté via des réseaux et qui petit à petit vont pénétrer tous les objets de notre quotidien. Grace aux terminaux mobile et les services offerts, les objets du quotidien sont partout et accessible à tout moment. C'est l'informatique ubiquitaire. Pour bien assimiler ce concept nous allons aborder dans ce premier chapitre ses principales généralités.

1.2 Définition de l'informatique ubiquitaire

Le concept d'informatique ubiquitaire ou informatique ambiante converge vers la vision de l'informatique du futur que donnait Mark Weiser en 1991 dans son articleintitulé "The Computer for the 21st Century" :

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [1].

L'informatique ubiquitaire signifie l'omniprésence de supports informatiques par tout autour de nous et à tout moment. Certains de ces supports font désormais partie de notre environnement quotidien comme les ordinateurs embarqués dans les voitures, et dans les appareils électroménagers. De nos jours le concept d'informatique ubiquitaire prend tout son sens grâce aux nouvelles technologies émergentes (RFID, Wifi, ..) et aux comportements associés (mobilité, réseaux sociaux de proximité, ...), Ainsi l'informatique ubiquitaire a donné naissance à plusieurs termes parmi eux on trouve :

- **Ubiquitaire :** Accessible de n'importe où ;
- **Mobile :** Qui intègre les terminaux mobiles ;
- **Context-aware (sensibilité au contexte) :** Qui prend en compte le contexte d'exécution ;
- **Pervasif :** Qui associe ubiquité, mobilité et context-awareness ;
- **Ambiant :** Qui est intégré dans les objets quotidiens.[1]

1.3 Définition d'un environnement ubiquitaire

Un environnement de l'informatique ubiquitaire (ou environnement ubiquitaire) est un espace physique (une maison, un hôpital, une école, une autoroute, ou une ville) équipé d'une multitude d'entités matérielles ou logicielles communiquant grâce à un réseau. Ces entités sont des capteurs (de température, de luminosité), des actionneurs (lampes, caméra), des technologies mobiles (téléphones portables multi-fonctions), ou encore des composants logiciels (agendas, applications web). Ces entités capturent les informations de l'environnement ubiquitaire et sont programmables, donnant ainsi accès à leurs fonctionnalités, elles communiquent entre elles pour coopérer. Ceci se fait d'une manière transparente pour l'utilisateur sans son intervention explicite et lui offre la possibilité de se concentrer sur sa tâche principale au lieu de configurer et de gérer l'ensemble des équipements informatiques mis à sa disposition (Figure 1.1). L'informatique ubiquitaire favorise par exemple la création d'environnements intelligents tels que la maison intelligente capable de gérer automatiquement les différents équipements présents au domicile de l'utilisateur, elle est appelée informatique "sensible au contexte", où le contexte représente les informations capturées par les entités [2].

1.4 Scénario illustrant l'informatique ubiquitaire

Pour rendre clairs les concepts fondamentaux des systèmes diffus, nous présentons dans la section suivante un simple scénario d'un système diffus que nous supposons être installé dans le pavillon A de l'école de technologie supérieure [3]. Ensuite, nous procédons à une analyse de ce scénario pour expliquer et mettre en évidence le fonctionnement de ces systèmes ainsi que leurs composants.

1.4.1 Scénario

En entrant au pavillon A de l'école de technologie supérieure pour assister à un cours qui démarrera dans dix minutes et qui durera trois heures. Paul reçoit un appel sur son téléphone cellulaire de la part de son directeur de thèse pour lui rappeler qu'aujourd'hui (et avant deux

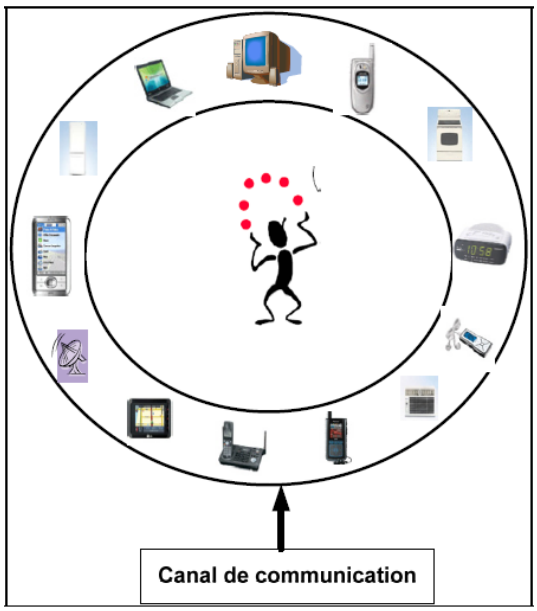


FIGURE 1.1 – Ubiquité des équipements informatiques.

heures) est la date limite pour envoyer trois rapports (documents volumineux sauvegardés dans son ordinateur portatif) à son codirecteur de thèse qui se trouve en France pour finaliser la procédure d'accueil pour l'été prochain. Tout calcul fait, Paul se rend compte qu'il est dans l'obligation de faire cette tâche avant d'assister au cours et sans s'absenter. Il se dirige alors vers la cafétéria et s'authentifie pour l'accès au réseau sans fil en utilisant son ordinateur portable. Il ouvre sa boîte de courriel et tape le message qui accompagne les rapports. Lors de la jointure de son premier document, un message apparaît sur son écran lui signalant que d'après la vitesse actuelle du réseau, l'expédition sera trop lente parce que plusieurs étudiants à la cafétéria naviguent sur internet. Le système remarque que le débit à la bibliothèque est très bon et qu'il y a peu d'étudiants qui utilisent internet, alors il propose à Paul de se déplacer vers la bibliothèque qui se trouve à deux minutes de la cafétéria. Paul accepte la proposition et se dirige vers la bibliothèque.

En entrant dans celle-ci, son téléphone cellulaire se met automatiquement en mode vibreur pour respecter les règles du lieu. Paul procède vite à l'attache des trois documents à son message. Au cours de cette opération, il reçoit une alerte SMS qui s'affiche automatiquement cette fois sur son téléphone cellulaire indiquant qu'il lui reste trois minutes pour le prochain cours. Paul termine sa tâche et quitte le lieu en route vers la salle de cours.

1.4.2 Analyse du scénario

D'après le scénario, on peut facilement dégager les caractéristiques fondamentales d'un système informatique diffus. Il offre :

- **U**n service transparent sur un réseau, en particulier sur internet pour différentes tâches (naviguer ou expédier un courriel) ;
- **D**es équipements connectés par différents moyens (WiFi, Bluetooth,) ;
- **U**n système qui surveille le contexte et les paramètres du service qu'il offre aux utilisateurs.

Le système doit s'adapter aux différents contextes pour aider l'utilisateur dans ses tâches et lui permettre de se concentrer sur ces tâches principales sans se préoccuper des moyens pour les accomplir ; c'est le cas dans notre scénario :

1. Pour offrir un meilleur service à Paul, le système détecte que la vitesse de la connexion internet est trop faible et propose une solution. La qualité de la communication ici est un élément important pour le choix de service, ainsi que l'activité (expédition d'un courriel contenant des documents volumineux) ;
2. Le type de service dépend dans notre scénario de la localisation de l'utilisateur (Signalisation des appels pour le téléphone cellulaire avec mode sonnerie ou mode vibreur) ainsi que le service lui-même (alerte par bip sonore ou par SMS) ;
3. La communication entre les équipements est aussi présente dans notre scénario. À l'entrée dans la bibliothèque, le cellulaire est mis automatiquement en mode vibreur (ou le cellulaire est équipé d'un système GPS et il est programmé pour changer de mode dans la région de la bibliothèque ou le système de surveillance de la bibliothèque envoie des requêtes aux cellulaires à l'entrée dans la bibliothèque pour changer de mode et ces derniers sont programmés pour accepter ces requêtes) ;
4. De même, la communication entre le cellulaire et l'ordinateur portatif (le cellulaire en mode vibreur à l'intérieur de la bibliothèque est susceptible d'alerter Paul du temps restant pour le prochain cours. Il communique alors avec son ordinateur portatif pour voir s'il est en train de l'utiliser. Si c'est le cas il envoie une alerte SMS) ;
5. La stratégie générale de découverte d'activité utilisée par les systèmes informatiques diffus consiste à la lecture d'une base de données (le cas de l'alerte du cellulaire pour indiquer le temps restant pour le prochain cours. L'emploi du temps de Paul est inscrit dans l'agenda interne du cellulaire qui constitue une source d'informations).

1.5 Equipement pour environnement Ubiquitaire

Il existe plusieurs équipements pour environnement ubiquitaire :

1.5.1 Les réseaux sans fil et filaires

Actuellement différents types de réseaux sans fil sont employés dans l'informatique ubiquitaire, on distingue plusieurs catégories :[4]

- **Les réseaux personnels sans fil WPAN (Wireless Personal Area Network) :** Plusieurs technologies sont utilisées pour les WPAN dont la principale est la technologie Bluetooth (IEEE 802.15.1) proposant un débit théorique de 1Mbps pour une portée maximale d'une trentaine de mètres et possède l'avantage d'être très peu gourmande en énergie.
- **Les réseaux locaux sans fil WLAN (Wireless Local Area Network) :** Plusieurs technologies concurrentes existent dont le wifi (IEEE 802.11) qui offre un débit jusqu'à 54Mbps sur une distance de plusieurs centaines de mètres.
- **Les réseaux métropolitains sans fil WMAN (Wireless Metropolitan Area Network) :**
  Ce type de réseaux se basent sur la norme IEEE 802.16 offre un débit utile de 1 à 10Mbit/s pour une portée de 4 à 10 kilomètres. La norme de réseau métropolitain sans fil la plus connue est le WiMAX, permettant d'obtenir des débits de l'ordre de 70 Mbit/s.
- **Les réseaux étendus sans fil WWAN (Wireless Wide Area Network) :**
  Ils sont connus sous le nom réseaux cellulaires sans fil. Il s'agit des réseaux de la téléphonie mobile qui incluent les technologies GSM (Global System For Mobile Communication), GPRS (General Packet Radio System), UMTS (Universal Mobile Télécommunication System).
  Pour les réseaux filaires ils offrent un débit important, comme les réseaux Ethernet, PSDN (Packet Switched Data Network) et DSL (Digital Suscriber Line).

1.5.2 Les dispositifs mobiles

Plusieurs types de dispositifs mobiles existent actuellement comme :[4]

- **Le Personal Digital Assistants (PDA) :** C'est un ordinateur de poche sous forme de boîtier compact de petite taille qui possède un écran tactile et qui est à la fois micro-ordinateur, calculatrice, agenda, réveil, téléphone, fax, ... etc
- **Smart phone :** C'est un téléphone portable couplé au PDA. Il offre en plus, des fonctions comme la navigation web, la consultation de courrier, la connectivité à un client de messagerie instantanée, la visualisation de fichiers PDF ou Microsoft office et la navigation à l'aide du système de géo localisation GPS.
- **Le Tablet PC :** Un ultra portable équipé de stylet, permettant d'écrire ou de dessiner manuellement à l'écran, comme sur un bloc-notes de format A4. Il est muni d'un système de reconnaissance de l'écriture naturelle et parfois de reconnaissances vocales.
- **Les puces RFID :** Radio Frequency IDentification : méthode utilisée pour stocker et récupérer des données à distance en utilisant des balises métalliques, les " Tag RFID". Ces balises, qui peuvent être collées ou incorporées dans des produits, et qui sont composées

d'une antenne et d'une puce électronique, réagissent aux ondes radio et transmettent des informations à distance. La puce RFID (tag ou étiquette) émet les informations contenues dans sa mémoire lorsqu'elle est en présence d'un lecteur.

1.5.3 Logiciel pour environnement Ubiquitaire

Une des problématique majeur de l'informatique diffuse est de faire communiquer de façon dynamique, spontanée et transparente les différents dispositifs de l'environnement ubiquitaire entre eux indépendamment de leurs hétérogénéité matérielle et logicielle. Les intergiciels ont été introduits dans cet objectif.

1.5.3.1 Définition

Un intergiciel (middleware en anglais) désigne la couche logicielle intermédiaire qui fournit un haut niveau d'abstraction de programmation permettant de masquer l'hétérogénéité des réseaux de communication, des ressources matérielles, des systèmes d'exploitation et des langages de programmation. Un intergiciel se situe au-dessus du système d'exploitation et en dessous des applications (c'est-à-dire des clients et/ou des services) de son hôte. Il offre à ces derniers des services de haut niveau permettant de faire des abstractions sur le système et le matériel sous-jacent accessible via des API (Applications Programming Interface) (Figure I.2), on distingue deux types d'API : des API spécifique aux systèmes d'exploitations (chaque système d'exploitation a sa propre API avec l'intergiciel) et des API standards à tous types d'applications.

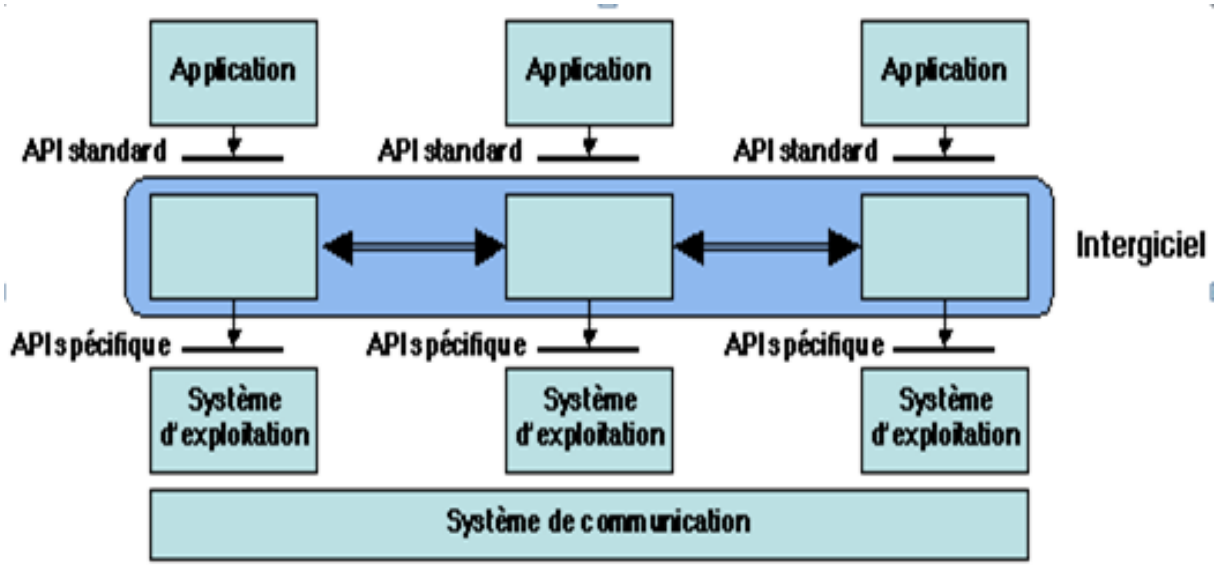


FIGURE 1.2 – Architecture de l'intergiciel [5].

L'utilisation de l'intergiciel présente plusieurs avantages dont la plupart résultent de,la capacité d'abstraction qu'il procure : cacher les détails des mécanismes de bas niveau, assurer l'indépendance vis-à-vis des langages et des plateformes, permettre de réutiliser l'expérience et parfois le code, faciliter l'évolution des applications. En conséquence, la réduction du coût et de la durée de développement des applications, l'amélioration de leur portabilité et de leur interopérabilité.

Un inconvénient potentiel est la perte de performances liée à la traversée de couches supplémentaires de logiciel.

Selon leurs rôles on distingue plusieurs types d'intergiciels parmi eux on trouve :
les intergiciels de communications et les intergiciels sensibles aux contextes.

1. **les intergiciels de communications :** Ils ont pour rôle de faciliter le développement des applications répartis et de permettre aux développeurs d'instancier des objets et invoquer des méthodes à distance sans se soucier de la plateforme système sur lequel ces objets sont exécutés, ces intergiciels sont devenus des standards, trois entre eux se distinguent : CORBA, RMI, DECOM.
2. **les intergiciels sensibles aux contextes :** Contrairement à l'informatique traditionnelle, l'informatique ubiquitaire demande aux applications d'être capable de fonctionner dans un environnement extrêmement dynamique en sollicitant le moins possible l'attention des utilisateurs. Ces exigences ont conduit à la conception d'applications dites sensibles au contexte c'est le rôle d'un intergiciel sensible au contexte.

1.6 Quelques domaines d'application de l'informatique ubiquitaire

De nos jours l'informatique ubiquitaire s'introduit dans diverses domaines parmi eux on cite :

1.6.1 Domaine publique

L'exemple le plus simple est le domaine des transports, dans ce cadre plusieurs projets ont été réalisés comme le projet STI systèmes de transport intelligent [6], qui vise à l'application des technologies de l'information et de la communication (TIC) aux transports, voire à la mobilité en général. Les STI collectent, stockent, traitent et distribuent de l'information relative à l'état d'infrastructures, à la progression de véhicules et au mouvement des personnes et des marchandises. Plusieurs grands domaines d'application sont également concernés : la sécurité / sûreté, l'exploitation, l'information et l'assistance des voyageurs

1.6.2 Domotique

La domotique désigne l'ensemble des technologies et techniques mises en uvre pour apporter des fonctions de gestion de la maison, telles que l'automatisation du fonctionnement d'appareils pour délester ses habitants de certaines tâches. Des applications ordinaires de la domotique sont les systèmes de gestion d'énergie, servant à automatiser les appareils de climatisation et d'éclairage pour le confort de l'habitant. Ces applications doivent pouvoir être paramétrées selon les préférences de l'habitant et l'état courant de l'environnement (par exemple, la température, la luminosité ambiante, ou l'heure de la journée). La domotique est généralisable à des espaces physiques plus larges, que sont les écoles, les musées, ou les bâtiments d'entreprise. Elle porte alors parfois la dénomination d'immotique [2].

1.6.3 Domaine Médical

La télémédecine est à la mode. Elle ratisse large et intègre tout ce qui permet la connexion distante entre un médecin et un patient, et même les dispositifs d'auto surveillance du patient : les environnements ubiquitaire fournissent un service médical personnalisé n'importe quand et n' importe où afin qu'un meilleur niveau de vie puisse être atteint.[7]Le groupe K. Ouchi et al [8], ont proposés un modèle conceptuel pour la construction d'un système de soin dans une voiture mobile (Smart Hospital) qui est LIFEMINDER. Ce dernier est un modèle médical pour la surveillance des changements d'états d'un patient ; il est composé d'une montre capteur (WristWatch-sharped wearable sensor) et d'un PDA. Le capteur sert à mesurer les trois axes physiologiques tel que : l'accélération du sang, le taux de la pulsation du cur et la température du corps. Le PDA sert à reconnaitre des activités du patient comme la marche et la nutrition via un réseau Bluetooth.

1.7 Caractéristiques de l'environnement ubiquitaire

Un environnement ubiquitaire exige de relever plusieurs défis, certains d'entre eux sont présentés par les points suivants :

- **Hétérogénéité**

  L'hétérogénéité peut être au niveau des composants logiciels, au niveau des architectures matérielles ou au niveau de l'infrastructure de communication. En effet l'environnement ubiquitaire est peuplé d'une multitude de services logiciels développés dans des langages de programmation différents tels que Java, C ou Python etc. et déployés sur des plates-formes diverses, ainsi que des dispositifs matériels (ordinateur portable, PDA, smart phone, etc.) hétérogènes. Ces dispositifs incluent les capteurs et les effecteurs, les dispositifs embarqués dans les objets, les appareils. Cela nécessite une infrastructure qui, maintient des

connaissances sur les caractéristiques des dispositifs et gère l'intégration des dispositifs dans un système cohérent qui permet des interactions entre ces dispositifs. La communication entre ces dispositifs est assurée par une infrastructure réseau hybride, contenant aussi bien des connexions filaires, chaque dispositif a des capacités et une configuration matérielle différente et éventuellement différentes plateforme logicielles pour l'exécution des programmes. Ces dispositifs doivent interagir entre eux d'une manière transparente malgré une grande différence entre leurs capacités matériels et logiciels. que des connexions sans fils. Cette infrastructure réseau hybride impose de nombreuses contraintes à cause du caractère très dynamique de ces réseaux (connexions / déconnexions), du changement fréquent au niveau de leurs topologies, et de l'hétérogénéité des dispositifs et des protocoles utilisés [9].

- **La distribution**

  L'informatique ubiquitaire est de nature distribuée. Plusieurs intergiciels (middleware) de communication tels que CORBA, les services web offrant une transparence de la distribution des communications ont été développé. Cependant ces intergiciels ne prennent pas en compte la mobilité, la synchronisation et la coordination[9].

- **Contrainte des ressources**

  Les utilisateurs qui entrent dans ces environnements sont généralement munis de terminaux mobiles et veulent réaliser des tâches en utilisant les dispositifs et services disponibles dans l'environnement. Ces dispositifs sont non seulement de petite taille, mais ils sont aussi caractérisés par des ressources limitées en termes de capacité de stockage, affichage, puissance de calcul, bande passante, et ressources énergétiques.

  Par exemple le PDA a un petit écran et une batterie limitée ; le téléphone portable a aussi un petit écran mais une batterie de plus longue durée de vie. De toute évidence ces limitations influent le développement des applications et leurs capacités. Les applications doivent donc s'adapter aux changements des capacités matérielles et la variabilité des services logiciels disponibles [9].

- **Limitation réseaux**

  Les connexions sans fil utilisées dans les dispositifs mobiles ont généralement une bande passante limitée et instable. A présent, les technologies standard sans fils incluent le Bluetooth, Wifi, GPRS, IEEE 802.11a, IEEE 802.11b, IEEE 802.11g et d'autres en cours de développement. La vitesse lente de ces technologies par rapport au réseau filaire a été toujours une barrière pour exécuter de grandes applications et déplacer des données volumineuses sur les mobiles [9].

- **Mobilité**

  La mobilité est un atout pour les systèmes pervasifs mais elle soulève néanmoins des problèmes. Le principe des systèmes pervasifs réside sur la collaboration des services et

donc des dispositifs qui les hébergent. La mobilité des dispositifs entraine des variations de connexion qui peuvent avoir de lourdes conséquences sur le fonctionnement des services : faible débit qui entraine la lenteur des transmissions et parfois la déconnexion, tout comme l'épuisement d'une batterie, les déconnexions provoquent, au moins de manière temporaire l'indisponibilité des services et par conséquent dégradent la qualité de service de l'application.

- **Variation du contexte**

  L'utilisateur et son périphérique évoluent dans un environnement changeant sans cesse. L'utilisateur modifie ses préférences, les ressources du périphérique évoluent tout comme l'environnement physique. Le fonctionnement des applications peut être perturbé par des modifications du contexte. Par exemple, dans une application de visioconférence, la luminosité et le bruit ambiant peuvent perturber son fonctionnement et se répercuter sur la satisfaction de l'utilisateur et sur l'utilisabilité de l'application, c'est-à-dire modifier la qualité de service perçue par ce dernier.

1.8 Conclusion

L'assimilation et la compréhension de tout travail de recherche devront passer par une description de l'ensemble des concepts clés. Ainsi nous avons essayé dans ce chapitre de recenser les concepts de base de l'informatique ubiquitaire et de les définir d'une manière à nous servir dans les chapitres ultérieurs.

Plusieurs problématiques peuvent être posées dans l'étude de l'informatique ubiquitaire. Le chapitre suivant traitera l'une des plus importantes problématique qui est " la modélisation hybride des informations de contexte dans l'environnement ubiquitaire ".

CHAPITRE 2

L'ÉTAT DE L'ART SUR LA MODÉLISATION DES INFORMATIONS CONTEXTUELLES

2.1 Introduction

Par rapport à l'informatique traditionnelle, l'informatique ubiquitaire introduit de nouveau chalenges pour la conception des applications. En particulier, l'informatique ubiquitaire demande aux applications d'être capable de fonctionner dans un environnement extrêmement dynamique en sollicitant le moins possible l'attention des utilisateurs. Ces exigences ont conduit à la conception d'applications dites sensibles au contexte. Ces applications doivent détecter les variations de l'environnement, tels que la localisation et l'identité des utilisateurs et des objets, et adapter leurs comportements en conséquence. Cela implique que dans différents contextes, les utilisateurs accèdent aux même données et aux mêmes services mais reçoivent des réponses qui peuvent être différentes ou présentées différemment ou à des niveaux de détails différents. Les caractéristiques variées des informations de contexte comme leur hétérogénéité, leur dynamicité, leur imperfection et leur complexité nécessitent des modèles abstraits de description de contexte pour simplifier leur utilisation. Plusieurs travaux ont défini des modèles de description contexte, ces modèles varient selon leur expressivité et le type d'informations de contexte qu'ils permettent de décrire. Nous cherchons à définir des modèles qui ont un haut pouvoir expressif, pouvant supporter le raisonnement sur le contexte, et ont de bonnes performance dans le raisonnement.

2.2 Notion du contexte

2.2.1 Définition du contexte

Le contexte étant une notion complexe et abstraite, il est difficile de trouver une définition détaillée. Les définitions générales sont les plus nombreuses ; nous listons ici les principales [10].Parmi les premiers à essayer de définir le contexte, se trouvent Schilit et Theimer, pour lesquels le contexte est constitué de la localisation de l'utilisateur, ainsi que des identités et des états des personnes et des objets qui l'entourent [11]. Brown et al. [12] ajoutent à cette définition des données telles que l'identité de l'utilisateur, son orientation ou la température. Ryan et al. [13] ajoutent la notion de temps.

Pascoe [14] introduit un élément important : l'intérêt. En effet, il définit le contexte comme un sous-ensemble d'états physiques et conceptuels qui ont un certain intérêt pour une entité donnée. Cette notion d'intérêt ou de pertinence est reprise par Abowd, Dey et al. [15] dans leur définition, qui est communément acceptée :

" Le contexte couvre toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet que l'on considère pertinent par rapport à l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes. "

Winograd [16] reprend cette définition pour la détailler, car il considère que, malgré le fait qu'elle couvre tous les travaux existants, elle est trop générale : tout élément peut être considéré comme faisant partie du contexte. En premier lieu, il précise que le contexte est un ensemble d'informations. Cet ensemble est structuré et partagé, et il peut évoluer dans le temps. En deuxième lieu, selon lui, l'appartenance d'une information au contexte ne dépend pas de ses propriétés inhérentes, mais de la manière dont elle est utilisée. Une information fait partie du contexte seulement si le système dépend d'elle, d'une façon ou d'une autre.

