République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderahmane MIRA - 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

Une modèlisation hybride des connaissances contextuelles dans le domaine des systèmes ubiquitaires

---

Présenté par :

M^{elle} **ACHOUR Kahina**

M^{elle} **KERKAR Douriya**

Soutenu devant le jury composé de :

M^{r} **F. MIR**	**Président**	**U. A/Mira Béjaia.**
M^{me} **H.KHALED**	**Examinatrice**	**U. A/Mira Béjaia.**
M^{elle} **N. KHOULALENE**	**Examinatrice**	**U. A/Mira Béjaia.**
M^{me} **S. BOUTRID**	**Encadreur**	**U. A/Mira Béjaia.**

Juin 2013

Remerciement

Nous tenons à exprimer notre gratitude à M^{me} S. BOUTRID, notre promotrice pour nous avoir fait profiter de son expérience de recherche et son savoir faire, sa patience et pour nous avoir orienté. Ses conseils ont été très utiles et nous ont permis de mener à bien ce travail.

Nous voudrons exprimer toute notre gratitude à M^{r} F. MIR, pour nous avoir fait l'honneur de présider le Jury de soutenance.

Nos remerciements vont également à M^{me} H. KHALED et M^{elle} N. KHOULALENE qui nous ont fait l'honneur de participer au jury de soutenance.

Nous voudrons témoigner tous nos remerciements à nos familles pour leur encouragement et leur soutien.

Enfin, nos sincères remerciements sont adressés à nos amis et à toutes les personnes qui nous ont aidé et soutenu tout en long de la réalisation de ce mémoire.

Dédicaces

Je dédie ce mémoire...

À mes très chers parents

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Ce travail est le fruit des sacrifices que vous avez consentis pour mon éducation et mes études.

À mon très cher fiancé Walid

Ton soutien, tes conseils et tes encouragements m'ont appris à ne pas me sous-estimer et m'ont permis de réussir mes études.

À mon très cher grand père

L'homme au grand coeur généreux. C'est avec tous mon respect et mon admiration que je te dédie ce travail

À mes très chers frères et soeurs

Vous avez toujours été présents dans les moments difficiles, votre soutien m'a été d'un grand secours tout au long de mes études.

À mon très chèr binôme et amie Kahina

Malgré les obstacles et les difficultés qui se sont interposent,nous avons pu les surmontés. En témoignage de l'amitié qui nous uni et des souvenirs de tous les moments que nous avons passé ensemble, je te dédie ce travail et je te souhaite une vie pleine de santé et de bonheur

À tous les membres de ma famille

Je vous dédie ce travail avec tous mes voeux de bonheur, de santé et de réussite.

Douriya

Dédicaces

Je dédie ce mémoire...

À *mes très chers parents Abdallah et Malika.*

Vous vous avez dépensés pour moi sans compter.

En reconnaissance de tous les sacrifices consentis pour me

permettre d'atteindre cette étape de ma vie.

À *mes chers frères Salim et Naserdine.*

À *mes cheres soeurs Kamela et Naima.*

Vous avez de près ou de loin contribué à ma formation.

À *ma copine Douriya et à sa famille.*

Je te souhaite une belle vie.

À *Khoudir et à sa famille.*

Affectueuse reconnaissance et Sincère gratitude

Vous avez contribué en fonction de vos moyens à affermir ma formation.

M*erci infiniment pour tous.*

Kahina

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.

La modélisation du contexte est la première démarche dans le processus de création d'applications sensibles au contexte, pour permettre de simplifier leur développement.

En ce document nous discutons les besoins aux lesquels les approches de modélisation et de raisonnement de contexte devraient répondre. Donc nous proposons une approche hybride OWL/EC afin de résoudre le problème de négation dans les ontologies.

Mots clés : informatique ubiquitaire, sensibilité au contexte, modélisation du contexte, modélisation hybride, Ontologie et Event Calculus.

Abstract

The technological evolution of the mobile devices gave rise to new applicatifs needs to ensure the execution of the applications in dynamic environments. These applications called context-awarness applications must detect the variations of the environment and adapt consequently.