Malgré le grand nombre de définitions existantes et les similarités (la plupart font références à la localisation et l'environnement), le mot contexte reste toujours général. Deux techniques sont utilisées par les chercheurs pour la définition du contexte : l'une est basée sur l'énumération des exemples du contexte et l'autre fait plutôt des tentatives en vue de formaliser le terme. L'importance du contexte dans le domaine de l'interaction personne machine et les systèmes mobiles a généré des définitions centrées sur l'utilisateur et d'autre sur l'application. Une analyse faite par Brezillon et al.[17] concernant les définitions du terme contexte les a conduits à conclure que la plupart des définitions sont des réponses aux questions suivantes :

- **qui ?** identité de l'utilisateur courant et d'autres personnes présentes dans l'environnement ;
- **quoi ?** percevoir et interpréter l'activité de l'usager ;

- **où ?** localisation de l'usager, ou d'un événement du système ;
- **quand ?** repère temporel d'une activité, indexation temporelle d'un événement, temp écoulé de la présence d'un sujet à un point donné ;
- **pourquoi ?** il s'agit de comprendre la raison d'être de l'activité ;
- **comment ?** la manière de déroulement de l'activité.

Les réponses aux questions citées ci-dessus peuvent engendrer un grand ensemble d'informations dont une grande partie est inutile. Cela requiert également un plus grand effort pour la gestion de ces informations.

2.2.2 Définition du contexte pertinent

Un contexte pertinent pour une application est un sous ensemble du contexte observé dont les changements de valeur affectent cette application. En d'autre termes l'ensemble des contextes observables sélectionnés pour une application donnée du fait que cette dernière est sensible à leurs changements de valeur est appelé contexte pertinent.

2.2.3 Caractéristiques des informations de contexte

Les informations contextuelles sont des données détectées automatiquement par le système, collectées lors des interactions entre l'utilisateur et le système, ou bien déduites ou générées dynamiquement par le système. Elles sont collectées à partir de plusieurs sources hétérogènes et distribuées. Ces sources de contexte sont variées et différent en terme du type de l'information perçue (texte, image, son, vidéo,etc.), ainsi qu'en terme de capacité et de qualité de perception. De plus, un même type d'information peut être fournis par plusieurs sources différentes. Par exemple, la localisation d'un individu peut être donnée par un GPS, un GSM, une caméra, un badge IR, etc. De ce fait, elles ont des caractéristiques variables que ce soit au niveau modélisation, ou bien au niveau gestion et qualité. Chaque contexte observable peut être statique ou dynamique. Les observables statiques, représentent des informations qui ne changent pas au cours du temps. Il suffit de les collecter au lancement de l'application. Parmi les observables statiques, nous avons la taille de l'écran ou le profil d'un utilisateur. Les observables dynamiques représentent des informations dont les valeurs changent fréquemment, une observation de leur état peut devenir très rapidement obsolète. Les observations de contexte effectuées à partir de capteurs physiques peuvent être sujettes à des bruitages ou des erreurs de capture. Donc, ces observations sont incorrectes lorsqu'elles ne reflètent pas l'état réel de l'environnement, incohérentes lorsqu'elles contiennent des informations contradictoires et incomplètes lorsque certains aspects du contexte ne sont pas renseignés.

2.2.4 Catégorisation du contexte

La sensibilité au contexte exige que les informations contextuelles soient collectées et présentées à l'application d'adaptation. Vu l'hétérogénéité, la diversité et la qualité de ces informations, il est désirable de faire une classification ou bien une catégorisation pour faciliter l'opération d'adaptation. Dans ce domaine précis, plusieurs chercheurs ont proposé des catégorisations selon différentes approches.

Schilit et al.[18] et Dey.[19] ont catégorisé le contexte en deux classes : le contexte primaire qui contient les informations sur la localisation, l'identité, le temps et l'activité (statut) ; le contexte secondaire qui peut être déduit de ce dernier (exemple : de la localisation, on peut déduire les personnes à proximité).

Chen et Kotz .[20] ont proposé deux catégories : le contexte actif qui influence le comportement d'une application et le contexte passif qui est nécessaire mais pas critique pour l'application.

Petrelli et al.[21] ont fait connaître le contexte matériel (localisation, machine, plateforme existante) et le contexte social (les aspects sociaux comme la relation entre les individus).

Gwizdka.[22] a présenté deux catégories : le contexte interne contenant l'état de l'utilisateur et le contexte externe englobant l'état de l'environnement.

Hofer et al.[23] ont élaboré une catégorisation en deux classes : le contexte physique qui peut être mesuré par les capteurs physiques et le contexte logique qui contient les informations sur l'interaction (l'état émotionnel de l'utilisateur, ses buts, etc.)

Razzaque et al.[24] ont proposé comme suit une catégorisation en six classes :

- **Contexte utilisateur :** il permet d'obtenir les informations sur les utilisateurs du système informatique. Le profil de l'utilisateur par exemple en fait partie. Il peut contenir des informations sur son identification, ses relations avec les autres usagers, la liste de ses tâches, et d'autres ;
- **Contexte physique :** il offre la possibilité d'intégrer des informations relatives à l'environnement physique, telles que la localisation, l'humidité, la température, le niveau de bruit, etc. ;
- **Contexte du réseau :** il fournit aussi des informations de l'environnement, mais ces dernières se rapportent essentiellement au réseau informatique. Exemple : connectivité, bande passante, protocole, etc. ;
- **Contexte d'activité :** répertorie les évènements qui se sont déroulés dans l'environnement ainsi que leur estampille temporelle. Exemple d'évènements : entrée d'une personne, tempête de neige, etc.
- **Contexte matériel :** il permet d'identifier les appareils de l'environnement qui peuvent être utilisés. Il inclut par exemple le profil et les activités des dispositifs de l'environnement

(identification, localisation, niveau de la batterie, etc.)

- **Contexte de service :** il informe sur ce qui peut être obtenu. Par exemple : les informations relatives aux fonctionnalités que le système peut offrir.

Une autre catégorisation proposée par les mêmes auteurs mais fondée sur les valeurs que peut prendre une information contextuelle. Il y a :

- **Le contexte continu :** dans cette catégorie, les valeurs varient continuellement. Un élément d'un contexte continu est fonction de différents paramètres et sa valeur est calculée en se servant d'une formule. Exemple : les informations GPS ;
- **Le contexte énumératif :** les valeurs d'un composant du contexte sont un ensemble discret de valeurs ;
- **Information contextuelle d'état :** les éléments de cette catégorie peuvent prendre deux valeurs opposées. Par exemple : la lumière dans une pièce peut être allumée ou éteinte. Les valeurs de ces éléments sont obtenues à partir d'un calcul de prédicat ;
- **Le contexte descriptif :** il est basé sur les descriptions des éléments du contexte.

Il existe bien d'autres catégorisations qu'ont été proposées que celles qui ont été présentées ici, mais aucune d'elles ne se veut exhaustive. De nouveaux regroupements seront effectués à mesure que de nouvelles caractéristiques des informations contextuelles seront découvertes.

Il n'en demeure pas moins que ces efforts de classification sont louables et permettent aux développeurs de l'informatique diffuse de manipuler plus efficacement les informations contextuelles.

2.2.5 Acquisition du contexte

L'acquisition des informations de contexte peut se faire selon plusieurs méthodes, indépendamment de l'application. Mostefaoui [25] a classé ces méthodes en trois catégories :

- **acquisition par profil :** cette méthode consiste à récupérer des informations soit à travers une interface graphique soit par l'intermédiaire d'un fichier de profil .
- **acquisition par sonde :** cette méthode consiste à utiliser des sondes (capteurs) pour récupérer les informations de contexte. Schmidt et al.[26] font la distinction entre deux types de sondes : les sondes physiques et les sondes logiques. Les sondes physiques sont des composants électroniques qui permettent de mesurer des paramètres physiques dans l'environnement comme la température et la localisation, alors que les sondes logiques utilisent des logiciels pour récupérer des informations associées à l'hôte où s'exécute l'application comme la bande passante et la charge de la batterie. Le contexte collecté à l'aide de sondes est dynamique, il est donc nécessaire de mettre à jour les observations fréquemment .
- **acquisition par dérivation** la dérivation ou l'interprétation du contexte consiste à

utiliser un ou plusieurs observables pour déduire ou calculer à la volée un contexte de plus haut niveau en utilisant des méthodes d'interprétation. Par exemple, la rue où se trouve une personne est un contexte de haut niveau calculé à partir des données de localisation en latitude et longitude collectées par un capteur GPS. La méthode d'acquisition d'un observable dépend en grande partie de l'observable.

2.3 Sensibilité au contexte

L'idée de sensibilité au contexte émerge naturellement du concept d'informatique ubiquitaire. En effet, le fait de disposer d'informations contextuelles sert à adapter l'application au contexte perçu afin de mieux répondre aux attentes des utilisateurs.

La notion de sensibilité au contexte (context-awareness en anglais) a donc été produite dans le cadre des recherches sur l'informatique ubiquitaire. Schilit et Theimer [11] furent les premiers à proposer une définition de la sensibilité au contexte : il s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve. Ils signalent également l'importance de l'adaptation du système à ces changements.

Pour Abowd, Dey et al.[15] , un système est sensible au contexte s'il utilise celui-ci pour fournir à l'utilisateur des informations et des services pertinents. Cette pertinence dépend de l'activité que l'utilisateur est en train de réaliser. Ces auteurs proposent également une classification des systèmes sensibles au contexte selon leur réponse aux changements de contexte. Cette réponse peut soit présenter des informations ou des services à l'utilisateur, soit exécuter automatiquement un service, soit stocker des données contextuelles.

2.4 Modélisation des informations de contexte

Pour décrire la sensibilité d'une application à son contexte d'exécution, il faut déterminer les contextes auxquels cette application est sensible et les décrire dans un modèle. Par conséquent, la modélisation du contexte est la première étape dans le processus de création d'applications sensibles au contexte. Cette modélisation permet à l'application et/ou aux intergiciels de faciliter l'interaction avec le contexte en fournissant une description abstraite des observables. La diversité des informations de contexte et leur utilisation dans divers domaines engendrent différentes façons de les modéliser. Dans ce qui suit, nous classifions la modélisation du contexte en quatre types d'approches : l'approche paires/triplets, l'approche orientée modèles, l'approche basée sur la logique et l'approche orientée ontologie. Cette classification se base sur la façon dont le contexte est modélisé, les outils utilisés à cet effet, l'expressivité du modèle et la possibilité de le réutiliser. Plusieurs conditions doivent être prises en compte lors de la modélisation

des informations contextuelle, Bittini et al.[27] ont présenté certains d'entre eux par les points suivants :

1. **L'hétérogénéité et la mobilité :** Les modèles d'information sur le contexte doivent être capables de restituer une grande variété de sources d'information sur le contexte qui diffèrent dans leur taux de mise à jour et leur niveau sémantique. Un modèle de contexte devrait être en mesure d'exprimer les différents types d'informations sur le contexte. De plus, le système de gestion de contexte doit assurer la gestion de l'information sur le contexte en fonction de son type. Beaucoup d'applications sensibles au contexte sont aussi mobiles (en cours d'exécution sur un appareil mobile) ou dépendent de sources d'information mobiles (par exemple, des capteurs mobiles). Cela ajoute au problème de l'hétérogénéité. En effet, l'emplacement et la disposition spatiale des informations sur le contexte jouent un rôle important en raison de l'exigence de mobilité.
2. **Les relations et les dépendances :** Il existe des relations entre les différents types d'informations sur le contexte qui doivent être acquises pour assurer un comportement correct des applications. Les relations correspondent à la dépendance par laquelle les entités tirées du contexte peuvent dépendre d'autres entités : par exemple, un changement de la valeur d'une propriété (e.g. la bande passante réseau) peut affecter les valeurs d'autres propriétés (e.g. reste de la puissance de la batterie).
3. **L'historique des informations sur le contexte :** Les applications sensibles au contexte doivent avoir accès à l'état passé et à l'état futur. Par conséquent, l'historique des informations sur le contexte est une autre caractéristique qui doit être saisie par les modèles de contexte et gérée par le système de gestion de contexte. La gestion de l'historique des informations sur le contexte est difficile si le nombre de mises à jour est très élevé. Il est alors impossible de stocker toutes ces valeurs, en conséquence la technique de summarisation (condensation) doit être appliquée (exemple : l'agrégation des mises à jour de position à une fonction de mouvement en utilisant des techniques d'interpolation, ou des synthèses sur l'historiques des données).
4. **L'imperfection :** Due à leurs natures dynamiques et hétérogènes, les informations du contexte peuvent être d'une qualité variable ou même incorrectes. La plupart des capteurs comportent une inexactitude inhérente (par exemple, de quelques mètres pour des positions de GPS). Les informations de contexte peuvent être incomplètes ou en conflit avec d'autres informations de contexte, ainsi une bonne approche de modélisation de contexte doit prendre en compte le paramètre de qualité des informations de contexte.
5. **Raisonnement :** Les applications sensibles au contexte utilisent les informations de contexte pour évaluer s'il y a un changement de contexte de l'utilisateur et/ou des dispositifs dans l'environnement ; prenant des décisions si une adaptation à ces changements

est nécessaire qui souvent requièrent des capacités de raisonnement.

6. **Flexibilité :** La capacité du modèle de s'adapter à différents contexte.
7. **Granularité des variables du contexte :** C'est la capacité du modèle de représenter les caractéristiques de contexte à différents niveaux de détails.
8. **Ambiguïté et gestion incomplète :** Dans le cas où le système perçoit une ambiguïté ou une incohérence des informations de contexte ou encore une information incomplète, le système doit interpoler ou modifier d'une façon ou d'une autre ces informations et construire un contexte courant raisonnable (approprié à la situation courante de l'utilisateur et de l'environnement).
9. **Efficacité de stockage du contexte :** L'accès efficace à l'information de contexte est nécessaire. Les caractéristiques variées des informations de contexte nécessitent des modèles de description abstraits. Une fois les informations de contexte collectées, la première démarche de création d'une application sensible au contexte est la modélisation des informations de contexte. Dans ce qui suit nous allons présenter les différentes approches de modélisation de contexte.

2.4.1 Approches de modélisation de contexte

Plusieurs approches de modélisation de contexte existent dans la littérature, après une analyse de ces différentes approches nous les avions classifiés comme suit :

1. **Approche clé-valeur [28]**

   Le modèle clés- valeurs utilise la paire clés-valeurs pour définir la liste des attributs et leurs valeurs décrivant les informations de contexte, par exemple le contexte1 est définit par l'utilisateur 'x' qui est localisé dans un emplacement 'y' à un temps 't' est modélisé comme suit : " Name = 'contexte1', User = 'docteur kamran', localisation ='Hôpital Khelil Amrane', temps = 'lundi 28 mars 2012 16 :41 :29' ".

   Avantages et inconvénients

   La modélisation clés-valeurs utilise des structures de données simples à gérer. Par contre, cela ne permet pas une description complète du contexte, ni l'expression des relations qui peuvent exister entre les informations de contexte.

2. **Modélisation orientée objet**

   Henriksen et al.[29] ont proposé un ensemble de concepts de modélisation basés sur une approche orientée objet. Dans cette approche, les informations de contexte sont egroupées en un ensemble d'entités. Chaque entité représente un objet conceptuel ou physique tel qu'une personne, un dispositif ou un réseau. Les propriétés des entités telles que le nom

d'une personne sont représentées par des attributs. Les entités sont liées à leurs attributs à travers des associations. Voici quelques exemples de modèles orientés objet.

- **Modèle basé sur UML (Unified Modeling Language)**
  La généricité du langage UML en fait un langage approprié pour modéliser le contexte. Sheng et Benatallah [30] ont proposé un méta-modèle basé sur une extension d'UML qui permet de modéliser le contexte auquel des services web sont sensibles. Ce langage est appelé ContextUML.
- **Modèle basé sur un langage de balise**
  Le point commun entre toutes les approches qui utilisent un langage de balises est la structure hiérarchique des données modélisées. Cette modélisation consiste en un ensemble de balises avec des attributs. Dans XML (eXtensible Markup Language), les balises sont définies dans une DTD (Document Type Definition) et dans CC/PP (Composit Capability/Prefrence Profile) [31] qui est la proposition du W3C pour la représentation de profils. C'est un cadriciel basé sur RDF (Resource Description Framework) [32] pour stocker des paires clé-valeur avec les balises appropriées. CC/PP permet de décrire les capacités d'un dispositif ainsi que les préférences de l'utilisateur en utilisant une structure de profils. Le vocabulaire offert par CC/PP n'est pas riche et a besoin d'être étendu car il est restreint à la description de profil. De plus, il ne permet pas la description de relations et de contraintes complexes entre les informations de contexte. Les travaux de Indulska et al. [33] ont étendu le vocabulaire de CC/PP pour pouvoir décrire la localisation, les caractéristiques du réseau et les dépendances d'une application. Ainsi, cette modélisation peut être utilisée pour décrire les contextes observables associés à une application sensible au contexte, mais Indulska et al. ont conclu que malgré cette extension, cette modélisation reste non intuitive et difficile à utiliser pour décrire des informations complexes [33].
- **Modèle basé sur CML (Context Modeling Langage)**
  Afin de pouvoir modéliser les caractéristiques des informations de contexte et leurs propriétés, Henricksen et al.[34] ont proposé un langage orienté objet appelé CML (Context Modeling Language) qui permet de modéliser le contexte auquel une application est sensible d'une manière formelle. Un outil graphique assiste le concepteur d'applications dans la tâche de description du contexte auquel son application est sensible. Il lui offre un moyen de décrire les caractéristiques des informations de contexte (capturé,statique, dérivé ou information de profile) et les dépendances entre ces informations. CML permet aussi de spécifier la qualité de chaque information observée et sa validité temporelle.

Avantages et inconvénients

Ce type d'approche permet de décrire le contexte de manière plus riche que les approches

clés/valeurs. De plus, cela offre aux concepteurs d'applications la possibilité de réutiliser le modèle pour d'autres applications.

Les modélisations basées sur un langage de balises permettent de décrire des profils ou des contextes simples, sans offrir la possibilité de décrire des relations de dérivation et de dépendance entre ces informations.

Les modélisations existantes basées sur UML permettent de décrire des relations entre les informations de contexte mais ne prennent pas en compte la description des dépendances entre ces informations, ni la qualité ou la validité temporelle des données décrites,

CML est venu remédier à certains de ces manques en proposant un modèle avec visualisation graphique qui permet de typer le contexte, et de décrire un ensemble de relations entre plusieurs contextes observables.

3. **Modèles basés sur la logique**

   Les modèles basés sur la logique sont caractérisés par un très grand degré de formalité. Pour les modèles classiques ils utilisent l'algèbre booléenne et la logique du premier ordre pour modéliser le contexte. La logique permet de définir des conditions qui nécessitent de déduire des faits ou des expressions à partir d'un autre ensemble d'expressions ou de faits. Par conséquent, dans les modèles basés sur la logique, le contexte est défini comme des faits, des expressions ou des règles. Il existe aussi les modèles récents basés sur la logique comme : Event Calculs ,Situation Calculs et Fluent Calculs.

   Avantages et inconvénients

   Cette modélisation ne permet pas de décrire la validité temporelle des informations ni les relations qui peuvent exister entre les informations de contexte, mais elle est très efficace pour raisonner sur le contexte et déduire des actions de réaction si une situation pertinente est détectée. L'approche basée sur la logique peut être utilisée dans l'informatique sensible au contexte afin d'intégrer et d'interpréter les données collectées.

4. **Modèles basés sur les ontologies**

   Une ontologie est une description sémantique, structurée et formelle des concepts d'un domaine et de leurs interrelations. Les auteurs dans [35] justifient l'utilisation des ontologies par trois arguments :

   - **U**ne ontologie permet de partager les connaissances dans un système distribué.
   - **U**ne ontologie comprend des sémantiques déclaratives permettant d'élaborer les raisonnements sur les informations de contexte.
   - **A**vec une représentation explicite d'une ontologie commune, l'interopérabilité des applications et des terminaux est assuré.

   Plusieurs modèles d'ontologies ont été proposés pour décrire le contexte. Ces approches permettent non seulement de modéliser le contexte, mais aussi de raisonner sur les don-

nées décrites. Les approches orientées ontologies sont caractérisées par une possibilité d'extension et de partage des données.

De nombreux langages informatiques sont apparus pour construire et manipuler des ontologies.Dans le but de mettre au point un langage standardisé, le W3C a créé en novembre 2001un groupe de travail qui a abouti à la recommandation OWL (Ontology Web Language) en février 2004 [38].

Avantages et inconvénients

L'avantage de ces approches réside dans les caractéristiques même de l'ontologie, qui offrent non seulement le moyen de faire des descriptions sémantiques, mais aussi de publier les données décrites à travers le réseau. Cependant ces modèles restent souvent complexes à implémenter dans des cas réels.

Synthèse

Après avoir étudié ces différentes approches on a aboutie au bilan suivant (table 2.1) :

Les approches clés-valeurs sont caractérisées par une pauvreté d'expression et la simplicité des données qu'elles représentent. Elles sont basées sur une description par un tuple (clé/valeur). De ce fait, elles ne permettent pas le raisonnement sur les informations contextuelles ni la description des relations existantes entre ces informations, également les données décrites ne sont pas publiées. Ainsi dans le cas de développement d'une grande application évoluant dans un environnement dynamique comme l'environnement ubiquitaire (caractérisé par une très grande variation de contexte) l'utilisation de cette approche n'est pas appropriée.

Les approches orientées objets sont caractérisées par leurs possibilité de réutilisation du fait qu'elles utilisent des modèles formels pour la description de contexte et qu'elles permettent la description d'un méta-modèle qui peut être réutilisé par plusieurs applications ou intergiciels. Les approches orienté objet existantes ne permettent pas le raisonnement sur les informations contextuelles ni la publication de ces dernières, cependant il reste souvent un modèle très répandu vu la possibilité d'intégré des mécanismes pour remédier à ses insuffisances.

Les approches basées sur la logique permettent de raisonner sur les informations de contexte pour déduire de nouvelles valeurs du contexte ou pour lancer des réactions au niveau de l'application ou du système. Les approches appartenant à cette catégorie permettent de décrire des relations entre les informations de contexte, mais leurs applications aux systèmes existants restent limitées ; cela est dû à la difficulté de description qu'engendre ce type d'approche. Ces approches ne permettent pas également de publier l'information collectée.

Les approches orientée ontologies quant à elle bien qu'elles sont caractérisées par leurs difficulté de mise en oeuvre, elles sont très répandu à l'environnement ubiquitaire du fait qu'elles répondent pratiquement à toutes les exigences. En effet son utilisation de moteur d'inférence lui permet de décrire des relations entre les informations du contexte et un raisonnement sur ces dernières et surtout leurs publications qui facilitent à l'application de les utiliser.

Modèles \ conditions	**Attribut \ valeur**	**Basée sur la logique**	**Modélisation orientée modèle**	**Orientée ontologies**
Caractéristiques	Simplicité d'utilisation, pauvreté d'expression	Très formelle	Utilisation d'un méta-modèle	Utilisation d'un moteur d'inférence
Relations et dépendance	Non	Non	Oui	Oui
Raisonnement	Non	Oui	Non	Oui
Réutilisation et extension du Modèle	Non	Non	Oui	Oui
Publication des données décrites	Non	Non	Non	Oui

TABLE 2.1 – Synthèse de différentes approches de modélisation de contexte

2.5 Pourquoi la modélisation hybride ?

Chacune des approches de modélisation présentés précédemment peut fournir une solution efficace pour un domaine particulier, et / ou pour un type particulier de raisonnement,mais aucun d'eux ne semble apporter une solution au problème général, de l'acquisition de données à partir de capteurs à la livraison des applications de haut niveau de contexte des données. De même, aucun d'eux ne peut simultanément satisfaire à toutes les exigences de la modélisation des informations contextuelles.

Les considérations ci-dessus semblent suggérer que les différents modèles et outils de raisonnement doivent être intégrés les uns aux autres. Même si une expression unique du langage de représentation qui remplie la plupart des besoins identifiés pourraient probablement être défini, il ya de fortes indications que la complexité qui en résulte le raisonnement, sera inutile dans des scénarios réels. Dans le domaine de la représentation de connaissance, une approche alternative à l'utilisation d'un formalisme très expressif simple a été identifiée dans des formalismes hybrides de représentation des connaissances ; c-à-d, formalismes composés par des différents sous-langages de modélisation pour représenter différents types de connaissance. Un des avantages de tels formalismes est que la complexité du raisonnement hybride n'est généralement pas plus mauvaise que la complexité du raisonnement avec les sous-langages simples[27]. Dans ce qui suit nous présentons des différentes approches hybrides proposées pour la représentation informations contextuelles et le raisonnement sur ces dernières.

2.6 Les approches hybrides proposées

2.6.1 Modèle hybride fact-based/ontologique

Henricksen et autres. [36][27]proposent une approche hybride de la modélisation du contexte, les ontologies combinant avec le fact-based qui est une approche fourni par le modèle CML.L'objectif est de combiner les avantages particuliers des modèles CML, en particulier le traitement des informations de contexte imparfait et la description des dépendances entre ces informations avec le soutien de l'interopérabilité et de divers types de raisonnement des modèles ontologiques. Cette approche hybride est basée sur une cartographie de structures de modélisation CML pour les classes et les relations d'OWL-DL.

Discussion et critique

Il est intéressant de noter que, en raison de certaines limitations d'expressivité de OWL-DL, une cartographie complète entre le modèle CML et le modèle OWL-DL ne peut pas être obtenue. En ce qui concerne les questions d'interopérabilité, les avantages acquis par le modèle ontologique de représentation de contexte sont clairement reconnaissables. Cependant, par rapport à la dérivation de nouvelles données de contexte, les expériences avec le projet de modèle hybride ont montré que le raisonnement ontologique avec OWL-DL et son extension SWRL n'a apporté aucun avantage par rapport au raisonnement avec le modèle CML. Pour cette raison, le raisonnement ontologique est effectué que pour la vérification automatique de la cohérence du modèle de contexte, et pour la cartographie sémantique des différents modèles contextuels.

2.6.2 Modèle hybride markup-based(à base de balisage)/ontologique

Bettini et autres.[37][39][27] proposent une approche hybride de la modélisation du contexte, qui est basé sur une interaction souple entre un modèle de balisage étendu à la politique des règles exprimées dans un langage de programmation logique restreinte et les modèles ontologiques. L'interaction entre ces modèles est réalisée à travers la représentation des données de contexte par l'intermédiaire du modèle logique CC / PP qui contiennent une référence au classes et les relations de OWL-DL. Afin de préserver l'efficacité, le raisonnement ontologique est principalement réalisé à l'avance. Chaque fois que de nouvelles données de contexte pertinentes sont acquises, le raisonnement ontologique est démarré, et l'information dérivée est utilisée, si elle est encore valide au moment de service d'approvisionnement avec évaluation des règles efficaces. Les données du contexte complexe (par exemple, l'activité actuelle de l'utilisateur) dérivée par un raisonnement ontologique peuvent être utilisées dans les conditions préalables de la règle pour dériver des données de nouveau contexte tel que préférences de l'utilisateur. A titre d'exemple, considérons la règle suivante : hasCurrActivity * (x, Réunion de travail) ? hasAvailState (x, Occupée). La condition préalable de la règle concerne les données contextuelles

complexes identifiés par une étoile le symbole qui représente l'activité courante d'une instance x d'un individu (dans ce cas, l'utilisateur en cours).Le raisonnement ontologique est effectué de manière asynchrone pour tous les prédicats qui peuvent être utiles dans les règles. Dans ce cas, il est utilisé pour identifier les classes connues des activités des utilisateurs et pour mieux s'adapter à la disposition des données de contexte pour l'utilisateur x ; dans cet exemple, si Réunion de travail est identifié par ce processus, le moteur de règles en découle que l'état de disponibilité du courant utilisateur est occupé. Notez qu'il peut être le cas que les prédicats étoiles ne peuvent pas être évalués en respectant les contraintes imposées en temps réel par l'application, et / ou que la valeur pro-activement dérivé n'est plus valable au moment de l'évaluation. D'ou le raisonnement ontologique est utilisé avec une stratégie de meilleur effort. Afin d'assurer l'efficacité élevée requise pour les services en temps réels en informatiques mobiles, la ligne de raisonnement à base de règles est effectuée par un moteur d'inférence à usage spécial, séparément de raisonnement ontologique qui est fait avec le raisonneur RacerPro.