The modeling of the context is the first step in the creative process of context-awarness applications, to make it possible to simplify their development.

In this document we discuss the needs which the approaches of modeling and reasoning of context should answer. Thus we propose a hybrid approach OWL/EC in order to solve the problem of negation in ontologies.

Keywords : ubiquitous computing, context-awarness, context modeling, hybrid modeling, ontology and Event Calculus.

Table des Matières

Liste des tableaux

Table des figures

Liste des acronymes

ADSL : Asymmetric Digital Subscriber Line.

AL : Attributive Language.

API : Application Programming Interface.

BEC : Basic Event Calculus.

CC/PP : Composit Capability/Prefrence Profile.

CML : Context Modeling Langage.

DEC : Discrete Event Calculus.

DSL : Digital Suscriber Line.

DTD : Document Type Definition.

EC : Event Calculus.

GPRS : General Packet Radio System.

GPS : Global Positionning System.

GSM : Global System for Mobile communication.

IA : Intelligence Artificielle.

OEC : Original Event Calculus.

ORM : Object Role Modeling.

OWL : Web Ontology Language.

OWL-LD : Web Ontology Language-Description Logic.

PC : Personnel Computer.

PDA : Personnel Digital Assistants.

RDF : Resource Description Framework.

RFID : Radio Frequency Identification.

RuleML : Rule Markup Langage.

SWRL : Semantic Web Rule Language.

SEC : Simplified Event Calculus.

UML : Unified Modeling Language.

UMTS : Universal Mobile Telecommunication System.

URI : Uniform Resource Identifier.

W3C : World Wide Web Consortium.

WIFI : Wireless Fidelity.

WLAN : Wireless Local Area Network.

WMAN : Wireless Metropolitan Area Network.

WPAN : Wireless Personal Area Network.

WWAN : Wireless Wide Area Network.

XML : eXtensible Markup Language.

Introduction générale

"The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it."
Marc Weiser, (1952 - 1999),
The Computer for the 21^{st} Century (1991)

Depuis les années 1940, l'informatique a accéléré de plus en plus la diffusion de l'information et des connaissances. L'invention du transistor [1], la démocratisation de l'ordinateur personnel, la révolution du Web ou l'accès aux réseaux sociaux depuis des terminaux mobiles, sont des étapes qui ont de plus en plus rapproché les technologies de l'information et de la communication de notre vie de tous les jours.

En 1991, Mark Weiser [6] présentait sa vision futuriste de l'informatique du 21^{eme} siècle en établissant les fondements de ce que nous appelons aujourd'hui l'informatique ubiquitaire (*Ubiquitous Computing*) : les ordinateurs doivent s'intégrer dans notre environnement et devenir omniprésents.

Le dictionnaire définit le terme ubiquitaire comme quelque chose "qui est ou semble être partout à la fois." Dans le cas de l'informatique, cet adjectif traduit que les dispositifs informatiques, capables de nous assister dans nos tâches quotidiennes, vont continuer à nous entourer progressivement. Nous allons passer d'une interaction de type "bureau", un-vers-un, vers une autre de type ubiquitaire, plusieurs-vers-plusieurs. Au lieu de nous asseoir face à un ordinateur pour lui demander des actions, une multitude d'ordinateurs, de capteurs et d'actionneurs vont interagir autour et avec nous dans nos bâtiments, dans nos moyens de transport, dans nos

1. Dispositif à semi-conducteurs, utilisé en électronique comme amplificateur, modulateur, oscillateur, interrupteur. Récepteur radiophonique portatif, muni de ce dispositif

rues... etc. Ces dispositifs vont devenir de plus en plus petits et nombreux, ce qui va changer radicalement notre façon de les utiliser. Pour désigner ces espaces riches en dispositifs et en interactions, on parle d'environnements ubiquitaires.