Discussion et critique

Le raisonnement ontologique est également effectué pour vérifier la cohérence du modèle de contexte mais les problèmes liées à la complexité de calcul en un temps acceptable le rendent inutilisable car son raisonnement n'apporte pas d'avantage par rapport au raisonnement avec le modèle cc /pp, En autre cc /pp aussi reste non intuitive et difficile a utiliser pour décrire les situations complexes et pour décrire les informations du contexte en un temps réels donc il est inutile dans ce cas de l'utiliser comme un intermédiaire pour décrire les données du contexte complexe dérivée par un raisonnement ontologique et pour dériver de nouvelles données de contexte telles que préférences de l'utilisateur a partir de ce dernier.

2.7 Conclusion

Dans ce chapitre nous avons étudié le contexte dans l'informatique ubiquitaire, nous avons présentés ces différentes caractéristiques, ces catégories, la sensibilité au contexte et mis en évidence les exigences de la modélisation des informations contextuelles et les différentes approches de la modélisation. Les limites de ces approches en ce qui concerne la satisfaction de certaines exigences de la modélisation ont conduit les chercheurs à combiner entre les différents modèles ce qu'on appelle la modélisation hybride afin d'obtenir plus de systèmes souples et généraux.

Les approches de modélisation hybrides déterminent des avantages évidents en ce qui concerne ces exigences mais elles ne sont pas généralement satisfaisantes en raison de certaines limitation.

Dans le chapitre suivant, nous allons faire une étude approfondie des deux approches de modélisation Event calculus basé sur la logique et les ontologies.

CHAPITRE 3

LES APPROCHES DE MODÉLISATION LOGIQUE EVENT CALCULUS ET ONTOLOGIE

3.1 Introduction

L'objectif de départ de Event Calculus est, assez ambitieux, pouvoir traduire naturellement, et traiter des situations complexes faisant intervenir le temps, dans un langage de programmation logique. Dans ce qui suit, nous allons vous présenté l'approche de modélisation Event calculus, ses différents concepts ainsi que les différentes versions proposées de cette approche,le raisonnement avec Event Calculus afin de pouvoir atteindre les objectifs de départ et nous décrivons les principes de base des ontologies OWL, du raisonnement avec OWL et des règles SWRL , ainsi que les possibilités qu'ils offrent. Cette description permettra de comprendre les concepts et les choix détaillés dans le reste de ce chapitre.

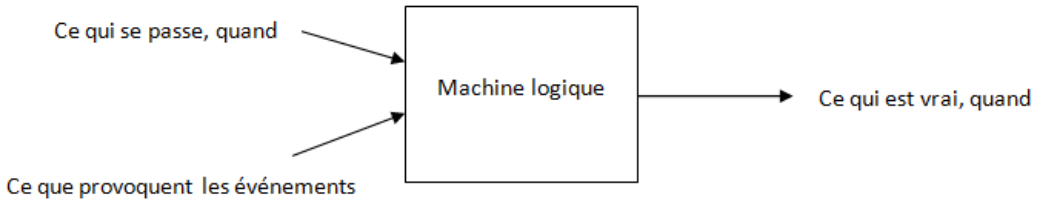
3.2 Event calculus

Event calculus est introduit par Kowalski et Sergot [41] comme un formalisme de programmation logique pour la représentation des événements et leurs effets.

Event calculus est un mécanisme logique qui permet d'inférer ce qui est vrai, sachant :

– Ce qui se passe, et quand, et

– Ce que provoquent les événements.



3.3 Types et formes de base de EC

Il existe trois types de base dans EC : events, fluents et les points temporels. Il inclut aussi un ensemble de prédicats et d'axiomes.

Event : peut être une action qui se produit dans le monde réel.

ξ(variables e,e1,....)

Fluent : définit comme une fonction dont le domaine est l'espace des situations. La valeur du fluent change dans le temps (ça peut être : une propriété « il pleut », quantité «température » ...), les fluents sont donc réifiés.

ℑ(variables f,f1, ...)

Point temporel : les points temporels sont utilisés dans EC afin de renforcer la notion de séquence. Un point temporel peut être une période (les premières versions de EC), des réels positifs ou négatifs (certaines versions considèrent que le temps est positif). Les opérateurs utilisés pour la comparaison des points temporels sont : <, >, >=, <=, =.

τ(variables t, t1,.)

Prédicats : Les prédicats permettent :

– **D**'exprimer quels événement se produisent, et quand ils vont se produire.

– **D**e décrire les effets de ces événements.

– **D**e donner des valeurs de fluents, selon le temps.

Chaque version de EC définit ses propres prédicats.

Les axiomes :

C'est un ensemble de propositions considérées vraies. Chaque version de EC définit ses propres axiomes de base.

3.4 Les capacités représentationnelles de EC

3.4.1 La circonscription

Dans EC, la circonscription est le fait de ne considérer que les events connus (Il n'y a pas d'autres types d'events), et le raisonnement ne s'effectuera que sur ces événements connus

(supposés) [42].

La circonscription, dans EC, est appliquée sur tous les prédicats.

3.4.2 Le raisonnement révisable (defeasible reasonning)

Le raisonnement révisable est un type particulier de raisonnement non démonstratif, où le raisonnement ne produit pas une démonstration complète ou définitif d'une assertion [43].

Dans un formalisme à raisonnement révisable [44], un argument est utilisé comme révisable pour appuyer des conclusions. Une conclusion sera considérée comme valide uniquement lorsque l'argument qui l'appui devient une justification.

Dans EC, le prédicat **terminates** permet au raisonnement révisable d'être possible en arrêtant un fluent d'être vrai à un certain instant. Le prédicat a comme paramètre un event **e**, un fluent **f** et un instant **t**, **terminates (e,f,t)** qui veut dire que l'event e termine (essaie de terminer) **f** après l'instant **t**. Inversement, le prédicat **Initiates (e,f,t)** met le fluent **f** à vrai après **t**. De cette façon, il est possible que les hypothèses d'un modèle peuvent être contestées (challenged), par des changements potentiels des valeurs de vérité des états d'un fluent. Les formules associées aux prédicats **Initiates** et **terminates** sont comme étant des faits.

3.4.3 Manipuler les contradictions

Il est possible de faire face aux observations contradictoires des faits dans EC. Pour le fluent **f** l'instant **t**, les deux prédicats **HoldsAt (f, t)** et ¬**HoldsAt(f, t)** sont contradictoire quand ils apparaissent dans une même base de connaissances de EC. Alors, les contradictions doivent être évitées dans l'ensemble des prédicats de EC.

3.4.4 Catégorisation du contexte

La sensibilité au contexte exige que les informations contextuelles soient collectées et présentées à l'application d'adaptation. Vu l'hétérogénéité, la diversité et la qualité de ces informations, il est désirable de faire une classification ou bien une catégorisation pour faciliter l'opération d'adaptation. Dans ce domaine précis, plusieurs chercheurs ont proposé des catégorisations selon différentes approches.

3.4.5 Le changement continu

La nécessité de représenter les changements continus est une exigence bien établie dans les schémas de représentation en IA[45].

Dans EC, ce problème est résolu en intégrant les prédicats **Trajectory** et **Antitrajectory**. **Trajectory** apparait dans les premier travaux de shanahan. **Antitrajectory** est ajouté dans EC par la suite par Miller et Shanahan ([46][47][48]).

cond ⇒ Trajectory (f1, t1, f2, t2)

f1, f2 représentent le fluent qui varie dans l'intervalle [t1,t2[.

Si un event initie **f1** à l'instant **t1** alors **f2** doit se produire à l'instant **t2** .

3.4.6 la loi d'inertie (the commonsense law of inertia)

Le concept de la loi d'inertie est d'abord défini par référence au calcul de situation par Lifschitz [49] où il est présenté comme la loi qui veille à ce qu'un fluent est vrai par défaut après qu'il a été rendu vrai par une action,ou faux par défaut aprés qu'il a été rendu faux par une action . Cela se traduit par EC dans les prédicats **Release** et **ReleasedAt** , présentés par Miller et Shanahan [48]. Une déclaration de la forme **Release(e, f, t)** dit que l'événement **e** libère un fluent **f** à l'instant **t** , ce qui signifie que son état devient sujet aux changements, tandis que la déclaration **ReleasedAt (f, t)** est une observation que **f** est libéré de la loi d'inertie à l'instant **t** .

3.4.7 La concurrence

La concurrence est relativement facile à représenter dans EC, deux événements peuvent être considérés comme concurrents s'ils se produisent au même point temporel. Dans toutes les versions de EC, les fluents peuvent être utilisés pour représenter les processus qui se produisent sur des intervalles, et dans le but de représenter deux fluent **f1** et **f2** qui se produisent en même temps, il est seulement nécessaire d'avoir deux états **HoldsAt(f1, t1)** et **HoldsAt (f2, t2)** : par défaut, **f1** et **f2** sont supposés se produire en concurrence sur un intervalle **t1** et **t2** .

3.5 Caractéristiques de EC

Nous pouvons résumer les caractéristiques de EC dans les points suivants :

3.5.1 Parcimonie(économie) de la représentation

EC permet de résoudre le problème de cadre (frame problem : concerne la représentation des non effects d'un event) en utilisant un minimum des nouvelles informations. Comme la circonscription assure qu'il est suffisant de considérer seulement les effets d'un événement et il n'est pas nécessaire de prendre ses non effets [50].

3.5.2 Flexibilité expressive

La flexibilité expressive de EC est démontrée par sa capacité de faire face à tous les besoins représentationnels y compris les événements concurrents, les événements révisables (defeasible events), les contradictions, le non déterminisme et le changement continus (countinous change).

3.5.3 La tolérance à l'élaboration

Selon la définition de McCarthy's, un formalisme logique est tolérant à l'élaboration (tolerant elaboration formalism) si l'ensemble des efforts requis pour l'ajout d'une nouvelle information à la représentation est proportionnel à la complexité de cette information [51].

EC est considéré comme étant un formalisme tolérant à l'élaboration car l'ajout d'un nouveau fait (exemple : valeur d'un fluent) à la base de connaissance de EC nécessite seulement l'ajout d'une nouvelle phrase (concernant la nouvelle information) et ne requiert pas l'ajustement de la base de connaissance existante.

3.6 Les différentes versions d'Event Calculus

Event Calculus a considérablement évolué par rapport à sa première version originale. Dans la section qui suit nous allons présenter les plus importantes versions de EC.

3.6.1 Original Event Calculus (OEC)

OEC est introduit par Kowalski et Sergot [41]. Les types de base de OEC sont les occurrences d'event, les fluents et les périodes de temps. Les prédicats et les fonctions de base de OEC sont donnés dans la table 01. Les axiomes de OEC sont les suivants :

OEC1. Initiates(e,f) $\equiv$ Holds(After(e,f)).

OEC2. Terminates(e,f) $\equiv$ Holds(Before(e, f)).

OEC3. Start(After(e,f),e).

OEC4. End(Before(e, f), e).

OEC5. After(e1, f) = Before(e2, f) $\supset$ Start(Before(e2, f), e1).

OEC6. After(e1, f) = Before(e2, f) $\supset$ End(After(e1, f), e2).

OEC5. Holds(After(e1, f)) $\wedge$ Holds(Before(e2, f)) $\wedge$ e1 $<$ e2 $\wedge$ $\neg$Broken(e1,f, e2) $\supset$ After(e1, f) = Before(e2, f).

OEC8. Broken(e1, f, e2) $\equiv \exists$ e, f1 ((Holds(After(e, f1))$\vee$ Holds(Before(e, f1)))$\wedge$

Incompatible(f, f1) $\wedge$ e1 $<$ e $<$ e2).

Où :

$\supset$: représente une implication.

≡ : représente une bi-implication.

Prédicat ou fonction	**Le sens**
Holds(p)	p a eu lieu
Start(p, e)	e débute p
End(p, e)	e termine p
Initiates(e, f)	e initie f
Terminates(e, f)	e termine f
e1 < e2	e1 précède e2
Broken(e1, f, e2)	f est interrompu entre e1 et e2
Incompatible(f1, f2)	f1 et f2 sont incompatible
After(e, f)	f est vrai durant la période déclenchée par e
Before(e, f)	f est vrai la période terminée par e

TABLE 3.1 – Prédicats et fonctions de OEC

Sachant que (e, e1, e2 = event occurrences, f, f1, f2 = fluents, p = time period).

3.6.2 Simplified Event Calculus (SEC)

Simplified Event Calculus (SEC) est proposé par Kowalski en 1986[52] et développé par Sadri[53], Eshghi [54], et Shanahan[55].

Les différences entre SEC et OEC sont résumées dans les points suivants :

- Remplace les périodes de temps par les instants de temps qui peuvent être des entiers positifs ou des réels positifs.
- Remplace la notion d'occurrence d'un event par la notion de type d'event.
  Exemple : le prédicat **Happens(e,t)** veut dire que : le type d'event **e** se déroule à l'instant **t** .
- Il élimine l'incompatibilité.
- Ajoute le prédicat **Initially(f)** qui veut dire que le fluent **f** est initialement vrai.

Les prédicats et les fonctions de base de SEC sont donnés dans la table 02. Les axiomes de SEC sont les suivants :

SEC1. ((Initially(f) ∧ ¬StoppedIn(0, f, t)) ∨ ∃ e, t1 (Happens(e, t1) ∧ Initiates(e, f, t1) ∧ t1 < t ∧ ¬StoppedIn(t1, f, t))) ≡ HoldsAt(f, t).

SEC2. StoppedIn(t1, f, t2) ≡ ∃ e, t (Happens(e, t) ∧ t1 < t < t2 ∧ Terminates(e, f, t)).

Prédicat ou fonction	**Le sens**
Initially(f)	f est vrai à l'insatant 0
HoldsAt(f, t)	f est vrai à l' instant t
Happens(e, t)	e se produit à l'instant t
Initiates(e, f, t)	si e se produit à l'instant t, alors f est vrai après t
Terminates(e, f, t)	si e se produit à l'instant t, alors f est faux après t
StoppedIn(t1, f, t2)	f est interrompu entre t1 et t2

TABLE 3.2 – Prédicats et fonctions de SEC.

Sachant que (e, e1, e2 = event type, f, f1, f2 = fluents, t = timepoint)

3.6.3 Basic Event Calculus (BEC)

Shanahan [50,56] a étendu le SEC en permettant au fluents d'être libéré de la loi d'inertie par le prédicat Releases, et en ajoutant la capacité de représenter le changement continu via le prédicat **Trajectory** . Le prédicat **Initially** est divisé en deux prédicats **InitiallyP** et **InitiallyN** . Nous appelons cette version d'event calculus Basic Event Calculus (BEC).

Les prédicats et les fonctions de base de BEC sont donnés dans la table 03. Les axiomes de BEC sont les suivants :

BEC1. StoppedIn(t1,f,t2) $\equiv \exists$ e,t (Happens(e, t) $\wedge$ t1<t<t2 $\wedge$ (Terminates(e,f,t) $\vee$ Releases(e,f,t))).

BEC2. StartedIn(t1,f,t2) $\equiv \exists$ e, t (Happens(e,t)$\wedge$ t1<t<t2 $\wedge$ (Initiates(e,f,t) $\vee$ Releases(e,f,t))).

BEC3. Happens(e, t1) $\wedge$ Initiates(e, f1, t1) $\wedge$ 0<t2 $\wedge$ Trajectory(f1, t1, f2, t2)
$\wedge \neg$ StoppedIn(t1, f1, t1 + t2) $\supset$ HoldsAt(f2, t1 + t2).

BEC4. InitiallyP(f) $\wedge \neg$StoppedIn(0, f, t) $\supset$ HoldsAt(f, t).

BEC5. InitiallyN(f) $\wedge \neg$StartedIn(0, f, t) $\supset \neg$HoldsAt(f, t).

BEC6. Happens(e, t1) $\wedge$ Initiates(e, f, t1) $\wedge$ t1<t2 $\wedge \neg$StoppedIn(t1,f,t2) $\supset$ HoldsAt(f,t2).

BEC7. Happens(e,t1) $\wedge$ Terminates(e,f,t1) $\wedge$ t1 < t2 $\wedge \neg$StartedIn(t1,f,t2) $\supset \neg$HoldsAt(f,t2).

Prédicat ou fonction	**Le sens**
InitiallyN(f)	f est faux à l'insatant 0
InitiallyP(f)	f est vrai à l' instant 0
HoldsAt(f, t)	f est vrai à l' instant t
Happens(e, t)	e se produit à l'instant t
Initiates(e, f, t)	si e se produit à l'instant t, alors f est vrai après t et f n'est pas libéré de la loi d'inertie.
Terminates(e, f, t)	si e se produit à l'instant t, alors f est faux après t et f n'est pas libéré de la loi d'inertie.
Releases(e, f, t)	si e se produit à l'instant t, alors f est est libéré de la loi d'inertie après t.
StoppedIn(t1, f, t2)	f est interrompu entre t1 et t2
StartedIn(t1, f, t2)	f est déclenché entre t1 et t2
Trajectory(f1, t1, f2, t2)	f1 est initié par un event qui se produit à l'instant t1, alors f2 est vrai à l'instant t1+t2

TABLE 3.3 – Prédicats et fonctions de BEC.

Sachant que (e, = event type, f, f1, f2 = fluents, t1,t2 = timepoint)

3.6.4 Event Calculus (EC)

Miller et Shanahan [48, 42] ont introduit plusieurs formulations alternatives de BEC. Un certain nombre de leurs axiomes peuvent être combinés [17] pour produire ce que nous appelons Event Calculus EC, qui diffère de event calculus de base dans les points suivants :

- Il permet le temps négatif. Timepoints sont des nombres entiers ou réels.
- Il élimine les prédicats **InitiallyN** et **InitiallyP** .
- Il représente explicitement qu'un fluent est libéré de la loi de l'inertie en utilisant le prédicat **ReleasedAt** .
- Il ajoute le prédicat **AntiTrajectory** .
- Il traite **StoppedIn** et **StartedIn** comme des abréviations plutôt que des prédicats, et introduit d'autres abréviations.

Les prédicats et les fonctions de base de EC sont donnés dans la table 04. Les axiomes de EC sont les suivants :

EC1. Clipped(t1, f, t2) def $\equiv \exists$ e, t (Happens(e, t) $\wedge$ t1 $\leqslant$ t $<$ t2 $\wedge$ Terminates(e, f, t)).

EC2. Declipped(t1, f, t2) def $\equiv \exists$ e, t (Happens(e, t) $\wedge$ t1 $\leqslant$ t $<$ t2 $\wedge$ Initiates(e, f, t)).

EC3. StoppedIn(t1, f, t2) def ≡ ∃ e, t (Happens(e, t) ∧ t1 < t < t2 ∧ Terminates(e, f, t)).

EC4. StartedIn(t1, f, t2) def ≡ ∃ e, t (Happens(e, t) ∧ t1 < t < t2 ∧ Initiates(e, f, t)).

EC5. Happens(e, t1) ∧ Initiates(e, f1, t1) ∧ 0 < t2 ∧ Trajectory(f1, t1, f2, t2) ∧ ¬StoppedIn(t1, f1, t1 + t2) ⊃ HoldsAt(f2, t1 + t2).

EC6. Happens(e, t1) ∧ Terminates(e, f1, t1) ∧ 0 < t2 ∧ AntiTrajectory(f1, t1, f2, t2) ∧ StartedIn(t1, f1, t1 + t2) ⊃ HoldsAt(f2, t1 + t2).

EC7. PersistsBetween(t1, f, t2) def ≡ ¬ ∃ (ReleasedAt(f, t) ∧ t1 < t ⩽ t2).

EC8. ReleasedBetween(t1, f, t2) def ≡ ∃ e, t (Happens(e, t) ∧ t1 ⩽ t < t2 ∧ Releases(e, f, t)).

EC9. HoldsAt(f, t1) ∧ t1 < t2 ∧ PersistsBetween(t1, f, t2) ∧ ¬Clipped(t1, f, t2) ⊃ HoldsAt(f, t2).

EC10. ¬HoldsAt(f, t1) ∧ t1 < t2 ∧ PersistsBetween(t1, f, t2) ∧ ¬Declipped(t1, f, t2) ⊃ ¬HoldsAt(f, t2).

EC11. ReleasedAt(f, t1)∧ t1 < t2 ∧ ¬Clipped(t1,f,t2)∧ ¬Declipped(t1, f, t2) ⊃ ReleasedAt(f, t2).

EC12. ¬ReleasedAt(f, t1) ∧ t1 < t2 ∧ ¬ReleasedBetween(t1, f, t2) ⊃ ¬ReleasedAt(f, t2).

EC13. ReleasedIn(t1, f, t2) def ≡ ∃ e, t (Happens(e, t) ∧ t1<t<t2 ∧ Releases(e, f, t)).

EC14. Happens(e,t1) ∧ Initiates(e,f,t1) ∧ t1 < t2 ∧ StoppedIn(t1,f,t2) ∧ ¬ReleasedIn(t1,f,t2) ⊃ HoldsAt(f,t2).

EC15. Happens(e,t1) ∧ Terminates(e,f,t1) ∧ t1<t2 ∧ ¬StartedIn(t1,f,t2) ∧ ¬ReleasedIn(t1, f, t2) ⊃ ¬HoldsAt(f, t2).

EC16. Happens(e,t1) ∧ Releases(e,f,t1) ∧ t1<t2 ∧ ¬StoppedIn(t1,f,t2) ∧ ¬StartedIn(t1,f,t2) ⊃ ReleasedAt(f,t2).

EC17. Happens(e,t1) ∧ (Initiates(e,f,t1) ∨ Terminates(e,f,t1)) ∧ t1 < t2 ∧ ¬ReleasedIn(t1,f,t2) ⊃ ¬ReleasedAt(f,t2).

Prédicat ou fonction	**Le sens**
HoldsAt(f, t)	f est est vrai à l' instant t
Happens(e, t)	e se produit à l'instant t
ReleasedAt(f, t)	f est libéré de la loi d'inertie à l'instant t.
Initiates(e, f, t)	si e se produit à l'instant t, alors f est vrai après t et f n'est pas libéré de la loi d'inertie.
Terminates(e, f, t)	si e se produit à l'instant t, alors f est faux après t et f n'est pas libéré de la loi d'inertie.
Releases(e, f, t)	si e se produit à l'instant t, alors f est libéré de la loi d'inertie après t.
Trajectory(f1, t1, f2, t2)	si f1 est initié par un event qui se produit à l'instant t1, alors f2 est vrai à l'instant t1+t2
AntiTrajectory(f1, t1, f2, t2)	si f1 est terminé par un event qui se produit à l'instant t1, alors f2 est vrai à l'instant t1+t2

TABLE 3.4 – Prédicats et fonctions de EC.

Sachant que (e, = event type, f, f1, f2 = fluents, t1,t2 = timepoint).

3.6.5 Discrete Event Calculus (DEC)

Mueller [57, 58] a développé Discret Event Calculus (DEC) afin d'améliorer l'efficacité du raisonnement automatisé dans Event Calculus. DEC améliore l'efficacité en limitant le temps à des entiers. Les prédicats de DEC sont les mêmes que ceux de EC, comme le montre la table 04. Les axiomes et les définitions de DEC sont les suivants :

DEC1. StoppedIn(t1, f, t2) def $\equiv \exists$ e,t (Happens(e, t) $\wedge$ t1 $<$ t $<$ t2 $\wedge$ Terminates(e, f, t)).

Ceci déclare que le fluent f est arrêté entre le point temporel T1 et T2 s'il y à un événement qui termine f après T1 et avant T2.

DEC2. . StartedIn(t1, f, t2) def $\equiv \exists$ e, t (Happens(e, t) $\wedge$ t1 $<$ t $<$ t2 $\wedge$ Initiates(e, f, t)).

Ceci déclare qu'un fluent f est commencé entre le point temporel T1 etT2 s'il y à un événement qui lance f après T1 et avant T2.

Le **DEC1** et le **DEC2** vérifient la propriété stopped and started.

DEC3. Happens(e,t1)$\wedge$ Initiates(e,f1,t1)$\wedge$ 0 $<$ t2 $\wedge$ Trajectory(f1,t1,f2, t2)$\wedge \neg$ StoppedIn(t1,f1,t1 + t2) $\supset$ HoldsAt(f2,t1 + t2).

Ceci déclare que si un événement se produit pour lancer un fluent f1 à l'instant t1 et s'il y

a une trajectoire qui fait à ce déclenchement un autre fluent f2 après une période t2, alors le fluent f2 se tiendra à t1+t2, supposant que f1 n'est pas arrêté entre le t1 et le t1+t2.

DEC4. Happens(e,t1) $\wedge$ Terminates(e,f1,t1) $\wedge$ 0 $<$ t2 $\wedge$ AntiTrajectory(f1, t1,f2, t2) $\wedge$ $\neg$StartedIn(t1, f1, t1 + t2) $\supset$ HoldsAt(f2, t1 + t2).

Ceci définit l'équivalent au DEC3 pour des anti trajectoires, ainsi si un événement se produit pour terminer le fluent f1 à l'instant t1 et un anti trajectoire f1 fait le déclenchement de f2 à t1+t2, alors f2 se tiendra à t1+t2, supposant que f1 n'est pas remis en marche entre le t1 et le t1+t2.

Le **DEC3** et le **DEC4** vérifient la propriété trajectory et antitrajectory .

DEC5. HoldsAt(f, t) $\wedge$ $\neg$ReleasedAt(f, t + 1) $\wedge$ $\neg$ $\exists$ e (Happens(e, t) $\wedge$ Terminates(e, f, t)) $\supset$ HoldsAt(f, t + 1).

Ceci déclare que si un fluent f est vraie à l'instant t et n'est pas libéré de la loi d'inertie à l'instant t et il n'existe pas un événement qui se produit à t pour terminer le fluent f a l'instant t, alors le fluent f se tiendra au prochain point temporel, c.-à-d. t+1.

DEC6. $\neg$HoldsAt(f, t) $\wedge$ $\neg$ReleasedAt(f, t + 1) $\wedge$ $\neg$ $\exists$ e (Happens(e, t) $\wedge$ Initiates(e, f, t)) $\wedge$ $\neg$HoldsAt(f, t + 1).