L'essence même de cette vision consiste à mettre les nouvelles technologies de l'information et de la communication au service des utilisateurs non l'inverse. L'objectif est de permettre à ces derniers d'accéder aux différentes fonctionnalités offertes par les divers dispositifs informatiques hétérogènes présents dans leur environnement immédiat, à partir de n'importe quel terminal, par exemple leur téléphone portable, ou leur PDA (Personal Digital Assistant).

Afin que les applications répondent au mieux aux attentes des utilisateurs, il est nécessaire de prendre en considération les informations contextuelles. La prise en compte du contexte d'utilisation dans les applications, est un domaine de recherche d'actualité connu sous le nom de "sensibilité au contexte" (ou *context-awareness* en anglais) : une application sensible au contexte doit percevoir la situation de l'utilisateur dans son environnement et adapter par conséquent son comportement à la situation en question, l'idée principale de l'informatique sensible au contexte est de réduire au maximum les interactions entre l'homme et la machine.

L'étude de la sensibilité au contexte nous amène tout d'abord à étudier les travaux de recherche effectués pour définir et modéliser le contexte. La modélisation permet d'offrir les fondations pour une représentation expressive du contexte et de simplifier son utilisation.

Pour stocker une information, nous avons besoin de définir un modèle pour la décrire. Ainsi, un modèle de contexte est requis pour pouvoir l'utiliser dans l'application. Strang et Linnhoff-Popien [7] ont résumé les approches de modélisation de contexte les plus intéressantes de la littérature. Ils ont proposé une classification des approches de modélisation basée sur la structure de données utilisée pour la description et l'échange du contexte. Nous présentons ces approches en quatre catégories par degré de complexité de leur structure de données : approches clés/valeurs, orientées modèle, logique et les modèles basés sur les ontologies.

L'objectif de ce travail est de proposer une approche de modélisation hybride entre le modèle ontologique OWl et le modèle logique temporel Event Calculus, pour modéliser et raisonner sur les informations du contexte dans les systèmes ubiquitaires.

Ce mémoire comporte cinq chapitres :

Les deux premiers chapitres présentent un état de l'art. Dans le premier chapitre, nous définirons des généralités sur les systèmes ubiquitaires à savoir : définition de l'informatique ubiquitaire et de son environnement, les caractéristiques de l'environnement ubiquitaire, et des

champs d'applications.

Dans le deuxième chapitre, nous détaillerons la notion de contexte, la sensibilité au contexte et nous étudions les approches de modélisation du contexte en informatique ubiquitaire.

Le troisième chapitre détaillera les deux modèles Event Calculus et OWL, ainsi que leurs avantages et limites.

Le quatrième chapitre sera consacré à l'exposition de la problématique, notre proposition et architecture.

Le dernier chapitre sera consacré à la validation de la proposition.

Enfin, ce mémoire se clôtura par une conclusion générale résumant les différents points qui ont été traités, ainsi que les perspectives à accomplir prochainement.

1

Généralités sur les systèmes ubiquitaires

1.1 Introduction

L'évolution technologique réalisée dans le domaine de l'informatique ne cesse de croitre, cette dernière est appelée à être de plus en plus présente au quotidien. L'intégration des objets mobiles tels que les puces dans les voitures, l'explosion de la téléphonie mobile, l'arrivée sur le marché des ordinateurs portables et maintenant des ultra-portables, donnent une idée de ce qui peut nous attendre en informatique dans les années à venir.

Grâce à ces terminaux mobiles et les services offerts, les objets du quotidien sont partout accessibles à tous moment.

L'informatique ubiquitaire est un paradigme dans lequel des systèmes numériques, intégrés dans les objets de la vie quotidienne, sont mis en réseau pour coopérer et fournir un environnement intelligent qui facilite l'utilisation et automatise l'exécution de services pour les utilisateurs.

Dans ce premier chapitre, nous abordons les généralités sur les systèmes ubiquitaire à savoir : définition de l'informatique ubiquitaire et de son environnement, les caractéristiques de l'environnement ubiquitaire, et des champs d'applications.