C'est identique au DEC5 à moins qu'il traite des cas où le fluent f ne se tient pas à t et implicitement à t+1.

DEC7. ReleasedAt(f, t) $\wedge$ $\neg$ $\exists$ e (Happens(e, t) $\wedge$ (Initiates(e, f, t) $\vee$ Terminates(e, f, t))) $\supset$ ReleasedAt(f, t + 1).

Ceci déclare que si un fluent f est libéré de la loi d'inertie à l'instant t et un événement ne se produit pas à t pour terminer le fluent f à l'instant t ou pour initier le fluent f à l'instant t, alors le fluent f sera encore libéré de la loi de l'inertie au prochain point temporel, c.-à-d. t+1.

DEC8. $\neg$ReleasedAt(f, t) $\wedge$ $\neg$ $\exists$ e (Happens(e, t) $\wedge$ Releases(e, f, t)) $\supset$ $\neg$ReleasedAt(f, t + 1).

C'est identique au DEC7 à moins qu'il traite des cas où f n'est pas libéré à t et implicitement à t+1.

DEC9. Happens(e, t) $\wedge$ Initiates(e, f, t) $\supset$ HoldsAt(f, t + 1).

Ceci déclare que si l'événement e initie le fluent f à l'instant t alors f est vraie à l'instant t+1.

DEC10. Happens(e, t) $\wedge$ Terminates(e, f, t) $\supset$ $\neg$HoldsAt(f, t + 1).

Ceci déclare que si l'événement e termine f à l'instant t alors f n'est pas vraie à t+1.

DEC11. Happens(e, t) $\wedge$ Releases(e, f, t) $\supset$ ReleasedAt(f, t + 1).

L'axiome DEC11 déclare que si un événement e se produit à l'instant t et il libère le fluent f à l'instant t alors f sera libéré de la loi de l'inertie au temps t+1.

DEC12. Happens(e, t) $\wedge$ (Initiates(e, f, t) $\vee$ Terminates(e, f, t)) $\supset$ $\neg$ReleasedAt(f, t + 1).

Comme complément au **DEC11**, ceci déclare que si l'événement e lance ou termine le fluent f à t alors f ne sera pas libéré de la loi de l'inertie à t+1.

3.6.6 Tableau comparatif des différentes versions de EC

Version	Temps	Le changement continu	concurrence	circonscription	contradictions
OEC	Période	Non	Oui	Oui	Non
SEC	Timepoint Réel >0 Entier>0	Non	Oui	Oui	Oui
BEC	Timepoint Réel >0 Entier>0	Oui	Oui	Oui	Oui
EC	Timepoint Réel ou Entier	Oui	Oui	Oui	Oui
DEC	Timepoint Entier	Oui	Oui	Oui	Oui

TABLE 3.5 – Tableau comparatif des différentes versions de event calculus

3.7 Formalismes alternatifs à event calculus

Nous avons choisi EC parmi d'autres formalismes temporels pour différentes raisons. Nous allons voir dans ce qui suit les autres formalismes temporels existants et expliciter les raisons pour les quelles qu'on a choisi EC.

3.7.1 Situation Calculus

Situation Calculus est proposé par Mc Carthy et Hayes en 1961[59]. Situation Calculus donne une définition logique des concepts fluent et action, qui ont été par la suite pris comme points de départ pour EC et Fluent Calculus.

La différence clé entre EC et Situation Calculus est que SC traite les événements hypothétiques en créant un graphe ramifié des situations d'événements. Situation Calculus ne manipule pas facilement les événements concurrents mais des extensions de Situations Calculus ont été proposées afin de remédier à ces problèmes.

3.7.2 Fluent Calculus

Fluent Calculus [60] est un formalisme qui étend les concepts de Situation Calculus et ajoute la notion d'état qui correspond à un ensemble de fluents.

Fluent Calculus partage certaines similitudes avec EC. Comme EC, Fluent Calculus utilise le prédicat holds pour décrire les fluent qui deviennent vrai pour une situation donnée. Notons, cependant, qu'aucune valeur primitive ne représente un point dans le temps ou un intervalle. Fluent Calculus utilise le concept de situation, il se base sur la ramification plutôt que sur le temps linéaire. Le changement de l'état d'un fluent est provoqué par la modification de la situation existante.

Dans certains contextes, les traitements avec Fluent Calculus deviennent plus long que EC. Par exemple : quand il s'agi de représenter des actions concurrentes qui nécessite trois règles avec Fluent Calculus alors qu'avec EC, on peut exprimer la même chose en une seule règle.

Pour notre cas, Event Calculus offre une syntaxe compréhensible. Sur le plan pratique, il est probable que les règles de fluent Calculus seront plus longues. En particulier, dans le cas où le temps est important, car on sait que Fluent Calculus manque de notion de temps.

3.8 Conclusion

Il existe plusieurs similitudes entre EC et les formalismes étudiés dans cette section. La classification de Sandewall's des logiques temporelles montre les pouvoirs expressifs des différentes logiques, et que EC est juste une des nombreuses approches permettant de représenter les changements d'états à travers le temps.

Cependant, EC rivalise avec les autres formalismes pour sa facilité d'utilisation et sa concision (exprimer un état avec un minimum de règle).

3.9 La logique de description

Ce que nous entendons par logique descriptive est en fait une famille de formalismes pour représenter une base de connaissances d'un domaine d'application. Plus spécifiquement, ces logiques permettent de représenter des concepts (aussi appelés classes) d'un domaine et les relations (aussi appelées rôles) qui peuvent être établies entre les instances de ces classes[61].

3.9.1 Langage de base AL

Les langages de la logique descriptive sont déterminés par la forme des énoncés qui sont permis. La plupart des langages utilisés découlent du langage AL (attributive language), dont l'expressivité est plutôt limitée.

3.9.2 Syntaxe du langage AL

Dans ce langage, les axiomes sont construits à partir d'un ensemble de concepts. Le tableau suivant résume les axiomes et concepts de base de AL.

Symbole	**Signification**	**Concept**	**Exemple**
A	Concept atomique		Humain,Homme,Femme,Animal,Raisonnable
R	Rôle		a Enfant,mariéAvec
$\sqsubseteq$	Inclusion	A $\sqsubseteq$ C	Humain $\sqsubseteq$ Animal Humain est inclus dans Animal
$\equiv$	Définition	A$\equiv$C	Humain $\equiv$ Animal $\sqcap$ Raisonable Humain est un individu Animal et Raisonnable
$\sqcap$	Intersection	A $\sqcap$ C	Humain $\equiv$ Animal $\sqcap$ Raisonable Humain est un individu Animal et Raisonnable
$\bot$	Concept impossible	A $\sqsubseteq \bot$	Extraterrestre $\sqsubseteq \bot$ Impossible d'avoir un individu du type Extraterrestre
$\top$	Concept universel	A $\sqsubseteq \top$	Animal $\sqsubseteq \top$ Tout les individus sont du type Animal
$\neg$	Négation	$\neg$A	$\neg$Humain Les individus qui ne sont pas Humain
$\forall$	Quantificateur universel	$\forall$ R.C	Humain $\sqcap \forall$ aEnfant.Femme Les individus Humains ayant au moins un enfant et dont tous les enfants sont des femmes.
$\exists$	Quantificateur existentiel	$\exists$ R.T	$\exists$ aEnfant.T Individu qui n'a pas d'enfants
$\sqcup$	Union	C $\sqcup$ D	Humain $\equiv$ Homme $\sqcup$ Femme Humain est l'union des deux ensembles Homme et Femme
F	Fonction	F un (R)	F un (mariéAvec) Le Rôle mariéAvec est une fonction
$^-$	Inversion ($^-$)	C $\equiv$ D$^-$	estRegardéPar(Maria,Paulo) $\equiv$ Regarde$^-$(Paulo,Maria) Paulo regarde Maria,on sait aussi que Maria est regardée par Paulo.
N	Restriction non qualifiée ($\geqslant$ ou $\leqslant$)	$\geqslant$ n R $\leqslant$ n R	Femme $\sqcap \geqslant$ 0 aEnfant (Femme qui n'a pas d'enfants) FemmeMarié $\equiv$ Femme $\sqcap \exists$ mariéAvec. T $\sqcap \leqslant$1 mariéAvec (Femme mariée avec au plus 1 seul homme)
Q	Restriction qualifiée($\leqslant \geqslant$)	$\geqslant$ n R.C $\leqslant$ n R.C	FemmeMarié $\equiv$ Femme $\sqcap \exists$ mariéAvec.T$\sqcap \leqslant$1 mariéAvec.Homme

TABLE 3.6 – Les axiomes et les concepts de base de AL.

3.9.3 La ABox et la TBox

Dans une base de connaissances en logique descriptive, on distingue deux niveaux : terminologique (TBox) et assertionnel (Abox). Le premier niveau permet d'exprimer à l'aide d'axiomes des relations entre les concepts, comme la définition du concept mère qui est défini comme étant une femme qui est le parent d'au moins un humain.le second niveau de notre ontologie contient des assertions sur des individus, en spécifiant leur classe et leurs attributs. C'est dans la ABox,

par exemple, qu'on indiquerait que Marie est une femme et qu'elle a deux enfants[61].

Exemple : voici un exemple d'ontologie

TBox :

Célibataire $\equiv$ Personne $\sqcap \leqslant 0$ mariéAvec

Personne $\sqsubseteq \exists$ nom $\sqcap \exists$ age

Homme $\sqsubseteq$ Personne

Femme $\sqsubseteq$ Personne

Père $\equiv$ Personne $\sqcap \exists$ aEnfant.Personne

PèreDeFilles $\equiv$ Père $\sqcap$ aEnfant.Femme

$\top \sqsubseteq \sqcap$ aEnfant.Personne (image)

$\exists$ aEnfant.$\top \sqsubseteq$ Personne (domaine)

aEnfant $\equiv$ aParent$^-$(rôle inverse)

$\top \sqsubseteq \leqslant 1$ estConjointDe (rôle fonctionnel)

$\top \sqsubseteq \leqslant 1$ estConjointDe$^-$(rôle injectif)

estConjointDe $\equiv$ estConjointDe$^-$(rôle symétrique)

$\top \sqsubseteq \forall$ estConjointDe.Personne (image)

$\exists$ estConjointDe.$\top \sqsubseteq$ Personne (domaine)

mariéAvec $\sqsubseteq$ estConjointDe

$\top \sqsubseteq \forall$ age (image)

$\exists$ age.$\top \sqsubseteq$ Personne (domaine)

$\top \sqsubseteq \forall$ nom (image)

$\exists$ nom.$\top \sqsubseteq$ Personne (domaine)

ABox :

Homme(Bernard)

age(Bernard,56)

Femme(Sabine)

age(Sabine,46)

Homme(Valentin)

age(Valentin,10)

mariéAvec(Bernard,Sabine)

aEnfant(Bernard,Valentin)

3.9.4 Inférence

3.9.4.1 Inférence au niveau TBox

En logique descriptive, il y a quatre propriétés qu'on peut être intéressé à prouver pour une TBox [61] :

Satisfaisabilité :

Un concept C1 est consistant, si au moins, un individu du monde décrit appartient à l'ensemble d'individus associés à ce concept C1. Contre-exemple :

C1 ≡ Homme ⊓ Femme

Subsomption :

Un concept C1 subsume un concept C2, si C1 est plus général que C2 et surtout si les individus contenus dans C2 sont aussi des individus contenus dans C1. Exemple :

Homme ⊑ Humain

Équivalence :

Deux concepts C1 et C2 sont équivalents, si l'ensemble des individus associés au concept C1 est égal à l'ensemble des individus associés au concept C2. Exemple :

Humain ≡ Personne

Disjonction :

Deux concepts C1 et C2 sont disjoints si leur intersection est vide. Exemple :

Homme ⊓ Femme ⊑ ⊥

3.9.4.2 Inférence au niveau ABox

Le raisonnement sur une ABox se focalise sur le test de la correction d'un modèle du domaine. Il faut effectuer les deux tâches suivantes [61] :

Vérification d'instance :

Vérifier si un individu a d'une ABox A est une instance d'une description de concept donnée C.

Vérication de consistance :

Une ABox A est consistante par rapport à une TBox T, s'il existe une interprétation qui est un modèle des deux, A et T.

3.9.5 Les raisonneurs existants

Il existe plusieurs raisonneurs qui ont été développés (Open Source ou commerciaux)

- CerebraEngine, FaCT++ , FuzzyDL (fuzzy DL) ;
- KAON2, QuOnto ;
- Pellet, RacerPro.

3.10 Les Langages de spécification d'ontologie

Plusieurs langages de spécification d'ontologies (ou langages d'ontologies) ont été développés pendant les dernières années :

3.10.1 RDF Resource Description Framework

RDF [40] est un standard pour la description et l'exploitation des métas donnés dont la syntaxe est basée sur XML. De manière plus générale, RDF permet de voir le Web comme un ensemble de ressources reliées par les liens étiquetés " sémantiquement ". La sémantique dans RDF est spécifiée sous forme de triplets. Chaque triplet est constitué d'un sujet (ressource), un prédicat (propriété) et un objet (valeur). Une "ressource" (resource) est définie par des "propriétés" (properties) ; l'association d'une ressource à une propriété par une valeur de propriété est une "déclaration" (statement).un ensemble de tels triplets est appelé un graphe RDF. Ceci peut être illustré par un diagramme composé de noeuds et d'arcs dirigés, dans lequel chaque triplet est représenté par un lien noeud-arc-noeud (d'où le terme de "graphe").

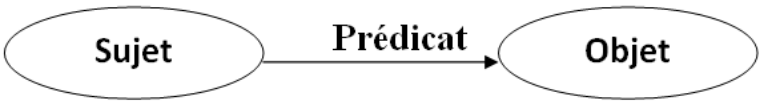


FIGURE 3.1 – Exemple de représentation d'un triplet sous forme de graphe.

3.10.1.1 Syntaxe de RDF

RDF n'est, en soi, qu'un modèle de graphes à arcs orientés et labellés, dont le label est le " prédicat " (ou" propriété "), le noeud de départ le " sujet ", et le noeud cible " l'objet ". RDF/XML propose une syntaxe de sérialisation de RDF. Pour cela, RDF/XML s'appuie sur XML par sa syntaxe, définie par une Recommandation du W3C de 1998, et mise à jour en 2004 [62].

Le même triplet sérialisé au format RDF/XML :

<rdf :Description about=sujet>
<prédicat>objet< /prédicat>
<rdf :Description/ >

Exemple :

Considérons l'information suivante : «l'auteur de http ://www.lacot.org/ est Xavier Lacot ». On peut choisir de représenter cette information en triplet RDF par la chaine « {auteur de http ://www.lacot.org/, Xavier Lacot, est} », ou encore « est(auteur de http ://www.lacot.org/, Xavier Lacot) », ou encore, plus communément, « <auteur de http ://www.lacot.org/> <est> <Xavier Lacot> ». En RDF/XML, cette information s'écrit :

<rdf :Description about="http ://www.lacot.org/">
<schema :auteur>Xavier Lacot</schema :auteur>
</rdf :Description>

3.10.2 RDFS

RDFS (pour RDF Schéma [63]) a pour but d'étendre ce langage en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les ressources seront des instances, comme les propriétés.RDFS s'écrit toujours à l'aide de triplets RDF en définissant la sémantique de nouveaux mots clés comme :

<ex:Vehicule rdf:type rdfs:Class> la ressource ex:Vehicule a pour type rdfs:Class, et est donc une classe
<sncf:TER8153 rdf:type ex:Vehicule> la ressource sncf:TER8153 est une instance de la classe ex:Vehicule que nous avons définie
<sncf:Train rdfs:subClassOf ex:Vehicule> la classe sncf:Train est une sousclasse de ex:Vehicule, toutes les instances de sncf:Train sont donc des instances de ex:Vehicule
<ex:localisation rdf:type rdfs:Property> affirme que ex:localisation est une propriété (une ressource utilisable pour étiqueter les arcs)
<ex:localisation rdfs:range ex:Ville> affirme que toute ressource utilisée comme extrémité d'un arc étiqueté par ex:localisation sera une instance de la classe ex:Ville.

FIGURE 3.2 – Exemple d'une définition de la sémantique de nouveaux mots clés.

On a vu que RDF et RDFS permettent de définir, sous forme de graphes de triplets, des données ou des métadonnées. Cependant, de nombreuses limitations bornent la capacité d'expression des connaissances établies à l'aide de RDF/RDFS. On peut citer, par exemple, l'impossibilité de raisonner et de mener des raisonnements automatisés (automated reasoning)sur les modèles de connaissances établis à l'aide de RDF/RDFS. C'est ce manque que se propose de combler OWL.

3.10.3 OWL Web Ontology Language

OWL [38] est un langage sémantique fondé sur la syntaxe de RDF/XML.Il étend les concepts définis dans le schéma RDF afin d'intégrer d'autre types d'informations qui ne sont pas pris en ;compte par la spécification RDF. OWL offre un moyen d'écrire des ontologies web. OWL se différencie du couple RDF/RDFS en ceci que, contrairement à RDF, il est justement un langage

d'ontologies. Si RDF et RDFS apportent à l'utilisateur la capacité de décrire des classes (ie. avec des constructeurs) et des propriétés, OWL intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc. Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grâce à un vocabulaire plus large et à une sémantique formelle définie par une syntaxe rigoureuse.

Il existe trois versions du langage [66] :

OWL-Lite, OWL-DL et OWL-Full.

- **OWL Lite :** fournit aux utilisateurs une classification hiérarchique et des contraintes simples. L'idée sous-jacente d'OWL Lite est de fournir un sous-ensemble minimal utile des caractéristiques d'OWL qui sont relativement simples, ainsi qu'une hiérarchie de base : des sous-classes et des contraintes sur les propriétés. Les implémentations qui supportent le vocabulaire d'OWL Lite font qu'un système OWL peut interagir avec des modèles RDFS, des bases de données ;
- **OWL DL :** supporte un niveau d'expressivité supérieur à celui d'OWL-Lite et inclut tout le vocabulaire d'OWL. Ce vocabulaire ne peut être utilisé que sous certaines contraintes (par exemple, alors qu'une classe est sous-classe de plusieurs classes, une classe ne peut pas être une instance d'une autre classe). OWL DL est nommé de cette manière en raison de sa correspondance avec les logiques de description (" description logics ") [64], un domaine de recherche qui fournit à un constructeur d'ontologies ou à l'utilisateur un support pour le raisonnement ;
- **OWL Full :** consiste en la combinaison de l'expressivité d'OWL, la flexibilité et les caractéristiques de modélisation de RDF (par exemple, dans OWL Full une classe peut être traitée simultanément comme une collection d'individus et comme un individu). Cependant, l'utilisation d'OWL Full signifie la perte de quelques garanties qu'OWL DL et OWL Lite peuvent fournir pour les systèmes de raisonnement.

Dans ce document, nous nous intéressons à OWL-DL.

3.10.3.1 Définition des éléments d'une ontologie OWL

Dans cette sous-section, nous fournissons des définitions nécessaires pour comprendre le principe des ontologies OWL. Nous commençons par une définition générale [65] :

Déf. 3.1 (Domaine du discours) Le domaine du discours est le domaine que l'on représente dans une ontologie, c'est-à-dire, la partie du monde qui est l'objet de la modélisation.

Ensuite, nous définissons les éléments principaux que l'on trouve dans une ontologie OWL :

Déf. 3.2 (Individu/instance) Les individus ou instances sont les objets du domaine de discours que l'on représente dans une ontologie.

Déf. 3.3 (Concept/classe, sous-classe, super-classe, taxonomie) Un concept ou classe regroupe un ensemble d'individus qui ont des caractéristiques communes. Une classe peut être sous-classe d'une autre, appelée super-classe : dans ce cas, tout individu appartenant à la sous-classe appartient aussi à la super-classe. Une taxonomie est une hiérarchie de classes qui ont des relations sous-classe/super-classe entre elles.

Déf. 3.4 (Relation/propriété, domaine, portée, sous-relation, super-relation) Une relation ou propriété modélise le rapport qui existe entre deux classes (relation objet) ou entre une classe et un type de données (relation type de données). Le domaine d'une relation est l'ensemble de classes qui peuvent être à l'origine de la relation. La portée de la relation est l'ensemble de classes ou de types de données qui peuvent être la destination de la relation. Une relation peut être une sous-relation d'une autre, appelée super-relation. Dans ce cas, le domaine et la portée de la sous-relation sont contenus respectivement dans le domaine et dans la portée de la super-relation.

Déf. 3.5 (Instance dune relation) Une instance dune relation relie un individu qui appartient au domaine de la relation à un individu ou à un type de données qui appartient à la portée de la relation.

Par abus de langage, souvent on appelle les instances de relations tout simplement relations. Le contexte du terme permet de distinguer si lon parle de relations proprement dites ou de leurs instances.

Les définitions suivantes détaillent les attributs des propriétés.

Déf. 3.6 (Propriété inverse) On peut définir une propriété comme linverse dune propriété donnée. Cela veut dire que, si une instance de cette dernière propriété relie lindividu a à lindividu b, alors on peut déduire quune instance de la propriété inverse relie b à a.

Déf. 3.7 (Propriété fonctionnelle, inverse fonctionnelle) Une propriété est fonctionnelle quand elle ne peut avoir quune seule instance pour un individu donné. Linverse dune propriété fonctionnelle est son inverse fonctionnelle.

Déf. 3.8 (Propriété transitive) Une propriété est transitive lorsque, si lindividu a est relié à b par une instance de cette propriété et que b est relié à c par une autre instance de la propriété, alors on peut déduire que a est relié à c par une instance de la propriété.

Déf. 3.9 (Propriété symétrique) Une propriété est symétrique quand pour tout a relié à b par une instance de cette propriété, on peut déduire que b est relié à a par une autre instance de la propriété.

Pour définir une classe dans OWL, on fournit un ensemble de conditions logiques. Ces conditions peuvent être nécessaires ou nécessaires et suffisantes. Elles sont construites à partir dautres classes, par union, par intersection ou par héritage. On peut également imposer des restrictions aux propriétés de la classe.

Déf. 3.10 (Restriction, restriction existentielle, universelle, de cardinalité, de valeur) Une

restriction consiste à limiter le nombre ou la nature des valeurs que peuvent avoir les propriétés des individus dune classe. Une restriction peut être existentielle (si elle oblige à avoir au moins une valeur de la propriété dans un ensemble donné), universelle (si elle oblige à avoir toutes les valeurs dune propriété dans un ensemble donné), de cardinalité (si elle oblige à avoir un nombre de valeurs minimal, maximal ou exact pour une propriété) ou de valeur (si elle oblige à avoir une valeur donnée pour la propriété).

Déf. 3.11 (Classes disjointes) Deux classes sont dites disjointes quand il ne peut exister des individus qui appartiennent à la fois aux deux classes.

Dans OWL, les classes ne sont pas disjointes par défaut ; il faut le déclarer explicitement.

Les ontologies OWL sont extensibles grâce au mécanisme dimportation :

Déf. 3.12 (Importation dontologie) Une ontologie peut importer une autre ontologie pour avoir une visibilité sur ses éléments. Lontologie qui importe a un accès en lecture seule à tous les éléments contenus dans lontologie importée. Elle peut ensuite ajouter de nouveaux éléments, qui ne seront visibles que dans lontologie qui importe. Limportation est transitive : une ontologie qui importe une deuxième importe indirectement toutes les ontologies importées dans cette deuxième ontologie.

En général, limportation se fait en indiquant, dans lontologie qui importe, lURL où se trouve lontologie importée. Ce mécanisme permet une grande flexibilité, car on peut réutiliser des ontologies existantes juste en les référençant dans un nouveau fichier OWL et en ajoutant de nouvelles classes, propriétés, individus, règles, etc. Souvent, on trouve des ontologies qui suivent ce schéma : une première ontologie de haut niveau qui contient des éléments génériques est importée par une deuxième ontologie de domaine qui les spécialise pour un domaine concret. Dans ce cas, on dit que la deuxième ontologie étend la première.

3.10.4 Règles SWRL

Le Semantic Web Rule Langage (SWRL) [67] est un langage de règles proposé par le W3C qui combine OWL-DL avec le Rule Markup Langage (RuleML). SWRL étend OWL-DL en ajoutant des clauses de Horn [69]. Cet ajout augmente l'expressivité d'OWL, mais, en général, cette expressivité implique la perte de la décidabilité [70]. Toutefois, pour la plupart des applications pratiques, il est possible de se limiter à un sous-ensemble de règles appelé DL-sûres (DL-safe en anglais), qui est décidable.

Une règle SWRL présente la forme suivante :

$$\text{b1} \wedge \ldots \wedge \text{bn} \longrightarrow \text{a1} \wedge \ldots \wedge \text{an}$$

où b1 $\wedge$...$\wedge$ bn est le corps ou antécédent de la règle et a1 $\wedge$...$\wedge$an est l'entête ou conséquent. Les termes a1, ..., an, b1,..., bn sont des atomes SWRL. Un atome peut représenter

une relation (prédicat binaire), un concept (prédicat unaire) ou un built-in (prédicats n-aires). L'interprétation de la règle est la suivante : si les conditions spécifiées dans l'antécédent sont vérifiées, alors on peut déduire que les propositions spéci ?ées dans le conséquent sont véri ?ées aussi. L'utilité de ces règles est d'exprimer des relations qui seraient trop compliquées, voire impossibles, à exprimer avec OWL-DL seulement. Par exemple, dans notre ontologie on peut représenter la relation entre un oncle et son neveu à partir des relations père-fils et frère-frère. La règle SWRL qui exprime cette relation est représentée dans l'exemple ci-dessous :

$$\text{Personne(?x)} \wedge \text{Personne(?y)} \wedge \text{Personne(?z)}$$
$$\wedge\ \text{pere(?x,?y)} \wedge \text{frere(?x,?z)} \longrightarrow \text{oncle(?z,?y)}$$

3.10.4.1 Les limites des règles SWRL

Le SWRL à des avantages reconnus comme son utilisation dans beaucoup de secteurs, tels que raisonnant au sujet des unités et les informations contextuelles, et la cartographie d'ontologie et nous a permis d'aller lointains au delà du OWL, mais nous avons identifié plusieurs limitations qui sont apparues dans différents domaines d'application : la limitation aux attributs unaires et binaires , le manque de négation-comme-échec et d'autres opérations non mono toniques, l'incapacité de produire de nouveaux individus en résultat d'évaluer une règle, et ne permet pas le raisonnement en temps réel [68].

3.11 Conclusion

Dans ce chapitre, nous avons exposé en premier l'approche de modélisation Event Calculus et ses différentes versions. En second, nous avons présenté la logique de description et en dernier nous avons donné un aperçu du langage ontologique OWL. Toute cette étude est réaliser afin d'étudier la possibilité de combiner l'approche de modélisation logique Event Calculus et le langage ontologique OWL , dans le prochain chapitre nous exposant les détails de notre contribution à la problématique.