1.2 Définition de l'informatique ubiquitaire

Le concept de l'informatique ubiquitaire ou informatique ambiante converge vers la vision de l'informatique du futur que donnait *Mark Weiser* en 1991 dans son article[6] intitulé "*The computer for the* 21^{st} *Century*".

L'informatique ubiquitaire appelée également informatique pervasive, a pour but de rendre accessible toutes sortes de services, n'importe où, ce qui offre aux utilisateurs la possibilité de surmonter les contraintes actuelles d'utilisation d'un ordinateur (être assis devant un clavier,

un écran... etc) lui rend sa liberté d'actions, notamment sa liberté de mouvement. L'ubiquité permet donc souvent la mobilité.

L'informatique ubiquitaire a donné naissance à plusieurs termes, parmi eux nous trouvons :

- Ubiquitaire : accessible de n'importe où ;
- Mobile : qui intègre les terminaux mobiles ;
- Pervasif : capacité du dispositif à détecter des éléments du contexte ;
- Sensibilité au contexte (*context-aware*) : qui prend en compte le contexte d'exécution ;
- Ambiante : qui est intégré dans les objets quotidiens.

1.3 Définition d'un environnement ubiquitaire

Un environnement ubiquitaire[8] est un espace physique (une maison, un hôpital, une école, une autoroute, ou une ville) équipé d'une multitude d'entités matérielles ou logicielles communiquantes grâce à un réseau. Ces entités sont des capteurs (de température, de luminosité), des actionneurs (lampes, caméra), des technologies mobiles, 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.

FIGURE 1.1 – Exemple d'un environnement ubiquitaire

1.4 Caractéristiques de l'environnement ubiquitaire

L'environnement ubiquitaire relève plusieurs défis, certains d'entre eux sont présentés par les points suivants[9] :

1. **Limitation de ressources**

   Le fonctionnement des systèmes pervasifs réside en partie sur la collaboration des petits dispositifs mobiles. La taille réduite de ces objets mobiles implique des capacités de calcul, des capacités de mémoire et autonomie énergétique limitées. Cette dernière est une contrainte critique à prendre en compte dans le développement des applications. En effet, lorsqu'un dispositif n'a plus d'énergie, tous les services qu'il hébergeait deviennent inutilisables.

2. **Variation du contexte**

   Les systèmes pervasifs doivent être sensibles à leur environnement afin de prendre des décisions appropriées et fournir des services adéquats aux utilisateurs. Cette sensibilité consiste en la prise en compte de l'évolution des paramètres physiques de l'environnement. Ces paramètres constituent le contexte de l'environnement. La mobilité dans un environnement ubiquitaire provoque des changements dans le contexte des applications. En effet, les applications sensibles au contexte en informatique ubiquitaire sont sensibles à la localisation. De façon générale, le contexte décrit la situation de l'utilisateur en termes de localisation, du temps, d'environnement et du terminal utilisé, de profil utilisateur... etc par exemple, une situation contextuelle peut être définie par les paramètres suivants : profil utilisateur = "enseignent", terminal utilisé = "PDA", localisation = "salle de conférence". Le changement d'une valeur sur l'un de ces paramètres définit une nouvelle situation contextuelle à laquelle l'application doit s'adapter. On parle alors de dynamicité du contexte.

3. **Mobilité**

   La mobilité est un atout pour les systèmes pervasifs, mais cela n'empêche pas d'être un problème. La mobilité des dispositifs entraîne des variations de connexion qui peuvent avoir de lourdes conséquences sur le fonctionnement des services : faible débit et donc la lenteur des transmissions et parfois la déconnexion, tous comme l'épuisement d'une batterie, les déconnexions provoquant l'indisponibilité temporaire des services ce qui entraîne la dégradation de qualités de services.

4. **Hétérogénéité**

   Les équipements utilisés dans un environnement ubiquitaire sont très variés tel que les or-

dinateurs portables, les capteurs, PDA, appareils électroménagers, appareils médicaux... etc. Ces dispositifs sont différents à tous niveaux : matériel et logiciel. De plus, ils s'appliquent sur diverses technologies de communications filaires ou non filaires (ADSL, Wifi, Bluetoot)...etc

5. **Limitation réseaux**

   Les connexions sans fil utilisées dans les dispositifs mobiles ont généralement une bande passante limitée et instable. La vitesse de transmission des technologies standard sans fils (exemple : WI-FI, IEEE 802.11a, le Bluetooth...etc) lente comparée au réseau filaire a été toujours un obstacle pour exécuter des applications complexes et déplacer des données volumineuses sur les mobiles.

1.5 Equipements d'un environnement ubiquitaire

1.5.1 Equipements matériels

Le développement rapide de l'informatique ubiquitaire a été favorisé par les dernières avancées technologiques, nous citons quelques équipements :

1.5.1.1 Les dispositifs mobiles

Dans un environnement ubiquitaire, différents terminaux mobiles[10] sont utilisés tel que :

a) Personnel 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, calendrier, agenda, téléphone, fax... etc.

b) Smartphone : Un Smartphone ou téléphone intelligent, est un téléphone mobile disposant aussi des fonctions d'un assistant numérique personnel. La saisie des données se fait par le biais d'un écran tactile ou d'un clavier. Il fournit des fonctionnalités basiques comme : l'agenda, le calendrier, la navigation sur le Web, la consultation de courrier électronique, de messagerie instantanée, le GPS... etc.

c) Tablet PC : C'est 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.

d) Les puces Radio Frequency Identification (RFID) : L'identification par radiofréquence (RFID) est une méthode pour stocker et récupérer des données à distance en utilisant des

marqueurs appelés Tag RFID. Les Tags RFID sont de petits objets, tels que des étiquettes autoadhésives, qui peuvent être collés ou incorporés dans des produits. Les Tags RFID comprennent une antenne qui leur permet de recevoir et de répondre aux requêtes radio émises depuis l'émetteur au récepteur.

e) **Capteurs :** Capteurs de température, humidité, luminosité, détecteur de présence, caméra...etc.

1.5.1.2 Les réseaux filaires et non filaires

Ces dernières années, de nombreux standards ont vu le jour pour couvrir différents types de réseaux sans fil à savoir les réseaux : WPAN, WLAN, WMAN et WWAN.

a) **Les réseaux personnels sans fil (WPAN) :** Le réseau WPAN pour Wireless Personal Area Network, concerne les réseaux sans fil d'une faible portée. Ce type de réseau sert principalement à relier des périphériques (imprimante, appareils domestiques, téléphone portable, ou un assistant personnel...etc.) à un ordinateur sans liaison filaire. Plusieurs technologies sont utilisées pour les WPAN dont la principale est la technologie Bluetooth (IEEE 802.15.1) proposant un débit théorique de 1Mb/s pour une portée maximale de trentaines de mètres et possède l'avantage d'être peu gourmande en énergie.

b) **Les réseaux locaux sans fil (WLAN) :** Le réseau WLAN pour Wireless Local Area Network, est un réseau sans fil permettant de couvrir l'équivalent d'un réseau local d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier les terminaux présents dans la zone de couverture. Plusieurs technologies concurrentes existent dont la principale est le Wifi (ou IEEE 802.11), offrant des débits allant jusqu'à 54Mbps.

c) **Les réseaux Métropolitains sans fil (WMAN) :** Le WMAN pour Wireless Metropolitan Area Network se base sur la norme IEEE 802.16 offrant un débit de 1 à 10Mb/s pour une portée de 4 à 10Km.

d) **Les réseaux Etendus sans fil (WWAN) :** Le WWAN pour Wireless Wide Area Network, est connu sous le nom de réseau cellulaire sans fil. Il s'agit des réseaux de la téléphonie mobiles qui inclue les technologies GSM (Global System for Mobile communication), GPRS (General Packet Radio System), UMTS (Universal Mobile Telecommunication System).