CHAPITRE 4

PROBLÉMATIQUE ET PROPOSITION

4.1 Introduction

La modélisation des informations contextuelles est la première démarche dans le processus de création d'applications sensibles au contexte, elle permet de simplifier le développement de ce type d'applications, ce qui nécessite des modèles abstraits de description de contexte, il existe différents approches de modélisation de contexte qui sont classifiés en plusieurs approches mais aucun d'eux ne peut simultanément satisfaire à toutes les exigences de la modélisation des informations contextuelles . Dans ce qui suit nous optons pour la combinaison des deux approches Event Calculus et le langage ontologique OWL afin de répondre au mieux aux exigences de la modélisation.

4.2 Problématique

Par rapport à l′informatique traditionnelle, l′informatique ubiquitaire introduit de nouveau chalenges pour la conception des applications. En particulier, l′informatique ubiquitaire demande aux applications d′être capable de fonctionner dans un environnement extrêmement dynamique en sollicitant le moins possible l′attention des utilisateurs. Ces exigences ont conduit à la conception d′applications dites sensibles au contexte. Ces applications doivent détecter les variations de l′environnement, tels que la localisation et l′identité des utilisateurs et des objets, et adapter leurs comportements en conséquence. Un des besoins clé pour réaliser des systèmes sensibles au contexte est de fournir aux machines la capacité de comprendre leurs contextes. Pour permettre cela, les informations contextuelles doivent être représentées par un modèle qui permet le traitement et le raisonnement automatique. La plupart des travaux de recherche portant sur l′adaptation d′applications au contexte focalisent leurs efforts, en proposant des

modèles de description de contexte ;et en proposant des approches de modélisation hybride qui tentent d′intégrer différents modèles et différents types de raisonnement afin d'obtenir plus de systèmes souples et généraux,mais chacun de ces modéles proposés présentent des inconvénients en ce qui concerne la satisfaction de certaines exigences de la modélisation citées précédement. Pour cela nous cherchons à définir d′autres approches de modélisation hybride qui permettent de répondre à ces exigences et qui permettent de décrire le contexte et de raisonner sur le changement des informations contextuelles afin d'adapter les applications ubiquitaires à ces changements.

4.3 Contribution

Notre travail consiste à proposer une approche de modélisation hybride. Après l'étude des deux approches de modélisation des connaissances contextuelles : Event Calculus et le langage ontologique OWL,avec la prise en considération de quelques exigences de la modélisation telles que la complexité, l'expressivité, la facilité d'implémentation, l'interopérabilité, le raisonnement, la sensibilité au contexte et la capacité représentationnelle des relations et des dépendances etc. ; nous optons pour la combinaison de ces deux approches , vu que l'approche de modélisation Event Calculus est fort coté raisonnement en temp réel, et ca capacité représentationnelle de la négation et malgré qu'il n'est pas capable de représenter les relations et les dépendances de notre base de connaissance ; et qu'il ne règle pas le problème de l'interopérabilité mais sa combinaison avec le langage ontologique OWL qui est fort coté représentation des relations et des dépendances et qui est fort aussi coté interopérabilité fait face à ces limitations, et les limites aussi du le langage ontologique OWL en raisonnement en temp réel et la représentation de la négation ;sont réglées vu qu'il est combiné avec l'approche de modélisation Event Calculus.

Notre approche hybride de la modélisation des connaissances contextuelles, est basé sur une interaction souple entre le modèle Event Calculus et le modèle ontologique. L'interaction entre ces modèles est réalisée par le raisonnement sur les données de contexte avec le modèle Event Calculus qui contient une référence au classes et les relations de OWL-DL. Afin de préserver l'efficacité,la modélisation des connaissances contextuelles et le raisonnement ontologique sont principalement réalisés à l'avance en ce qui concerne la disposition de service.

Chaque fois que de nouvelles données de contexte pertinentes sont acquises, le raisonnement ontologique est démarré, et l'information dérivée est utilisée, avec évaluation des règles efficaces. Les données du contexte complexe dérivées par un raisonnement ontologique peuvent être utilisées dans les conditions préalables des règles de Event Calculus pour dériver des données de nouveau contexte tel que préférences de l'utilisateur, ou pour déclencher des actions d'adaptation, pour beaucoup plus de détails voir la figure1.

4.3.1 Architecture

La figure 1 montre l'architecture globale de notre plate-forme, nous définissons plusieurs agents, chacun est responsable de l'exécution d'un endroit précis, le rôle de chaque agent est détaillé dans ce qui suit.

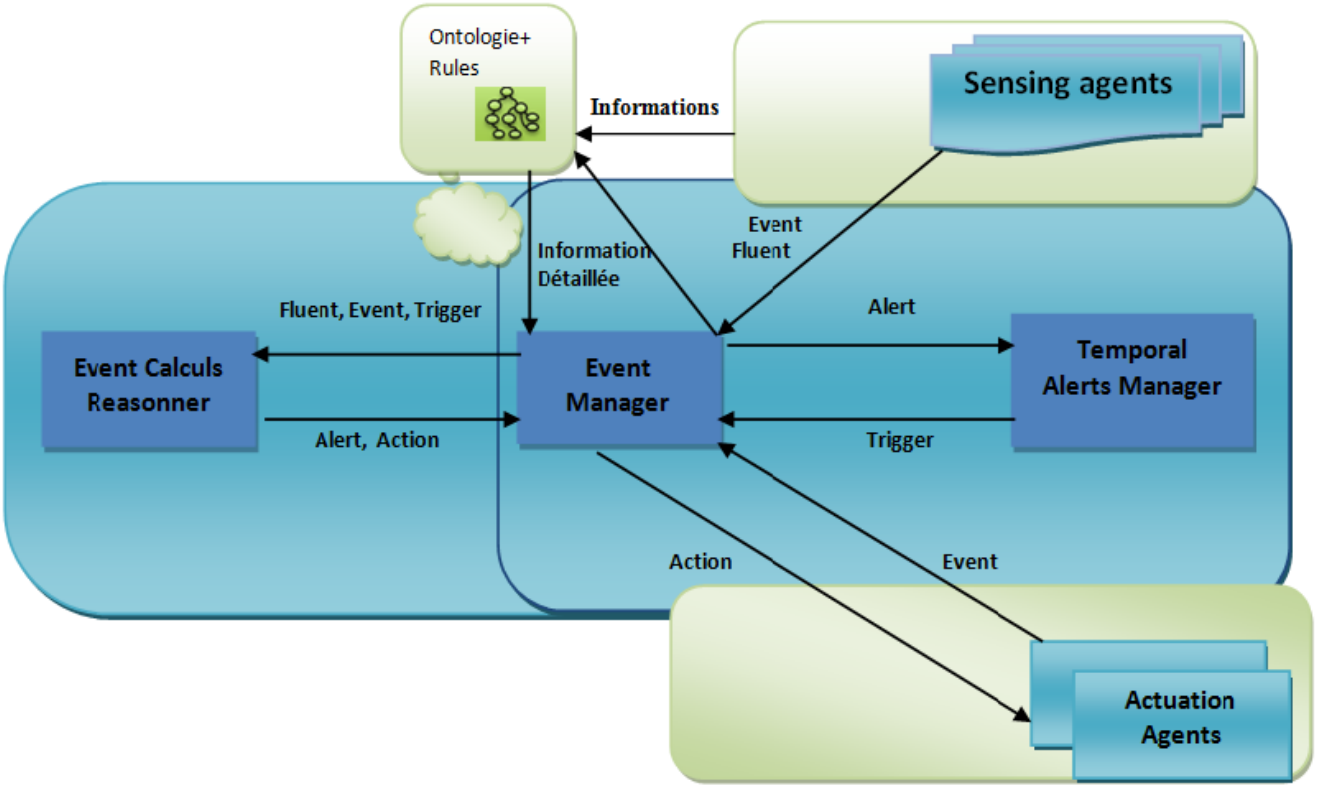


FIGURE 4.1 – La modélisation et le raisonnement effectués sur la base de connaissance.

Event Calculus Reasoner :Cerveau de l'infrastructure, il est l'agent responsable de les décisions des différents événements qu'il reçoit des autres agents, les environnements sont décrits en utilisant les prédicats et fluent de EC, la prise de décision est réalisé en envoyant différentes actions à exécuter a l'agent d'activation ou par l'activation des alertes et de les envoyer à l'Alert Manager.

Event Manager(gestionnaire d'évenement) : Coeur de l'infrastructure, il joue le rôle d'un passage pour relier les agents différents, cet agent est très important par ce qu'il permet une architecture très flexible, en effet il simplifie le développement de nouveaux agents puisque tous les agents communiquent seulement avec lui d'une manière directe, si un agent veut envoyer un message à un autre agent, il traverse l'Event Manager et son rôle n'est pas limité à cela, il permet aussi de générer un ensemble de données dans laquelle sont enregistrées toutes les épreuves produites à partir de l'exécution d'un des scénarios réels sur la plate-forme, ainsi nous pouvons reproduire le même comportement à tout moment pour enquêter sur toute activité.

Pour chaque événement tout est sauvegardé, l'événement lui-même, la date de livraison et l'agent qui l'a envoyé. Il permet également le filtrage des événements lors de la navigation.

En tant que gestionnaire d'événements est la passerelle qui relie les agents, tous les message passent à travers elle.

Sensing agent (agent de détection) : Responsable de la collecte d'informations à partir de différents capteurs, le cycle d'exécution de cet agent est décrit comme suit : quand il reçoit une information d'un capteur donné, il la compose dans un message de calcule d'évenement, et il l'envoie au raisonneur via l'Event Manager.

Events Simulator(simulateur D'événements) : Simule le rôle de l'agent de détection, à partir d'un ensemble de données il récupère les événements et leurs dates d'envoi au gestionnaire d'événements, il permet par ce comportement de reproduire une situation réelle.

Alerts Service (service d'alerte) :Cet agent est très particulier, il permet de gérer le système d'alertes mis en place. Après avoir reçu une alerte de raisonneur, il génère un déclencheur qui envoie au raisonneur après un temps défini par ce dernier, le directeur des alertes peut également désactiver certaines alertes actives par ordre de raisonneur, ceci est possible grâce aux noms des alertes.

Actuation agent (agent d'actionnement) : Exécute les actions reçues de raisonneur, cette agent se compose de plusieurs autres agents, par exemple un haut-parleur pour traduire les messages reçus à la voix, un agent displayer, son rôle comme son nom l'indique, est d'afficher une sorte de message.

Ontologies : Elle a les principaux objectifs suivants :

- Représentation sémantique de l'environnement ambiant.
- Résoudre le probléme de l'hétérogénéité des données de capteurs en transformant tout en : Ontologies
- Raisonner en utilisant des règles et en prenant en compte les relations entre les prédicats (propriétés et héritage de classes)

4.3.2 La communication entre les différents composants de l'architecture :

L'agent de détection collecte les informations à partir de différents capteurs ; il les composent dans un message de calcule d'évenement et les envoient à l'Event Manager,ces informations seront aussi représentées avec l'éditeur d'ontologie à qui l'agent de detection a une référence pour ces classes et ces relations.

L'Event Manager quand il reçoit le message de l'agent de détection, il accède à l'ontologie pour laquelle il a une référence pour détailler les événements reçus et pour extraire les informations qui ont une relation avec ces événements, dans ce cas les règles SWRL sont appliquées sur

les ontologies pour extraire les informations détaillées telles que le nom, l'âge et l'identifiant de la personne. Et quand ces informations seront reçues par l'Event Manager il les traitent et il les composent dans un message sous forme des prédicats de Event Calculs et il les envoient à ce dernier qui permet de raisonner sur elles, et le résultat de ce raisonnement c'est des actions qui envoient à l'agent d'actionnement ou à l'alerts service à travers l'event manager.

L'alerts service et l'agent d'actionnement exécutent les actions reçues de l'event calculs,telles que déclenchement d'alerte,activer la caméra,afficher un message ...

4.4 Conclusion

Dans ce chapitre, nous avons exposé en premier notre proposition et ses détailles. En second, nous avons présenté l'architecture qui permet de l'appliquer. Dans le chapitre suivant, nous allons proposé un scénario qui peut étre exécuté sur notre plate-forme,et nous allons ainsi appliquer la modélisation et le raisonnement sur ce scénario.

CHAPITRE 5

VALIDATION

5.1 Introduction

Après l'étude de la possibilité de combinaison entre les deux approches de modélisation Event Calculus et Ontologie, nous avons trouvé des avantages évident en ce qui concerne la modélisation et la représentation des connaissances avec les ontologies et le raisonnement avec Event Calculs, dans ce qui suit, nous allons vous présenté notre scénario proposé pour valider notre proposition , les étapes de la modélisation avec les ontologies et le raisonnement avec Event Calculus.

5.2 Proposition d'un scénario de supervision en utilisant une description du domaine avec une ontologie :

Pour implémenter notre proposition nous avons proposé le scénario de supervision suivant pour montrer et illustrer le genre de scénario qui peut être exécuté sur notre plate-forme.

Si Personne non autorisée **alors**
Déclencher caméra
Message vocale : " Vous n'êtes pas autorisée à y accéder "
Si la personne force l'entrée **alors**
Déclencher alarme.
Si personne autorisée **alors**
Ouvrir la porte
Déclencher caméra (la caméra ne s'éteint pas jusqu'à la fermeture de la porte)

Fermer porte.

Personne autorisée est rentrée alors

Si détection mouvement (Zmove) et non tag (Lecteur Rfid ne détecte pas de tag) **alors**

Personne inconnue

Déclencher Alarme

Fermer les portes et les fenêtres

Localiser la personne

Allumer lumière

Activer caméra

Sinon tag identifié et personne connue

Allumer lumière

Limiter l'accès aux armoires : placer un tracient devant chaque armoire.

Une personne veut ouvrir une armoire elle passe son tag devant le Lecteur RFID Tracient

Si personne autorisée **alors**

Déclencher caméra

Ouvrir armoire

Sinon personne non autorisée

Bloquer l'accès

5.3 Choix d'outils pour le traitement des ontologies OWL et des règles SWRL

Une des raisons du succès d'OWL et de Event calculs est l'existence de nombreux outils pour la gestion des ontologies. En effet, on dispose de bibliothèques, d'interfaces de programmation (APIen anglais), d'éditeurs, de moteurs d'inférence et de règles, etc. qui facilitent la création et l'édition d'ontologies et de règles, leur accès depuis un programme, l'exécution du raisonnement et le traitement de règles. De plus, une grande partie de ces logiciels ont des licences libres, ce qui permet de les obtenir, de les étudier, de les modifier et de les partager plus facilement.

5.3.1 Édition d'ontologies

Pour la création et l'édition d'ontologies nous avons choisi l'éditeur Protégé [71], développé par l'Université de Sanford en collaboration avec l'Université de Manchester, est le standard de facto pour la création et l'édition d'ontologies OWL. Son code source, libre, est écrit en Java, et il admet des extensions sous forme de plugins. Il existe de nombreux plugins, par exemple pour la visualisation des ontologies et pour l'édition des règles SWRL associées.

5.3.2 APIs pour le traitement des ontologies OWL

Ces bibliothèques logicielles permettent un accès par programme aux ontologies OWL ; elles fournissent des fonctions qui permettent de créer une ontologie, de la lire, de créer le modèle en mémoire correspondant, de le modifier, de l'enregistrer, etc. La plupart de ces bibliothèques sont implantées avec le langage Java.

On a choisi la bibliothèque Protégé-OWL API qui est un plugin de Protégé qui est utilisé pour l'accès par programme aux ontologies OWL.cette API permet aux programmeurs en JAVA de développer des applications qui peuvent accéder aux bases de connaissances de Protégé. Cette API fournit des packages et des classes JAVA pouvant effectuer des opérations complexes. L'interface entre les programmes et les projets de bases de connaissances de Protégé se fait en utilisant la classe "edu.standford.smi.protege.model.Project" qui se trouve dans le package "protege.jar" fournit avec Protégé. Cette classe possède une méthode getKnowledgeBase () permettant d'accéder au contenu de la base de connaissances.

5.3.3 Moteurs d'inférence

Ces logiciels sont capables, par déduction, d'extraire des connaissances implicites contenues dans une ontologie. Ils peuvent s'exécuter comme une application indépendante ou être appelés depuis un autre programme, notamment les APIs mentionnées ci-dessus. Il existe des implantations propriétaires, telles que Bossam ou RacerPro, ainsi que d'autres libres comme Pellet, Fact++, KAON ou HermiT1. Parmi ces outils nous avons choisi Pellet qu'est celui qui rencontre le plus de succès actuellement, grâce à ses performances, à ses fonctionnalités et à sa conception claire et simple.

5.4 Outils pour le traitement des règles Event Calculus

5.4.1 Jess

Afin de traiter les règles EC, il est possible d'utiliser des moteurs spécifiquement dédiés aux règles, tel que le moteur Jess71p2. Ce moteur de règles possède un langage propre pour l'expression des connaissances sous forme de règles.

5.5 Modélisation de l'ontologie pour écrire les règles de production

Pour la réalisation de notre scénario, nous avons besoin d'un certain nombre de concepts qui sont une représentation abstraite des objets du monde réel. Pour cela, nous avons proposé une ontologie qui définit les différents concepts dont nous aurons besoin pour écrire notre scénario (Figure 5.1)

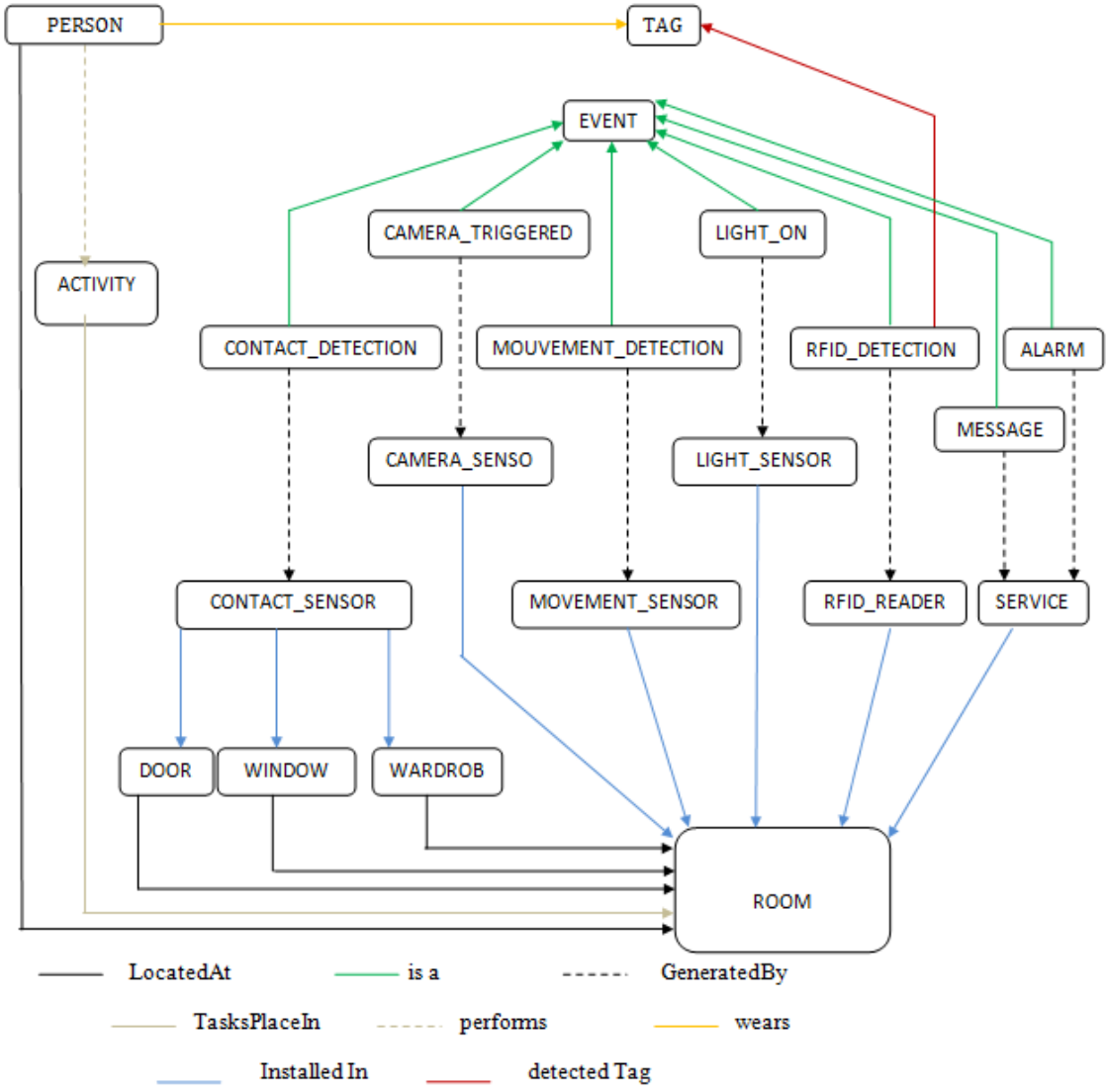


FIGURE 5.1 – Ontologie de notre scénario proposée.

Notre ontologie contient les concepts de base suivants :

- **PERSON :** représente la personne qui existe dans l'environnement à surveiller, elle porte un tag pour l'identifier et a les propriétés " hasName " de type string, elle performe une activité, et elle est localisé à l'intérieur d'une chambre.

- **TAG :** Il a la propriété " hasID " qui prend comme valeur l'identificateur de ce tag, et la propriété " isActive " de type booléen pour indiquer que le tag a été détecté par un lecteur RFID actif (isActive=True) ou bien par un tracient (isActive= false).
- **RFID−READER :** C'est le lecteur du tag, il est installé dans une chambre à un endroit précis et il peut nous informer des tags détectés.
- **RFID-DETECTION :** Un évènement qui se produit suite à la détection d'un tag au niveau d'un lecteur RFID.
- **MOVEMENT-SENSOR :** un type de capteur Cléode (appelé ZMove) qui détecte un mouvement, il est installé dans une chambre à un endroit précis.
- **MOUVEMENT-DETECTION :** Un évènement qui se produit suite à la détection d'un mouvement au niveau d'un capteur ZMove.
- **CONTACT-SENSOR :** Un autre type de capteur Cléode qui détecte l'ouverture/fermeture de portes, il est installé devant une porte ou une armoire qui se trouve dans une chambre a un endroit bien précis.
- **CONTACT-DETECTION :** Un évènement qui se produit suite à la détection d'ouverture ou fermeture de porte.
- **CAMERA-SENSOR :** C'est un type de capteur , qui déclenche la camera ,il est installé dans une chambre à un endroit précis .
- **CAMERA-TRIGGERED :** Un évènement qui se produit suite a la détection d'un événement à la présence d'une personne inconnue.
- **LIGHT-SENSOR :** Un autre type de capteur qui peut allumer/éteindre la lumière, il est installé dans la chambre.
- **SERVICE :** c'est le system installé dans la chambre, qui génère un message vocale " Vous n'êtes pas autorisée à y accéder " pour les personnes non autorisé, et des alerte pour des personnes qui force l'entrée.
- **MESSAGE :** Un évènement qui se produit suite à la détection d'une personne non identifier.
- **LIGHT-ON :** Un évènement qui se produit suite a la détection d'un tag d'une personne connu.
- **ALARM :** Evènement qui se produit quand une personne force l'entrée.

5.5.1 Quelques exemples du code de notre base de connaissance en logique de description

TBOX :

LIGHT-ON $\equiv$ EVENT $\wedge$ $\exists$ generatedBy.LIGHT$_S$ENSOR

CAMERA-TRIGGERED $\equiv$ EVENT $\wedge$ $\exists$ generatedBy.CAMERA

CONTACT-DETECTION ≡ EVENT ⋀∃ generatedBy.CONTACT$_S$ENSOR
MOUVEMENT-DETECTION ≡ EVENT ⋀∃ generatedBy.MOUVEMENT$_S$ENSOR
RFID-DETECTION ≡ EVENT ⋀∃ generatedBy.RFID$_R$EADER
MESSAGE ≡ EVENT ⋀∃ generatedBy.SERVICE
PERSON≡ ∃ hasName ⋀ ∃ wears.TAG ⋀∃ performs.ACTIVITY ⋀∃ lokatedIn.ROOM
ABOX :
PERSON(djafar)
ACTIVITY(activity1)
TAG(10)
ROOM(room1)
Wears(djafar,10)
Performs(djafar,activity1)
Lokated(djafar,room1)

5.5.2 Les étapes de l'édition de notre ontologie sous PROTÉGÉ-OWL

1. **Création d'un nouveau projet :** Lors du premier démarrage de PROTÉGÉ sous Windows, une boite de dialogue 'Welcome to PROTEGE' s'ouvre. Afin de créer un nouveau projet OWL, nous devons cliquer sur le bouton de création d'un nouveau projet " New Project...". Cette action génère une autre boite de dialogue 'Create New Project' contenant différent type du projet à choisir. Pour ce qui est de notre cas, il s'agit d'un fichier OWL " OWL Files (.owl or .rdf) ". Une fois que ce choix est validé, il faut préciser le langage avec lequel sera éditée l'ontologie, il s'agit d'OWL DL. Par la suite, l'interface de l'outil s'affiche (voir Figure 5.2) permettant d'éditer, de visualiser et d'enregistrer des ontologies en OWL DL.

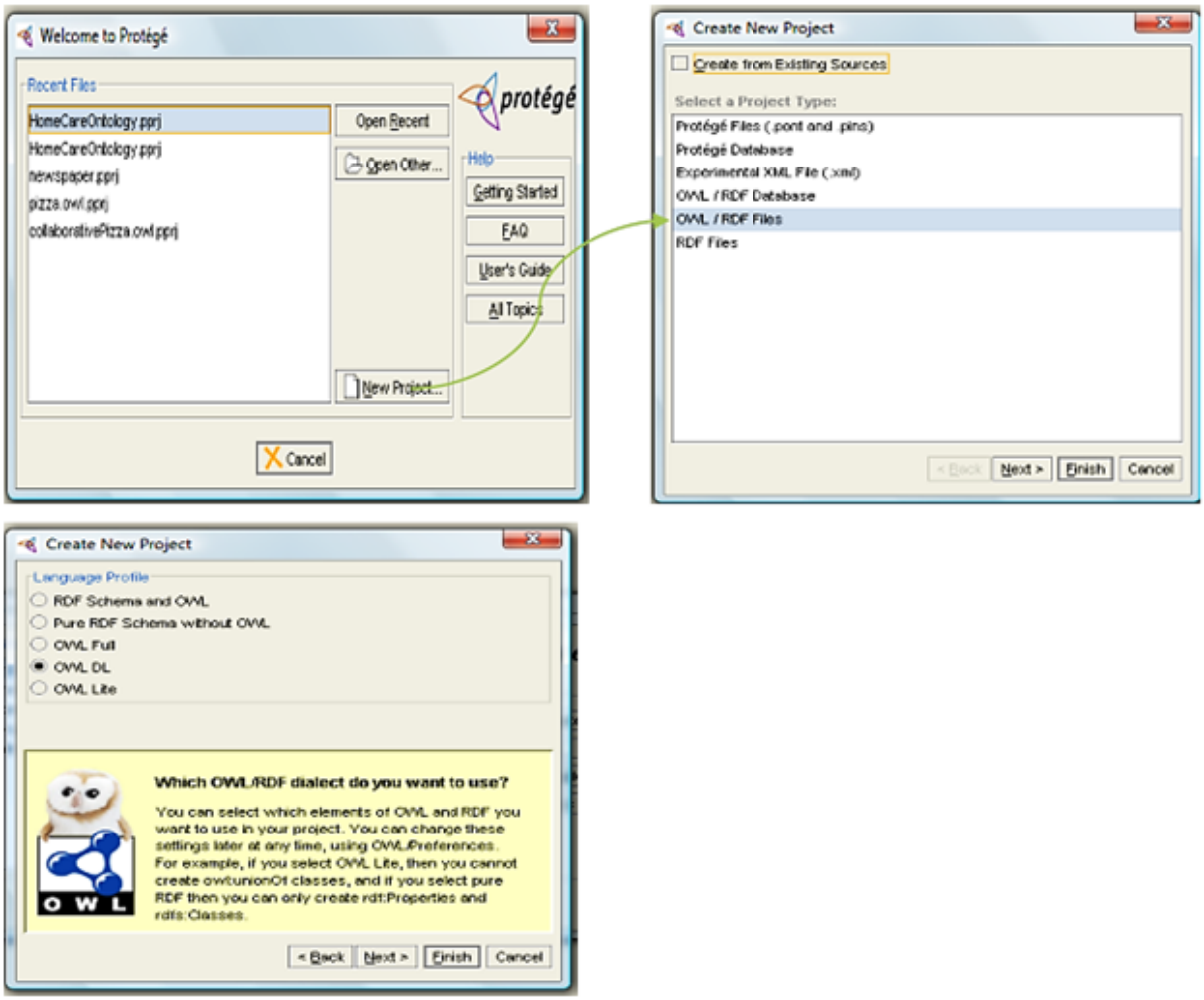


FIGURE 5.2 – Création d'un nouveau projet sous Protégé.

Comme illustré dans la figure 5.3, l'interface utilisateur de PROTÉGÉ-OWL plugin fournit un ensemble d'onglets. Les trois onglets les plus importants que nous allons utiliser par la suite pour l'édition des composants de l'ontologie sont :

- **L'onglet OWL Classes :** affiche la hiérarchie de classes de l'ontologie. Il permet aux développeurs de créer et d'éditer les classes et affiche le résultat de la classification.
- **L'onglet Propretés :** est utilisé pour créer et éditer les propriétés de l'ontologie.
- **L'onglet Individuals :** est utilisé pour créer et éditer les instances.

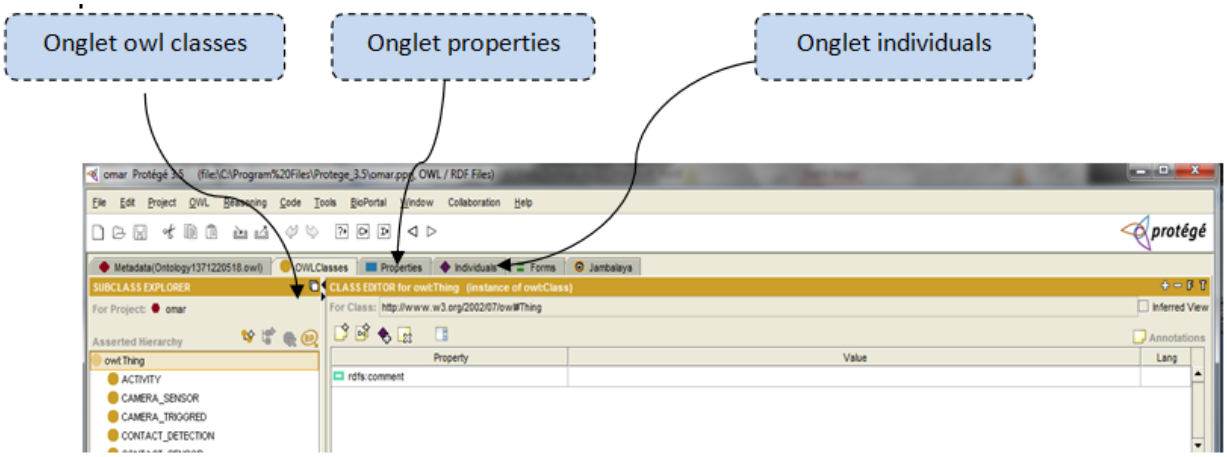


FIGURE 5.3 – Interface de PROTÉGÉ OWL.

2. **Création des classes et la hiérarchie des classes :**

   Une classe universelle (owl : thing) est utilisée comme racine pour cette hiérarchie, et la création des sous-classes se fait par le choix de la classe mère, suivi par un simple clic sur le bouton de création des sous-classes de la classe sélectionnée (voir figure 5.4).

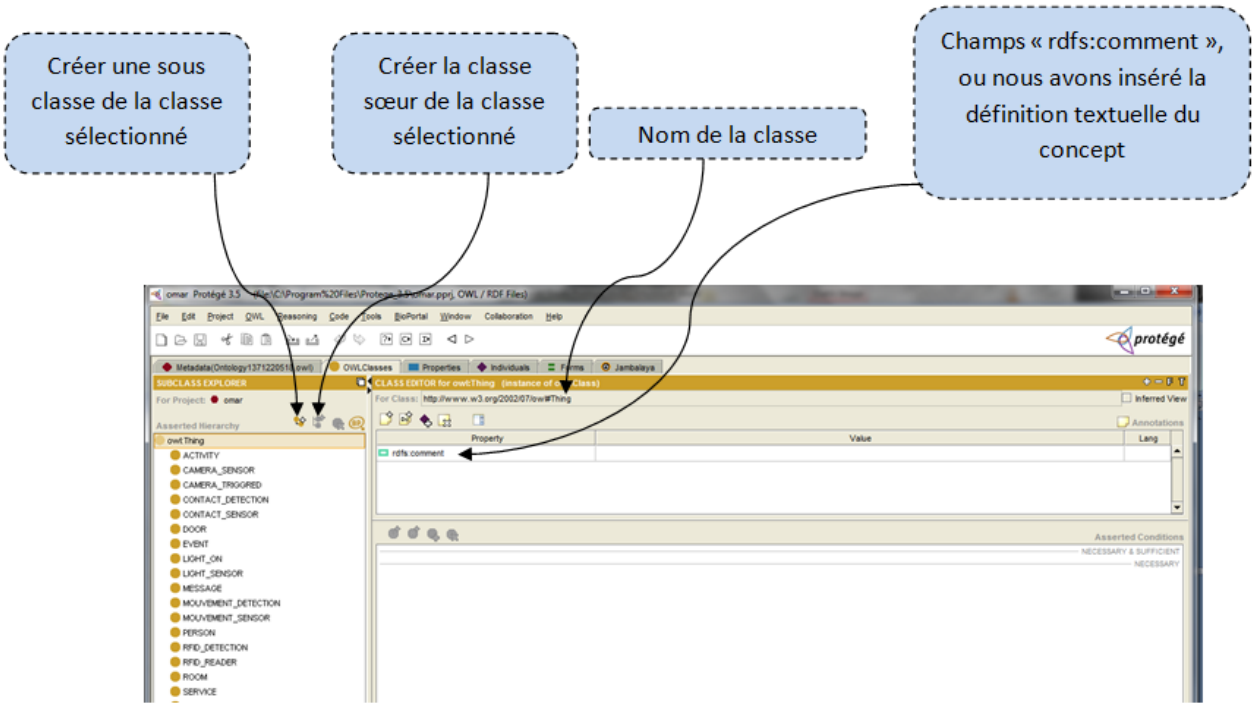


FIGURE 5.4 – Édition des concepts.

3. **Création des propriétés (Slots) :**

   Après avoir construit les classes, nous créons maintenant les propriétés (sont appelées rôles en logique de description) pour chacune d'elles en utilisant l'onglet Properties. Il existe deux types de propriétés, les attributs 'Datatypeproperties' et les relations 'Object properties'. Les propriétés d'une classe sont les propriétés héritées de sa superclasse, plus ses propres propriétés privées. On peut enrichir la sémantique des propriétés lors de la création de ces derniers, en leur spécifiant les caractéristiques suivantes :

   - **Functional property :** si une propriété est fonctionnel pour un individu donné, il peut exister au plus un individu qui est relié à l'autre individu par cette relation.
   - **Inverse property :** si une propriété P a sa propriété inverse, et P relie un individu A à un individu B, alors la propriété inverse relie l'individu B à l'individu A.
   - **Inverse functional property :** si une propriété est inverse functional, ça veut dire que la propriété inverse de cette propriété est fonctionnelle.
   - **Transitive property :** si une propriété P est transitive et P relie l'individu A à l'individu B et l'individu B à l'individu C, alors on peut déduire que A est relié à C par P.
   - **Symetric property :** si une propriété P est symetrique, et P relie l'individu A à l'individu B alors, B est relié à A par P.

   La figure 5.5 montre comment on peut créer ces propriétés.

a)Création des attributs (dataTypeProperty)

Un attribut permet de relier des individus à des valeurs de données. Une valeur étant un élément d'un des types de données prédéfinis de XMLSchema (float, string, etc.).Pour créer un attribut d'une classe, il faut cliquer sur le bouton de création d'une nouvelle propriété 'dataTypeProperty' . Cette action génère une fenêtre comme la montre la figure 5.6 ; où on doit spécifier le nom de l'attribut, le domaine, le type XML associé, les valeurs permises s'ils ont connues ainsi que le type de valeurs de l'attribut (la seule option qui est affiché pour un attribut est fonctionnel).

b)Création des relations (ObjectProperty)

Une relation permet de relier des individus à d'autres individus. Pour créer une relation entre deux classes, il faut cliquer sur le bouton de création d'une propriété, 'objectProperty' . Cette action permet de générer une nouvelle fenêtre (voir la figure 5.7) différente de celle des attributs où on spécifie le nom de la relation, le domaine, le co-domaine, la relation inverse si elle existe ainsi que les caractéristiques de la relation (functional, symétrique, transitive,). Ils peuvent être hiérarchisés en utilisant l'onglet (Properties).

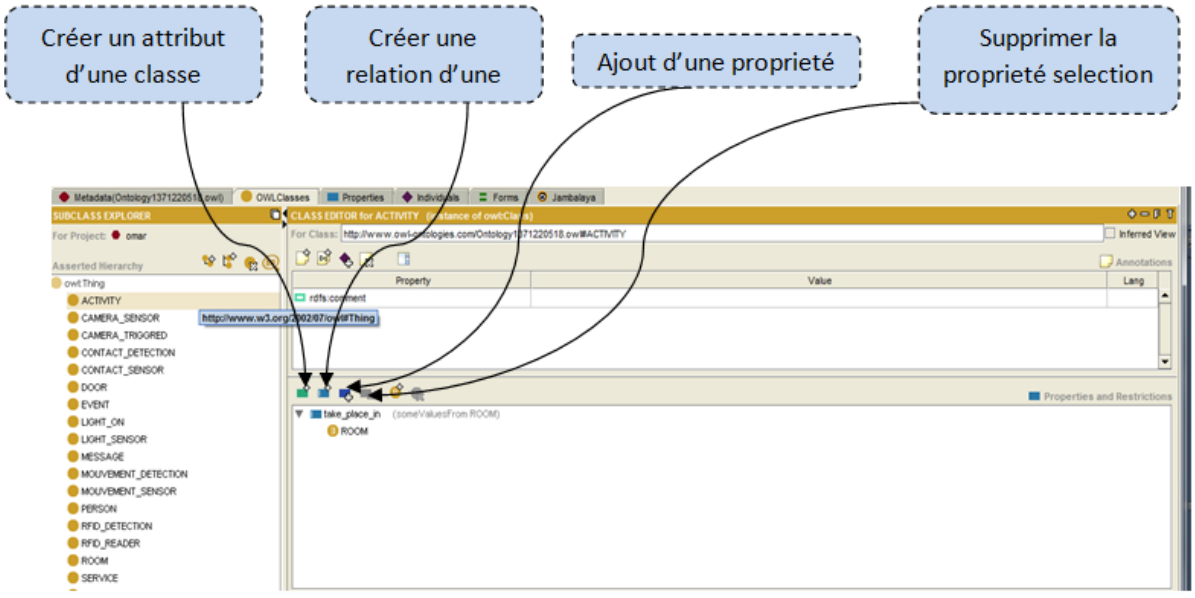


FIGURE 5.5 – Création de propriétés pour une classe.

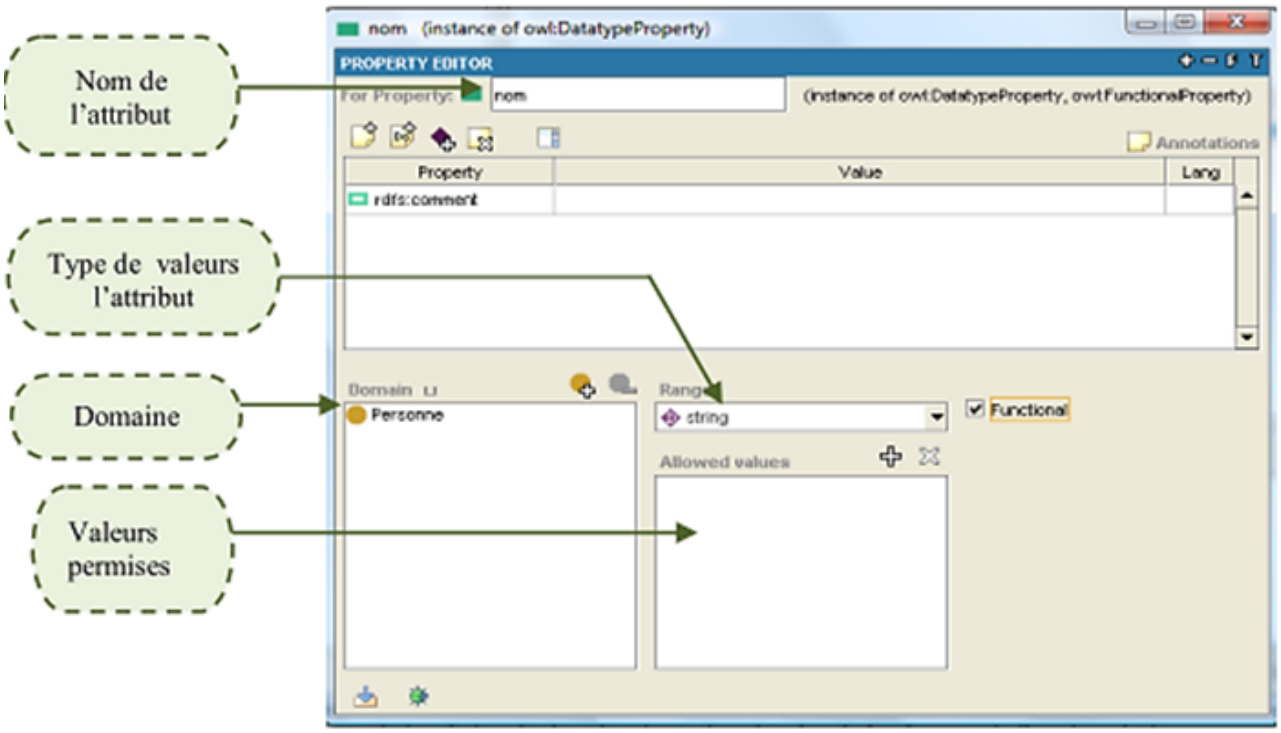


FIGURE 5.6 – Création d'un attribut.

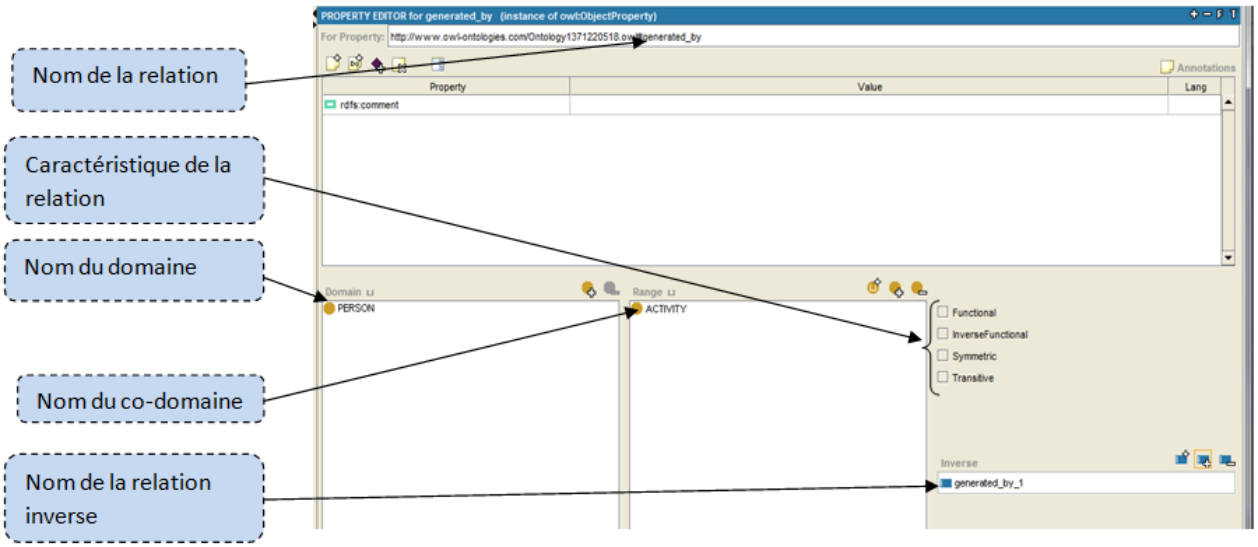


FIGURE 5.7 – Création d'une relation entre deux classes.

4. **Restrictions sur les propriétés**

   Les restrictions sont utilisées pour décrire ou définir une classe c.-à-d. Restreindre les individus appartenant à une classe. En OWL DL, il existe deux types de restrictions de propriétés :

   Les contraintes de valeurs (values contraints) : $\forall$,$\exists$

   Les contraintes de cardinalités (cardinality contraints) : >,<, =.

   Pour créer une restriction sur une propriété d'une classe en utilisant PROTÉGÉ, il suffit de sélectionner la classe qu'on veut décrit, puis cliquez au-dessus du bouton , la fenêtre montré dans la figure 5.8 s'affichera.

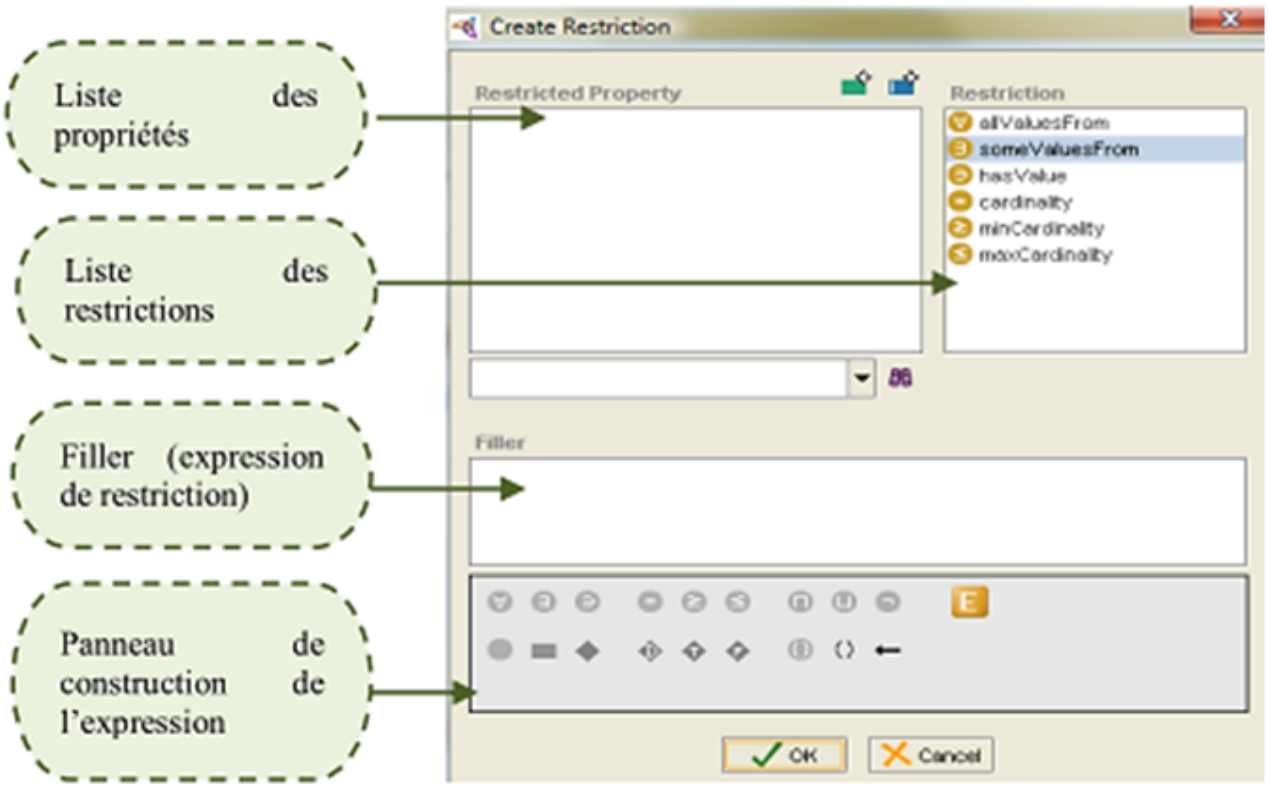


FIGURE 5.8 – Création d'une restriction.

Une fois la fenêtre s'affiche, vous sélectionnez la propriété sur laquelle on va faire la restriction à partir de la liste de propriétés, puis choisissez le type de la restriction (Cardinality, minCardinality, maxCardinality, hasValue, allValuesFrom, someValuesFrome) de la liste de restriction, et enfin spécifier le filler pour la restriction dans la zone 'filler' ou bien utiliser le panneau de construction de l'expression.

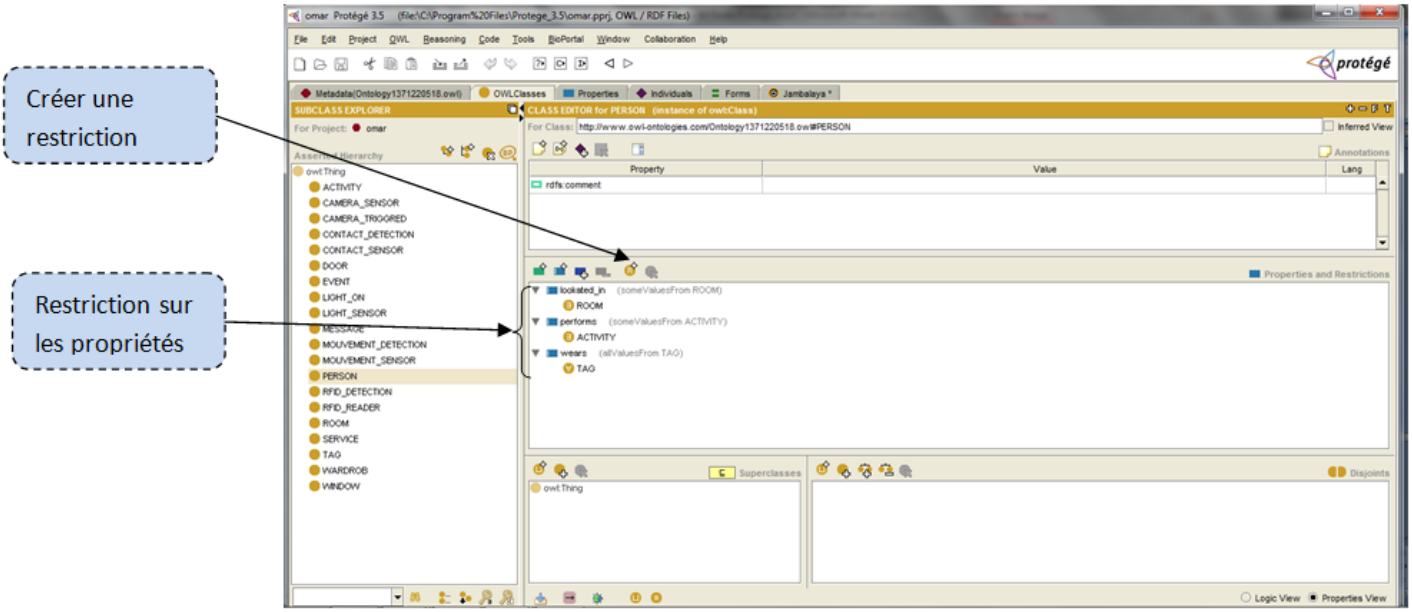


FIGURE 5.9 – Les restrictions crées sur les propriétés d'une classe spécifiée.

5. **Création des instances**

Une dernière tâche était la saisie des instances des classes de la hiérarchie dans les formulaires que nous a produit Protégé sous l'onglet 'Individuals'. Pour chaque instance d'une classe sélectionnée (la classe à instancier), on doit spécifier le nom de l'instance ainsi que les valeurs des propriétés (les valeurs des attributs et/ou les noms des instances avec lesquelles cette instance est reliée par une relation) comme illustré dans la figure 5.10.

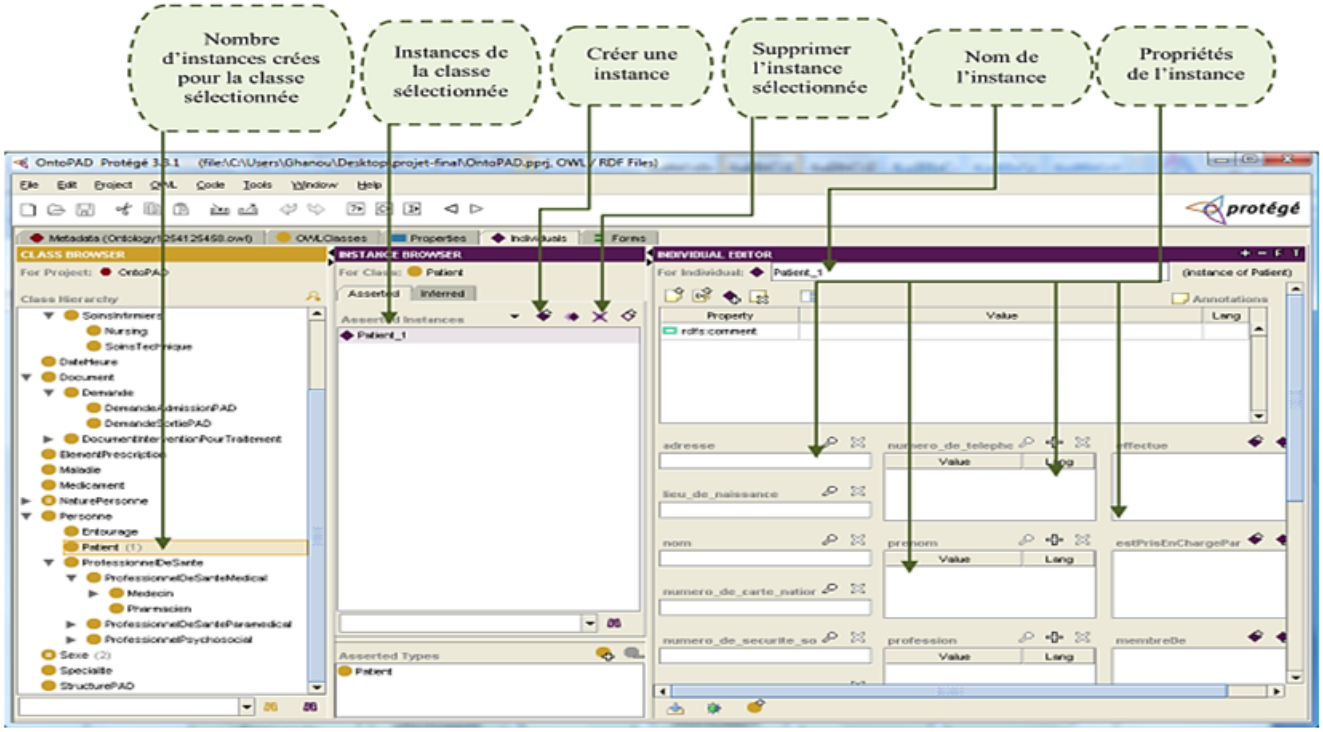


FIGURE 5.10 – Création des instances.