Pour les réseaux filaires, ils offrent un débit plus élevé par rapport aux réseaux sans fil comme les réseaux Ethernet, DSL (Digital Suscriber Line).

1.5.2 Outils logiciels

1.5.2.1 Définition d'un middleware (intergiciel)

Un intergiciel désigne la couche logicielle intermédiaire qui fournit un haut niveau d'abstraction de programmation permettant de masquer l'hétérogénéité des réseaux de communications, des ressources matérielles, des systèmes d'exploitation et des langages de programmation, il se situe au dessus du système d'exploitation et au dessous des applications de son hôte. Un intergiciel a pour objectif de faciliter le développement, évolution, réutilisation des applications et leurs portabilités entre plates-formes.

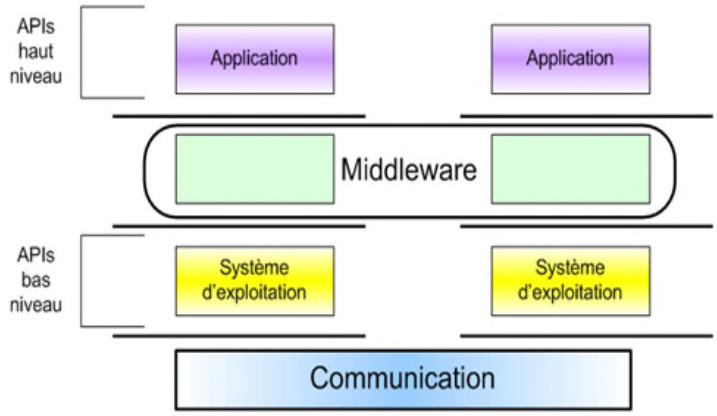


FIGURE 1.2 – Architecture d'un intergiciel (middleware)[1]

1.6 Scénarios utilisant l'informatique ubiquitaire

Pour mieux éclaircir les concepts fondamentaux des systémes ubiquitaires, nous présenterons deux exemples de scénario concret dans lesquels un utilisateur interagit avec différents services de son environnement.

Scénario1

Dans ce scénario[1], l'utilisateur cherche les clés de la maison. Il dit : "où sont mes clés ?"

Hypothèses

- Le système de reconnaissance vocale reconnaît ce qui a été dit et est capable de l'interpréter
- Le système est capable de connaître la position des clés
- Nous disposons d'une techno audio 3D, d'agents animés, d'un robot mobile

Réponse du système

- Émission d'un son 3D donnant l'impression de provenir de l'endroit où se trouvent les clés
- Affichage d'un agent animé sur l'écran du vidéo projecteur et qui fait un geste dans la direction de l'endroit où se trouvent les clés
- Déplacement d'un robot vers l'endroit où se trouvent les clés

Autres variations possibles

- Si l'utilisateur est aveugle, pas d'agent animé
- Si utilisateur sourd, pas de son
- Si vidéo projecteur occupé, utilisation de l'écran plasma pour l'affichage de l'agent animé

Scénario2

Les grandes conférences de recherches sont très dynamiques, il y a de nombreuses sessions de conférences en parallèle, sur des sujets très variés impliquant forcément des contextes changeants. Le *Conférence assistant* est un programme chargé dans l'ordinateur du participant et qui va lui permettre de décider, quelles conférences sont intéressantes pour lui, d'être au courant des activités de ses collègues, d'effectuer des interactions avec son environnement lors du déroulement de la conférence.

Les éléments de contexte utilisés sont très nombreux : temps, identité, localisation et activité.

Lors de son arrivée à la conférence, le participant se voit remettre l'assistant sous la forme d'un logiciel qu'il va charger dans son ordinateur. Les informations qu'il a donné lors de son inscription (centre d'intérêt, collègues présents, coordonnées) vont être intégrées par le logiciel assistant et serviront de contexte de base.

Lorsqu'il n'assiste pas à une conférence, l'application se présente à l'utilisateur sous la forme d'un planning de la conférence avec une présélection grisée des conférences qui peuvent l'intéresser.