6. **Visualisation de notre ontologie :**

   Pour visualiser notre ontologie crée sous protégé on clique sur l'onglet Jambalaya, comme illustré dans la figure 5.11 .

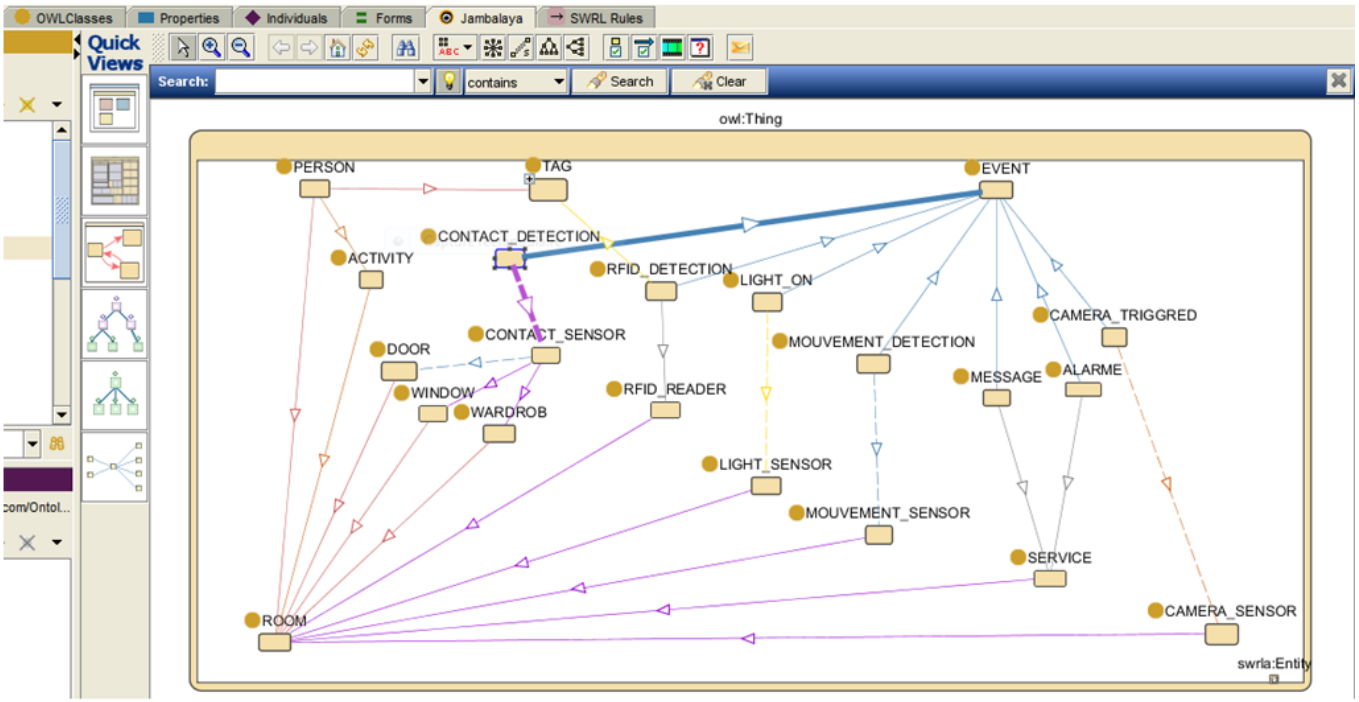


FIGURE 5.11 – L'ontologie crée.

7. **Génération du code**

   L'outil PROTEGE OWL a été conçu pour dégager et libérer le développeur de la complexité du codage, même pour implémenter une petite ontologie, celle-ci va prendre plusieurs lignes du code et nécessite un grand effort. Le code de notre ontologie sera donné à l'annexe A de ce mémoire.

8. **Appliquation des régles SWRL sur l'ontologie**

   L'interogation de l'ontologie se fait avec des regles SWRL.

   Exemple :

   - la régle 1 est pour afficher tous les noms des personnes comme illustré dans la figure 5.12.
   - la régle 2 est pour afficher les identifiants des personnes comme illustré dans la figure 5.13.

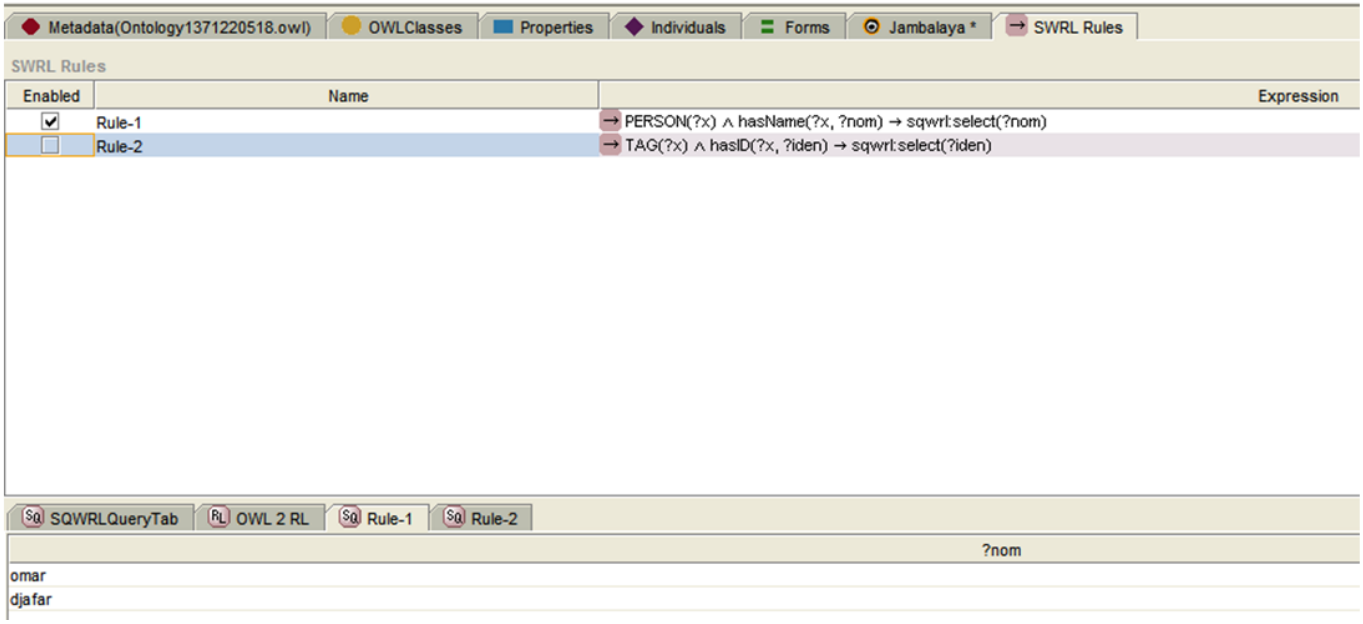


FIGURE 5.12 – Afficher les noms des personnes.

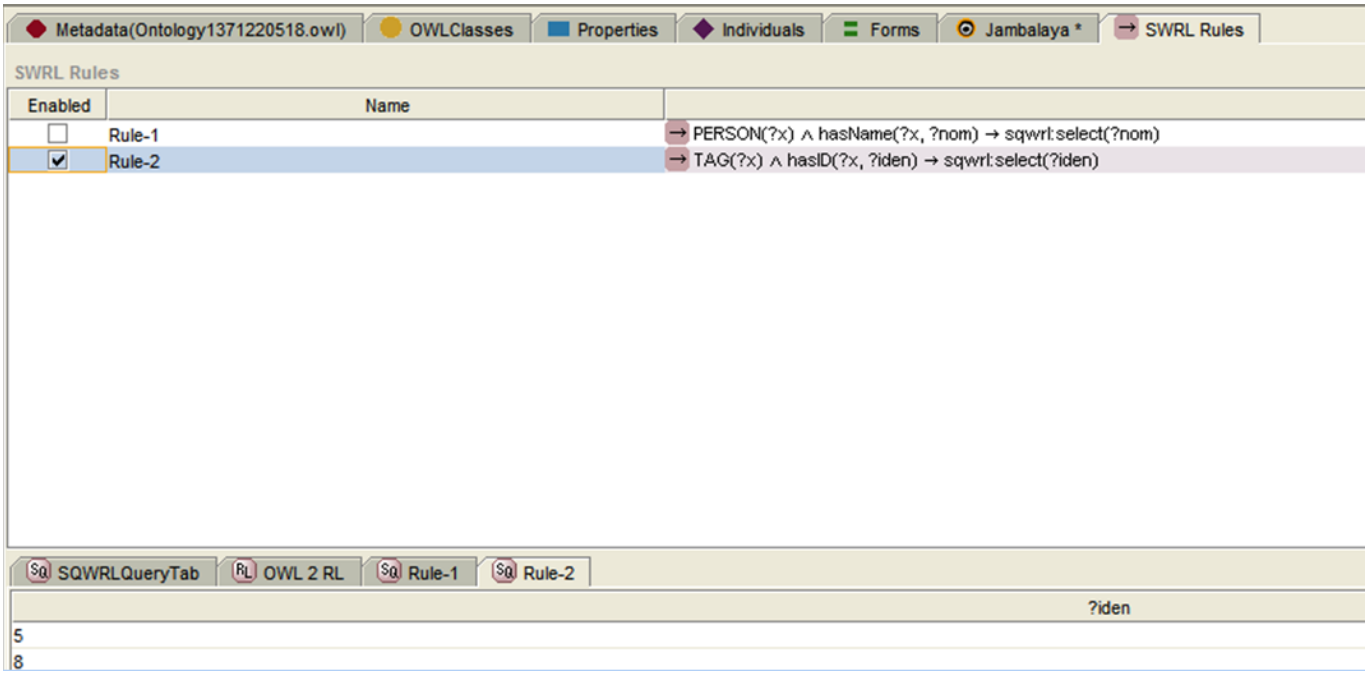


FIGURE 5.13 – Afficher les identifiants des personnes .

5.6 Le raisonnement avec Event Calculus

1. **Cas où, personne non reconnue et qui essaye de forcer la porte :**

   Le résultat si une personne a un Tag nonreconnue et essaye de'ouvrir la porte comme même après lui avoir dit qu'elle n'est pas autorisée à rentrer.

   Voici les prédicats correspondants :

   HoldsAt (TagDetected(tag1),1).

   ¬HoldsAt(RfidReader(room1), 1).

   HoldsAt(CameraSensor(camera1), 1).

   HoldsAt(ContactSensor(door1), 1).

   ¬HoldsAt(MovementSensor(room1), 1).

   ¬HoldsAt(LightOn(lightSensor1),1).

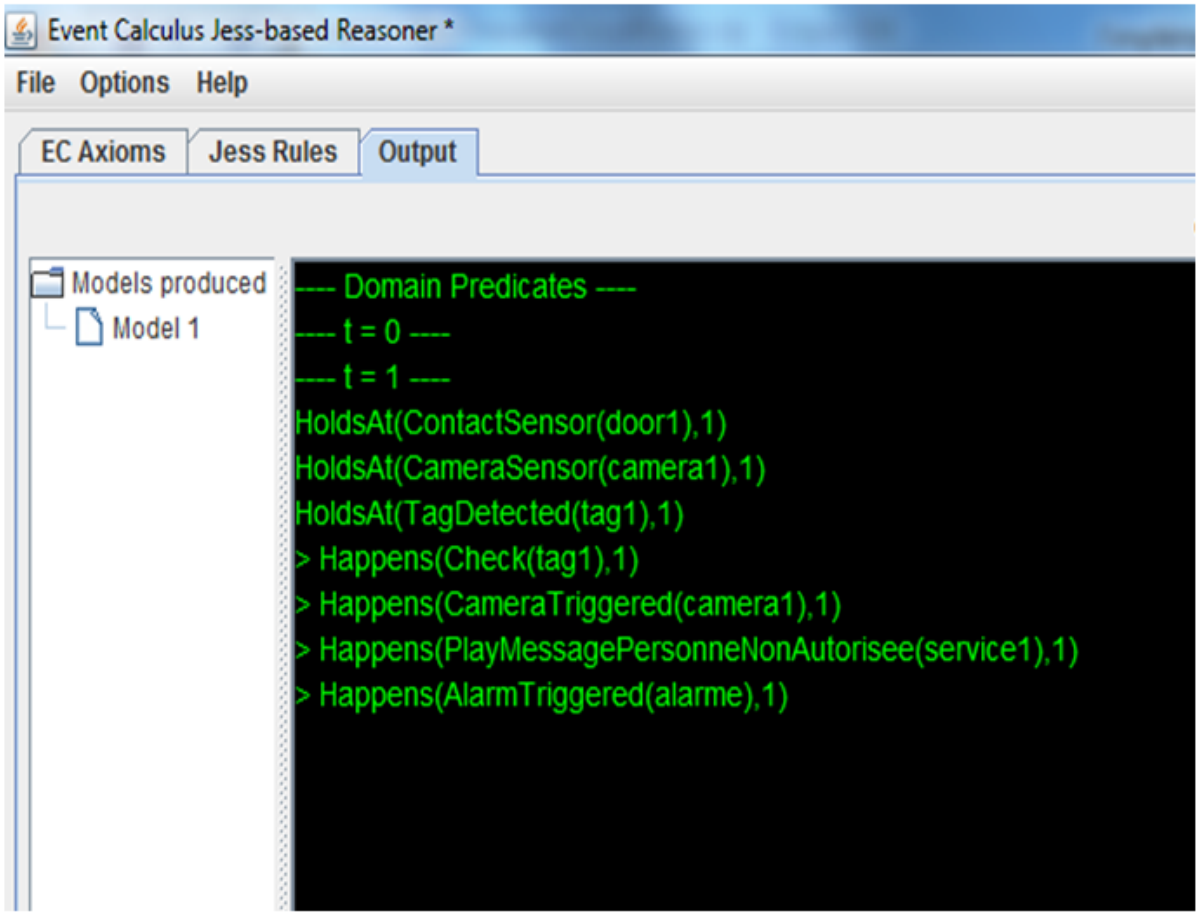


FIGURE 5.14 – Résultat d'éxécution si personne non reconnue et qui essaye de forcer la porte.

2. **Cas où, personne reconnue et qui ouvre l'armoire :**

   Voici les prédicats correspondants :

   HoldsAt (TagDetected(tag1),1).

   HoldsAt(RfidReader(room1), 1).

   HoldsAt(CameraSensor(camera1), 1).

   HoldsAt(ContactSensor(door1), 1).

   HoldsAt(MovementSensor(room1), 1).

   HoldsAt(LightOn(lightSensor1),1).

   HoldsAt(ContactSensoWardrobe(wardrobe1),1).

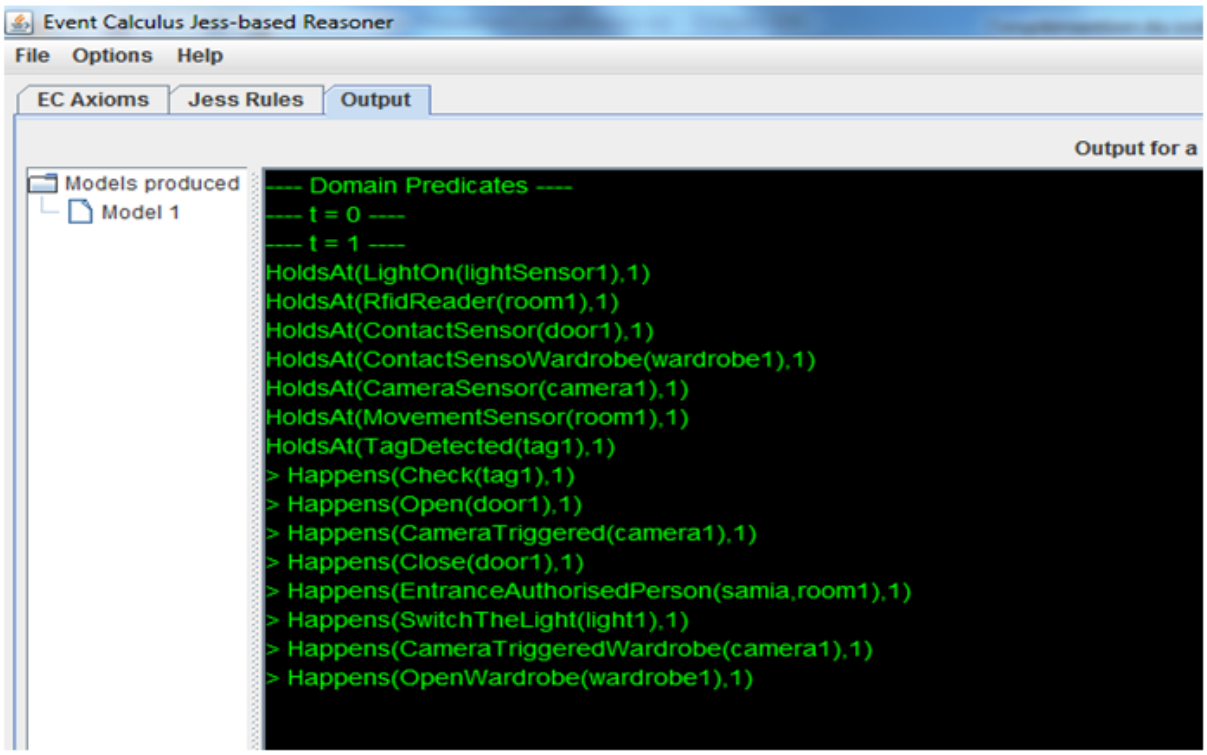


FIGURE 5.15 – Résultat d'éxécution si personne reconnue et qui ouvre l'armoire.

3. **Cas où, personne rentrée par infraction et qui essaye d'ouvrir l'armoire :**

   Voici les prédicats correspondants :

   ¬HoldsAt (TagDetected(tag1),1).

   ¬HoldsAt(RfidReader(room1), 1).

   HoldsAt(CameraSensor(camera1), 1).

   HoldsAt(ContactSensor(door1), 1).

   HoldsAt(MovementSensor(room1), 1).

   HoldsAt(LightOn(lightSensor1),1).

HoldsAt(ContactSensoWardrobe(wardrobe1),1).

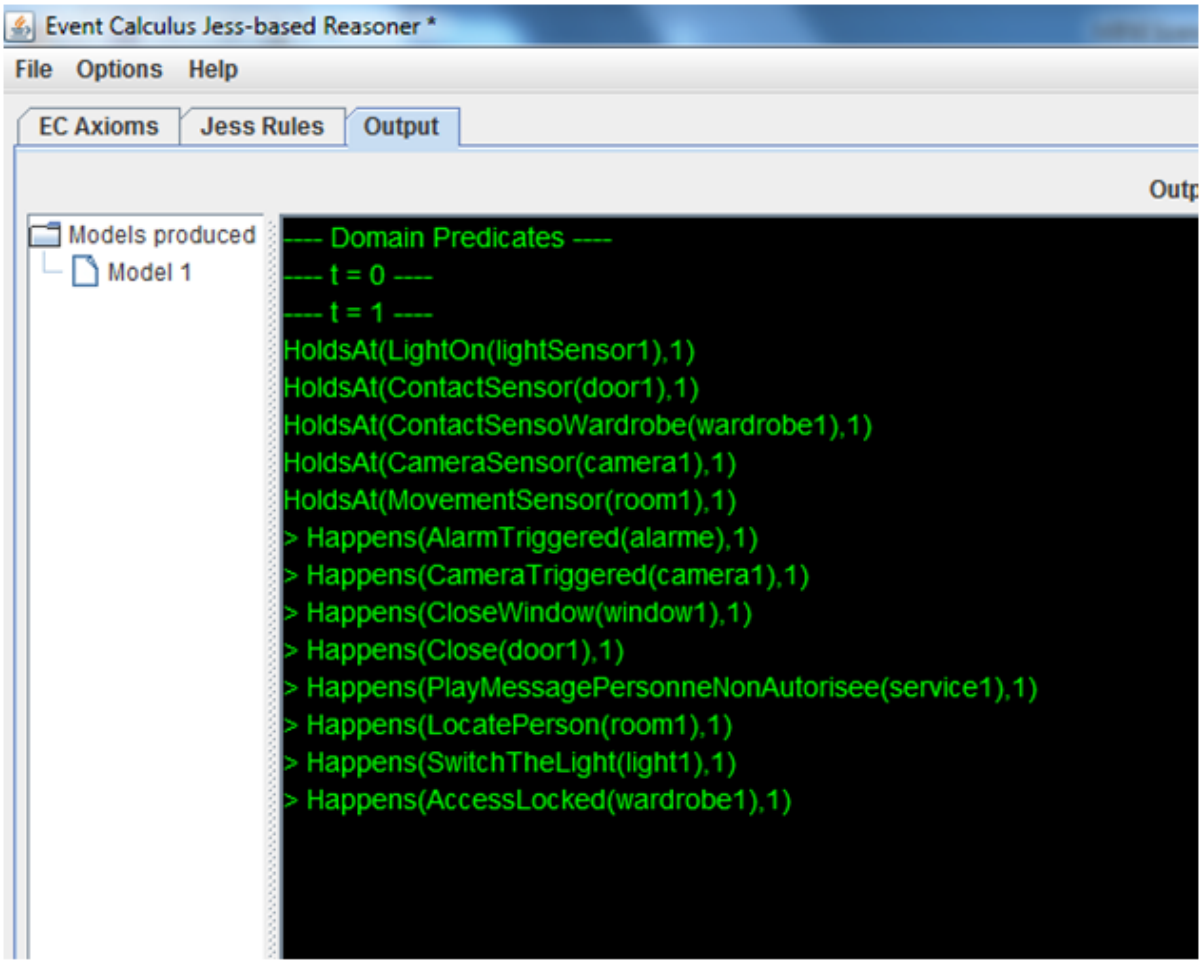


FIGURE 5.16 – Résultat d'éxécution si personne qui est rentrée par infraction.

Pour le code entier ,les déclarations et la partie axiomatisation voir l'annexe B.

5.7 Conclusion

Dans ce chapitre nous avons proposé un scénario qui peut étre exécuté sur notre plate-forme et nous avons essayé d'appliquer la combinaison entre les deux approches de modélisation Event Calculs et ontologie sur ce scénario et tous ça en modélisant notre scénario en utilisant le modéle basé sur les ontologies et en appliquant le raisonnement avec le modéle logique Event Calculs.

CONCLUSION GÉNÉRALE

Les environnements ubiquitaires sont des environnements ouverts et distribués. Ils se caractérisent par leurs hétérogénéités, leur incertitude et leur dynamicité, ces caractéristiques ont conduit à la conception d'applications dites sensibles au contexte.

Les caractéristiques variées des informations de contexte nécessitent des modèles abstraits de description de contexte pour simplifier leur utilisation. Plusieurs travaux ont défini des modèles de description contexte, ces modèles varient selon leur expressivité et le type d'informations de contexte qu'ils permettent de décrire mais aucun d'eux ne peut simultanément satisfaire à toutes les exigences de la modélisation des informations contextuelles.

Afin d'obtenir plus de systèmes souples et généraux ces modèles doivent être combinés les un aux autres ce qu'on appele les modèles hybride. C'est dans le cadre de cette problématique que notre travail de recherche s'est effectué.

D'après les différentes études des travaux menés sur le contexte, la sensibilité au contexte, les exigences de la modélisation et les approches de modélisation existantes et les approches de modélisation hybride ; nous avons pu proposer une combinaison des deux approches de modélisation Event Calculs et ontologie.

L'objectif de notre proposition est de combiner les avantages particuliers du modèle Event Calculs, en particulier le raisonnement en temps réel et la représentation des différents types de négation et de divers types de la représentation et de la description de contexte et l'interopérabilité des modèles ontologiques.

Nous avons aussi présenté l'architecture globale de notre plate-forme et nous avons pu faire une validation empirique par un petit scénario de supervision pour montrer et illustrer le genre de scénario qui peut être exécuté sur la plate-forme, nous avons présenté notre scénario avec les ontologies et effectué le raisonnement avec les règles et prédicats de Event Calculs.

Annexe

Annexe A :

Le code de l'ontologie crée avec l'éditeur protégé :

<?xml version="1.0" ?>

<!DOCTYPE rdf :RDF [

<!ENTITY owl "http ://www.w3.org/2002/07/owl" >

<!ENTITY swrl "http ://www.w3.org/2003/11/swrl" >

<!ENTITY swrlb "http ://www.w3.org/2003/11/swrlb" >

<!ENTITY xsd "http ://www.w3.org/2001/XMLSchema" >

<!ENTITY rdfs "http ://www.w3.org/2000/01/rdf-schema" >

<!ENTITY rdf "http ://www.w3.org/1999/02/22-rdf-syntax-ns" >

<!ENTITY protege "http ://protege.stanford.edu/plugins/owl/protege" >

<!ENTITY xsp "http ://www.owl-ontologies.com/2005/08/07/xsp.owl" >

<!ENTITY swrla "http ://swrl.stanford.edu/ontologies/3.3/swrla.owl" >

<!ENTITY sqwrl "http ://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl" >

]>

<rdf :RDF xmlns="http ://www.owl-ontologies.com/Ontology1371220518.owl"

xml :base="http ://www.owl-ontologies.com/Ontology1371220518.owl"

xmlns :sqwrl="http ://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"

xmlns :rdfs="http ://www.w3.org/2000/01/rdf-schema"

xmlns :swrl="http ://www.w3.org/2003/11/swrl"

xmlns :protege="http ://protege.stanford.edu/plugins/owl/protege"

xmlns :xsp="http ://www.owl-ontologies.com/2005/08/07/xsp.owl"

xmlns :owl="http ://www.w3.org/2002/07/owl"

xmlns :xsd="http ://www.w3.org/2001/XMLSchema"

xmlns :swrlb="http ://www.w3.org/2003/11/swrlb"

xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns"

xmlns :swrla="http ://swrl.stanford.edu/ontologies/3.3/swrla.owl">

<owl :Ontology rdf :about="">

<owl :imports rdf :resource="http ://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>

<owl :imports rdf :resource="http ://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>

</owl :Ontology>

<swrl :Variable rdf :ID="nom"/>

<swrl :Variable rdf :ID="x"/>

<owl :Class rdf :ID="ACTIVITY">

<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="take-place-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="ALARME">
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="generated-by"/>
<owl :allValuesFrom rdf :resource="SERVICE"/>
</owl :Restriction>
</rdfs :subClassOf>
</owl :Class>
<owl :Class rdf :ID="CAMERA-SENSOR">
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
</owl :Class>
<owl :Class rdf :ID="CAMERA-TRIGGRED">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>

```
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
<owl :versionInfo rdf :datatype="xsd ;string">TODO :</owl :versionInfo>
</owl :Class>
<owl :Class rdf :ID="CONTACT-DETECTION">
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
</rdfs :subClassOf>
</owl :Class>
<CONTACT-DETECTION rdf :ID="CONTACT-DETECTION-5"/>
<owl :Class rdf :ID="CONTACT-SENSOR">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="WINDOW"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="WARDROB"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :ObjectProperty rdf :ID="detected-tag"/>
<owl :Class rdf :ID="DOOR">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="lookated-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
```

```
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="EVENT"/>
<owl :ObjectProperty rdf :ID="generated-by"/>
<owl :ObjectProperty rdf :ID="generated-by-1">
<rdfs :domain rdf :resource="CAMERA-TRIGGRED"/>
<rdfs :range rdf :resource="CAMERA-SENSOR"/>
<rdfs :subPropertyOf rdf :resource="generated-by"/>
</owl :ObjectProperty>
<owl :ObjectProperty rdf :ID="generated-by-2">
<rdfs :domain rdf :resource="CONTACT-DETECTION"/>
<rdfs :range rdf :resource="CONTACT-SENSOR"/>
<rdfs :subPropertyOf rdf :resource="generated-by"/>
</owl :ObjectProperty>
<owl :ObjectProperty rdf :ID="generated-by-3">
<rdfs :domain rdf :resource="LIGHT-ON"/>
<rdfs :range rdf :resource="LIGHT-SENSOR"/>
<rdfs :subPropertyOf rdf :resource="generated-by"/>
</owl :ObjectProperty>
<owl :ObjectProperty rdf :ID="generated-by-5">
<rdfs :domain rdf :resource="MOUVEMENT-DETECTION"/>
<rdfs :range rdf :resource="MOUVEMENT-SENSOR"/>
<rdfs :subPropertyOf rdf :resource="generated-by"/>
</owl :ObjectProperty>
<owl :DatatypeProperty rdf :ID="hasID">
<rdfs :domain rdf :resource="TAG"/>
<rdfs :range rdf :resource="xsd ;int"/>
</owl :DatatypeProperty>
<owl :DatatypeProperty rdf :ID="hasName">
<rdfs :domain rdf :resource="PERSON"/>
<rdfs :range rdf :resource="xsd ;string"/>
</owl :DatatypeProperty>
<owl :ObjectProperty rdf :ID="installed-in">
<rdfs :domain rdf :resource="CONTACT-SENSOR"/>
<rdfs :range rdf :resource="DOOR"/>
<rdfs :seeAlso rdf :datatype="xsd ;string"></rdfs :seeAlso>
```

```
</owl :ObjectProperty>
<owl :ObjectProperty rdf :ID="is-a"/>
<owl :DatatypeProperty rdf :ID="isActive">
<rdfs :domain rdf :resource="TAG"/>
<rdfs :range rdf :resource="xsd ;boolean"/>
</owl :DatatypeProperty>
<owl :Class rdf :ID="LIGHT-ON">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="LIGHT-SENSOR">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :ObjectProperty rdf :ID="lookated-in"/>
<owl :Class rdf :ID="MESSAGE">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="generated-by"/>
<owl :allValuesFrom rdf :resource="SERVICE"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
```

```
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="MOUVEMENT-DETECTION">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="MOUVEMENT-SENSOR">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed¡n"/ >
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :ObjectProperty rdf :ID="performs"/>
<owl :Class rdf :ID="PERSON">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="wears"/>
<owl :allValuesFrom rdf :resource="TAG"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="performs"/>
<owl :someValuesFrom rdf :resource="ACTIVITY"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
```

<owl :onProperty rdf :resource="lookated-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<swrl :Imp rdf :ID="Rule-1">
<swrl :body>
<swrl :AtomList>
<rdf :first>
<rdf :Description>
<rdf :type rdf :resource="swrl ;ClassAtom"/>
<swrl :argument1>
<rdf :Description rdf :about="x"/>
</swrl :argument1>
<swrl :classPredicate rdf :resource="PERSON"/>
</rdf :Description>
</rdf :first>
<rdf :rest>
<swrl :AtomList>
<rdf :first>
<rdf :Description>
<rdf :type rdf :resource="swrl ;DatavaluedPropertyAtom"/>
<swrl :argument2>
<rdf :Description rdf :about="nom"/>
</swrl :argument2>
<swrl :argument1>
<rdf :Description rdf :about="x"/>
</swrl :argument1>
<swrl :propertyPredicate rdf :resource="hasName"/>
</rdf :Description>
</rdf :first>
<rdf :rest rdf :resource="rdf ;nil"/>
</swrl :AtomList>
</rdf :rest>
</swrl :AtomList>
</swrl :body>

<swrl :head>
<swrl :AtomList>
<rdf :first>
<rdf :Description>
<rdf :type rdf :resource="swrl ;BuiltinAtom"/>
<swrl :arguments>
<rdf :List>
<rdf :first>
<rdf :Description rdf :about="nom"/>
</rdf :first>
<rdf :rest rdf :resource="rdf ;nil"/>
</rdf :List>
</swrl :arguments>
<swrl :builtin>
<rdf :Description rdf :about="sqwrl ;select"/>
</swrl :builtin>
</rdf :Description>
</rdf :first>
<rdf :rest rdf :resource="rdf ;nil"/>
</swrl :AtomList>
</swrl :head>
</swrl :Imp>
<PERSON rdf :ID="PERSON-4">
<hasName rdf :datatype="xsd ;string">djafar</hasName>
</PERSON>
<owl :Class rdf :ID="RFID-DETECTION">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="detected-tag"/>
<owl :someValuesFrom rdf :resource="TAG"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="generated-by"/>
<owl :allValuesFrom rdf :resource="RFID-READER"/>
</owl :Restriction>

```
</rdfs :subClassOf>
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="is-a"/>
<owl :allValuesFrom rdf :resource="EVENT"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="RFID-READER">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="ROOM"/>
<owl :Class rdf :ID="SERVICE">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="installed-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :Class rdf :ID="TAG"/>
<TAG rdf :ID="TAG-4">
<hasID rdf :datatype="xsd ;int">55</hasID>
<isActive rdf :datatype="xsd ;boolean">true</isActive>
</TAG>
<owl :ObjectProperty rdf :ID="take-place-in">
<rdfs :domain rdf :resource="ACTIVITY"/>
</owl :ObjectProperty>
<owl :Class rdf :ID="WARDROB">
```

<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="lookated-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
<owl :ObjectProperty rdf :ID="wears"/>
<owl :Class rdf :ID="WINDOW">
<rdfs :subClassOf>
<owl :Restriction>
<owl :onProperty rdf :resource="lookated-in"/>
<owl :someValuesFrom rdf :resource="ROOM"/>
</owl :Restriction>
</rdfs :subClassOf>
<rdfs :subClassOf rdf :resource="owl ;Thing"/>
</owl :Class>
</rdf :RDF>

Annexe B :

Le code entier pour le raisonnement avec Event Calculs

B.1 Equipements d'observation

sort : Person(samia).
sort : Alarm(alarme).
sort : Camera(camera1).
sort : RfidReader(rfidReader).
sort : LightSensor(lightSensor1).
sort : MovementSensor(Zmove).
sort :ContactSensor(contactSensor).
sort : Tag(tag1).
sort : Room(room1).
sort : Door(door1).
sort : Wardrobe(wardrobe1).
sort : Window(window1).
sort : Light(light1).
sort : Service(service1).

B.2 Définition des événement(event) et des prédicats(fluent)

event : TagPassed(Person,Tag).

event : Check(Tag).

event : RfidDetection(RfidReader,Tag).

event : CameraTriggered(Camera).

event : VocalMessage(Person).

event : AlarmTriggered(Alarm).

event : Open(Door).

event : Close(Door).

event : MovementDetection(Room).

event : PlayMessagePersonneInconnue(Person).

event : LocatePersson(Person,Room).

event : SwitchTheLight(Light).

event : ContactDetction(wardrobe).

event : OpenWardrobe(Wardrobe).

event : CameraTriggeredWardrobe(Camera).

event : AccessLocked(Wardrobe).

event : PlayMessagePersonneNonAutorisee(Service).

event : EntranceAuthorisedPerson(Person,Room).

event : CloseWindow(Window).

fluent : LightOn(LightSensor).

fluent : RfidReader(Room).

fluent : ContactSensor(Door).

fluent : ContactSensoWardrobe(Wardrobe).

fluent : CameraSensor(Camera).

fluent : MovementSensor(Room).

fluent : TagDetected(Tag).

B.2 Les axiomes et les régles appliqués sur notre scénario

HoldsAt(TagDetected(?Tag), ?t)$\Longrightarrow$Happens(Check(?Tag), ?t).

La detection d'un tag provoque sa vérification.

Happens(Check(?Tag), ?t)$\wedge$ $\neg$HoldsAt(RfidReader(?Room), ?t)$\wedge$

HoldsAt(CameraSensor(?Camera), ?t)$\Longrightarrow$ Happens(CameraTriggered(?Camera), ?t).

Si vérification du tag et non lecture du tag et activation du capteur de la cam alors déclencher la cam

Happens(Check(?Tag), ?t) $\wedge\neg$HoldsAt(RfidReader(?Room), ?t)$\wedge$

Happens(CameraTriggered(?Camera), ?t) $\Longrightarrow$Happens(PlayMessage

PersonneNonAutorisee(?Service), ?t).

Si vérification du tag et non lecture du tag et activation de la cam alors envoyer message personne non autorisée

Happens(Check(?Tag), ?t) ∧¬HoldsAt(RfidReader(?Room), ?t)∧
HoldsAt(ContactSensor(?Door), ?t)∧ Happens(PlayMessagePersonneNonAutorisee(
?Service), ?t)⟹ Happens(AlarmTriggered(?Alarm), ?t).

Si vérification du tag et non lecture du taget tentative d'ouverture alors déclencher l'alarme

Happens(Check(?Tag), ?t) ∧HoldsAt(RfidReader(?Room), ?t)∧
HoldsAt(ContactSensor(?Door), ?t)⟹ Happens(Open(?Door), ?t).

Si vérification du tag et lecture du taget tentative d'ouverture alors ouvrir la porte

Initiates(OpenDoor(?Door),CameraSensor(?camera), ?t).

Ouverture de la porte initie l'activation de la caméra

Happens(Open(?Door), ?t)⟹Happens(CameraTriggered(?Camera), ?t).

Si ouverture de la porte alors déclenchement de la caméra

Happens(Open(?Door), ?t)∧Happens(CameraTriggered(?Camera), ?t)
⟹Happens(Close(?Door), ?t).

fermeture de la porte après son ouverture et déclenchement de la caméra

Happens(Close(?Door), ?t)⟹Terminates(Close(?Door),CameraSensor(?Camera), ?t).

la fermeture de la porte désactive la caméra

Happens(Check(?Tag), ?t) ∧HoldsAt(RfidReader(?Room), ?t)∧Happens(Close(?Door), ?t)∧
HoldsAt(MovementSensor(?Room), ?t)⟹Happens(EntranceAuthorisedPerson(?Person, ?Room), ?t).

Fermeture de la porte et détection de mouvement par le capteur induit à une détectin d'une entrée de personne autorisée.

Happens(EntranceAuthorisedPerson(?Person, ?Room), ?t)∧HoldsAt(LightOn(?LightSensor), ?t)
⟹Happens(SwitchTheLight(?Light), ?t).

Si entrée d'une personne autorisée alors allumer les lumières.

Happens(EntranceAuthorisedPerson(?Person, ?Room), ?t)∧HoldsAt(ContactSensoWardrobe
(?Wardrobe), ?t)∧Happens(SwitchTheLight(?Light), ?t)⟹
Happens(CameraTriggeredWardrobe(?Camera), ?t).

Si entrée d'une personne autorisée et tentative d'ouverture de l'armoire alors déclencher la caméra.

Happens(EntranceAuthorisedPerson(?Person, ?Room), ?t)∧HoldsAt(ContactSensoWardrobe
(?Wardrobe), ?t)∧Happens(CameraTriggeredWardrobe(?Camera), ?t)
⟹Happens(OpenWardrobe(?Wardrobe), ?t).

Si entrée d'une personne autorisée et tentative d'ouverture de l'armoire alors ouverture de l'armoire.

HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)
$\Longrightarrow$ Happens(AlarmTriggered(?Alarm), ?t).

Si détection de mouvement par le capteur et non lecture du tag alors déclencher l'alarme.

HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(AlarmTriggered(?Alarm), ?t)$\Longrightarrow$Happens(CameraTriggered(?Camera), ?t).

Si détection de mouvement par le capteur et non lecture du tag Déclencher la caméra

HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(CameraTriggered(?Camera), ?t)$\Longrightarrow$ Happens(CloseWindow(window1), ?t).

Si détection de mouvement par le capteur et non lecture du tag alors fermeture des fenêtres.

HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(CloseWindow(window1), ?t)$\Longrightarrow$Happens(Close(?Door), ?t).
HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(Close(?Door), ?t)$\Longrightarrow$Happens(PlayMessagePersonneNonAutorisee(?Service), ?t).

Si detection de mouvement par le capteur et non lecture du tag envoyer le message personne non autorisée.

HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(PlayMessagePersonneNonAutorisee(?Service), ?t)$\Longrightarrow$Happens(LocatePerson(?Room), ?t).
HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(LocatePerson(?Room), ?t)∧HoldsAt(LightOn(?LightSensor), ?t)
$\Longrightarrow$Happens(SwitchTheLight(?Light), ?t).
HoldsAt(MovementSensor(?Room), ?t)∧¬HoldsAt(RfidReader(?Room), ?t)∧
Happens(PlayMessagePersonneNonAutorisee(?Service), ?t)∧HoldsAt(ContactSensoWardrobe
(?Wardrobe), ?t) $\Longrightarrow$Happens (AccessLocked(?Wardrobe), ?t).

Si detection de mouvement par le capteur et non lecture du tag et tentative d'ouverture des armoires alors bloquer l'accès.

BIBLIOGRAPHIE

[1] Mark Weiser. The computer for the 21st century. Scientific American, September 1991.

[2] Z.Drey, Vers une méthodologie dédiée à l'orchestration d'entités communicantes. Thèse de doctorat en informatique ; département de formation doctorale en informatique université du Bordeaux1, septembre 2010.

[3] M .Miraoui . Architecture logicielle pour l'informatique diffuse : modélisation du contexte et adaptation dynamique des services .Thése de doctorat en informatique ; école de technologie supérieur université du Québec, Montréal, juillet 2009.

[4] R.Ajhoun, A.Begdouri, M.Berraho, L'informatique mobile au service des applications médicales, Séminaires SIM'07, université de Fés Sidi Mohammed Ben Abdellah, 2007.

[5] http ://www.lifl.fr/icar/main.pdf

[6] http : //www.lamsade.dauphine.fr/scripts/FILES/publi1449.pdf

[7] http : //www.atoute.org/n/spip.php ?page=articleprintn $\texttt{id\_article} = 150$

[8] M. Doi, K. Ouchi and T. Suzuki. A wearable healthcare support system with timely instruction based on the user's context-advanced motion control. Lifeminder : in proceeding 8th IEEE International Workshop on AMC, Kawasaki, Japan, pages 445-450, 2004.

[9] A.Kouicem, Composition dynamique de services en environnements ubiquitaires. Mémoire de Magister en informatique ; École Doctorale Réseaux et Systèmes Distribués université Abderrahmane Mira de Bejaïa, 2008.

[10] T. Chaari : Adaptation d'applications pervasives dans des environnements multi-contextes. Thèse de doctorat, INSA de Lyon, septembre 2007.

[11] B. Schilit et M. Theimer : Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) :22-32,1994.

[12] P. J. Brown, J. D. Bovey et X. Chen : Context-aware Applications : from the Laboratory to the Marketplace. IEEE Personal Communications, 4(5) :58-64, October 1997.

[13] N. S. Ryan, J. Pascoe et D. R. Morse : Enhanced Reality Fieldwork : the Context-aware Archaeological Assistant.In V. Gaffney, M. van Leusen et S. Exxon, éditeurs : Computer Applications in Archaeology 1997, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

[14] J. Pascoe : Adding generic contextual capabilities to wearable computers. In Wearable Computers, 1998. Digest ofPapers. Second International Symposium on, pages 92-99, octobre 1998.

[15] D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith et P. Steggles : Towards a Better Understanding of Context and Context-Awareness. In HUC '99 : Proceedings of the 1st international symposium on Handheld andUbiquitous Computing, pages 304-307, London, UK, 1999.

[16] T.Winograd : Architectures for context. Human-Computer Interaction, 16 :401-419, 2001.

[17] Brezillon, P., M. R. Borges, J.A. Pino et J.-Ch. Pomerol. " Context-Awareness in Group Work : Three Case Studies ". In 2004 IFIP Int. Conf. on Decision Support Systems (DSS 2004) (Juillet, 2004). Prato, Italie.

[18] Schilit, B. N., N. Adams et R. Want. " Context-Aware Computing Applications ". In Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, (Decenber 1994). p. 85-90. Santa Cruz, CA, : IEEE Computer Society.

[19] Dey, A. K. 2001" Understanding and using context ". Personal and ubiquitous computing, vol. 5, p. 4-7.

[20] Chen, G., et D. Kotz.2000 A Survey of Context-Aware Mobile Computing Research. TR2000-381. Dartmouth : Dartmouth College Computer Science

[21] Petrelli, D., E. Not, C. Strapparava, O. Stock et M. Zancanaro. 2000. " Modeling Context is Like Taking Pictures ". In CHI'2000 Workshop on Context Awareness (April 1-6, 2000). The Hague, Netherlands.

[22] Gwizdka, J. 2000. " What's in the Context ? ". In Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000) (April 3, 2000). The Hague, The Netherlands.

[23] Hofer, T., W. Schwinger, M. Pichler, G. Leonhartsberger et J. Altmann. 2002. " Contextawareness on mobile devices - the hydrogen approach ". In the 36th Annual Hawaii International Conference on System Sciences. p. 292-302. Hawaii, USA.

[24] Razzaque, M. A., S. Dobson et P. Nixon. 2005. " Categorization and modeling of quality in contextinformation ". In the IJCAI 2005 Workshop on AI and Autonomic Communications (August 2005). Edinburgh, Scotland.

[25] G. K. Mostéfaoui, J. Pasquier-Rocha, and P. Brézillon. Context-Aware Computing : A Guide for the Pervasive Computing Community. In ICPS : IEEE International Conference on Pervasive Services, pages 39-48, 2004.

[26] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde. Advanced Interaction in Context. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 89-101, London, UK, 1999. Springer- Verlag.

[27] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni. A survey of context modelling and reasoning techniques, 2010. P 161-180.

[28] T.Chaari. Adaptation d'applications pervasives dans des environnements multi-contextes. Thèse de Doctorat en informatique, Institut national des sciences appliquées de Lyon, 2007

[29] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications : Models and approach. Journal of Pervasive and Mobile Computing, volume 2(1) :pages 37-64, Elsevier, 2006.

[30] Q. Z. Sheng and B. Benatallah. ContextUML : A UML-Based Modeling Language for Model- Driven Development of Context-Aware Web Services. In The 4th International Conference on Mobile Business (ICMB'05), IEEE Computer Society. Sydney, Australia, July 11-13 2005.

[31] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. Composite Capability/Preference Profile (CC/PP) : Structure and vocabularies 1.0. Technical report, W3C recommandation, 15 january 2004.

[32] F. Manola and E. Miller. RDF Primer. Technical report, W3C recommandation, 10 February 2004.

[33] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in Using CC/PP in Context-Aware Systems. In 4th international Conference on Mobile Data Management, pages 247-261, London, UK, 2003. Springer-Verlag.

[34] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In Pervasive 2002, pages 167-180, Zurich, Switzerland, 2002.

[35] Patricia Dockhorn Costa. Towards a services platform for context-aware applications. Master thesis. En schede, The Netherlands : University of Twente, august 2003.

[36] K. Henricksen, S. Livingstone, J. Indulska, Towards a Hybrid Approach to Context Modelling, Reasoning and Interoperation, in : J. Indulska, D. D. Roure (eds.), Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning And Management, in conjunction with UbiComp 2004, Nottingham, England : University of Southhampton, 2004.

[37] C. Bettini, D. Maggiorini, D. Riboni, Distributed context monitoring for the adaptation of continuous services, World Wide Web Journal 10 (4) (2007) 503-528.

[38] W3C. OWL Web Ontology Language Use Cases and Requirements. http ://www.w3.org/TR/webont-req/, W3C Recommendation 10 February 2004.

[39] A. Agostini, C. Bettini, D. Riboni, Hybrid reasoning in the care middleware for context-awareness, International Journal of Web Engineering and Technology,To appear. (Extended and revised version of papers appeared in proc. of MobiQuitous 2005 and CoMoRea 2007.).

[40] B Tixiter. «La problématique de la gestion des connaissances », Le cas d'une entreprise de développement informatique bancaire, Rapport de Recherche No 01.9 Septembre 2001.

[41] Kowalski et Sergot, " Alogic Based Calculus of events", New Generation Computing, vol 4, 1986,p.67-95.

[42] R. Miller and M. Shanahan, "Some Alternative Formulations of the Event Calculus," in Computational Logic : Logic Programming and Beyond, 2002, pp. 95-111.

[43] D. Nute, "Defeasible Logic",INAP'01 Proceedings of the Applications of prolog 14th international conference on Web knowledge management and decision support Pages 151-16.

[44] J. L. Pollock, "Defeasible Reasoning", Cognitive Science journal, 1987,volume = 11.

[45] G. Hendrix, "Modeling simultaneous actions and continuous processes," Artificial Intelligence, vol. 4, pp. 145-180, 1973.

[46] R. Miller and M. Shanahan, "Narratives in the Situation Calculus," Journal of Logic and Computation, vol. 4, no. 5, pp. 513–530, 1994.

[47] M. Shanahan, "A circumscriptive calculus of events," Artificial Intelligence, vol. 77, no. 2, pp. 249-284, 1995.

[48] R. Miller and M. Shanahan, "The event calculus in classical logic - alternative axiomatizations," Electronic Transactions in Artificial Intelligence, vol. 4, pp. 77–105, 1999.

[49] V. Lifschitz, "Formal theories of action (preliminary report)," in Proc. of IJCAI, vol. 87, pp. 966-972, 1987.

[50] M. Shanahan, "Solving the frame problem : a mathematical investigation of the common sense law of inertia". MIT press, 1997.

[51] J. McCarthy, "Mathematical Logic in Artificial Intelligence," Daedalus, vol. 117, no. 1, pp. 297- 311, Winter. 1988.

[52] R.A. Kowalski. Database updates in the event calculus. Technical Report DOC 86/12, London : Imperial College of Science, Technology, and Medicine, 1986.

[53] F. Sadri. Three recent approaches to temporal reasoning. In A.P. Galton, editor. Temporal Logics and their Applications, pages 121-168. Academic Press, London, 1987.

[54] K. Eshghi. Abductive planning with event calculus. In R.A. Kowalski and K.A. Bowen, editors. Logic Programming : Proceedings of the Fifth International Conference and Symposium, vol. 1, pages 562-579. MIT Press, Cambridge, MA, 1988.

[55] M. Shanahan. Prediction is deduction but explanation is abduction. In N.S. Sridharan, editor, Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pages 1055-1060, San Mateo, CA, 1989. Morgan Kaufmann.

[56] M. Shanahan. Representing continuous change in the event calculus. In L.C. Aiello, editor, Proceedings of the Ninth European Conference on Artificial Intelligence,pages 598-603, London, 1990. Pitman.

[57] E.T. Mueller. Event calculus reasoning through satisfiability. Journal of Logic and Computation, 14(5) :703-730, 2004.

[58] E.T. Mueller. Commonsense Reasoning. Morgan Kaufmann, San Francisco,2006.

[59] J. McCarthy, P. Hayes, and S. U. C. D. O. C. SCIENCE, Some philosophical problems from the standpoint of artificial intelligence. Stanford University, 1968.

[60] M. Thielscher, "Introduction to the Fluent Calculus," Electronic Transactions in Artificial Intelligence, pp. 179–192, 1998.

[61] M.Gagnon, Logique descriptive et OWL , 2008.

[62] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau et John Cowan. Extensible Markup Language (XML) 1.1. http ://www.w3.org/TR/2004/RECxml11-20040204/ (en ligne au 16 juin 2005). Traduction française : Le langage de balisage extensible (XML) 1.1, http ://www.yoyodesign.org/doc/w3c/xml11/.

[63] BRICKEY Dan GUHA Ramanathan. Ressource Description Framework Schema specification http ://www.w3.org/TR/PRrdfschema.

[64] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. The Description Logic Handbook. Theory, Implementation and Applications. Cambridge University Press (2003), 574 pages.

[65] G .SANCHO, Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires.Contribution à l'interopérabilité par la sémantique, Université Toulouse 1 Capitole (UT1 Capitole), 2010.

[66] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider et Lynn Andrea Stein. OWL Web Ontology Language - Reference. http ://www.w3.org/TR/2004/REC-owl-ref-20040210/ (en ligne au 16 juin 2005). Traduction française : La référence du langage d'ontologie Web OWL, http ://www.yoyodesign.org/doc/w3c/owl-ref-20040210/.

[67] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof et M. Dean : SWRL : A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004, 2004. Url : http ://www.w3. org/Submission/SWRL/, accédé le 23/07/2010.

[68] D.Elenius, D.Martin, R.Ford, and G.Denker ,Reasoning about Resources and Hierarchical Tasks Using OWL and SWRL, SRI International, Menlo Park, California, USA.

[69] A. Horn : On sentences which are true of direct unions of algebras. Journal of Symbolic Logic, 16(1) :14-21, 1951.

[70] B. Parsia, E. Sirin, B. C. Grau, E. Ruckhaus et D. Hewlett : Cautiously approaching SWRL. Url : http ://www. mindswap.org/papers/CautiousSWRL.pdf, accédé le 23/07/2010, 2005.

[71] H. Knublauch, R. W. Fergerson, N. F. Noy et M. A. Musen : The Protégé OWL plugin : An open development environment for semantic web applications. pages 229- 243. Springer, 2004.