Il peut aussi savoir à quelles conférences assistent ses collègues. Si le participant rentre dans une salle de conférence, l'assistant détecte un changement de contexte et réagit en affichant le nom du conférencier, le titre, le thème et d'autres informations sur la conférence. Lors du déroulement de la conférence, l'utilisateur reçoit via l'assistant les diapositives qui sont en train d'être passées et peut prendre des notes sur celles-ci. Il lui est possible à la fin de la présentation, lors de questions, de désigner et faire apparaître la diapositive sur laquelle porte sa question sur l'écran du conférencier. Le participant a la possibilité de noter l'intérêt qu'il accorde à une

conférence dans son planning, cette information est répercutée chez ses collègues qui peuvent savoir quelles sont les sessions intéressantes.

Il est enfin possible à l'utilisateur, une fois rentré chez lui de retrouver les présentations de la conférence en fonction des sessions auxquelles il a participé, annoté avec ses informations de contexte (Quand est-il arrivé dans la session ? Quelle question a t-il posée ? Quand est il parti ?).

Analyse

D'après les deux scénarios, on peut dégager, les caractéristiques fondamentales d'un système informatique ubiquitaire.il offre :

- Un service transparent sur un réseau ;
- Des équipements connectés par différents moyens (Wifi, Bluetooth...etc.) ;
- Un système qui surveille le contexte et les paramètres du service qu'il offre aux utilisateurs ;
- L'informatique ubiquitaire suppose donc de déterminer l'état courant du contexte grâce aux caractéristiques pervasives du dispositif, pour fournir les informations ou les interactions adaptées à la situation de l'utilisateur.

1.7 Des champs d'application de l'informatique ubiquitaire

L'informatique ubiquitaire s'est développée dans de nombreux champs d'applications. Ces applications peuvent être classées selon les services qu'elles offrent aux utilisateurs. Elles doivent pouvoir être facilement adaptées pour satisfaire leurs besoins et préférences en confort, sécurité, ou encore assistance quotidienne.

1. **Domotique**

   La domotique désigne[8] l'ensemble des technologies et techniques mises en oeuvre pour apporter des fonctions de gestion de la maison, tel 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).

2. Gestion de l'information

L'information numérique est désormais disponible partout : dans les espaces publics, les lieux professionnels, les entités mobiles. Des exemples d'information numérique sont la publicité, la météorologie, les actualités politiques. Ces informations sont constamment mises à jour et doivent être personnalisées selon les personnes à qui elles sont destinées. Un objectif des applications de gestion d'information est de recueillir et traiter des sources d'informations aussi variées que versatiles, telles que des agendas personnels ou d'entreprise. Des applications typiques de ce champ sont les gestionnaires de conférence ou de bulletins d'information dans les entreprises et l'affichage d'emplois du temps des étudiants dans les écoles.

3. Assistance à la personne

L'assistance à la personne est l'ensemble des aides apportées à une personne pour lui permettre de vivre de manière digne et autonome à domicile. Les applications d'aide à une personne ont pour objectif de lui fournir une assistance technologique compensant ses déficiences. En particulier, par un suivi adapté des actions d'une personne déficiente, ces applications pourront l'aider à coordonner ses activités quotidiennes, par exemple pour compenser ses problèmes de mémorisation. Pour cela ces applications doivent mettre à sa disposition des technologies d'assistance agissante de manière approprie pour la guider dans l'accomplissement de ses activités. L'avantage de telles applications est de minimiser l'intervention de tiers pour assister les personnes déficientes et d'augmenter leur indépendance.

1.8 Conclusion

Nous retenons dans ce chapitre l'aspect de l'informatique, l'environnement ubiquitaire et l'impact que pourrait avoir la mobilité des entités communicantes sur le contexte.

L'informatique ubiquitaire fait en sorte de déterminer l'état courant du contexte grâce aux caractéristiques pervasives du dispositif pour fournir des informations/interactions adaptées à la situation de l'utilisateur. Pour cela, il est nécessaire de décrire les caractéristiques des informations contextuelles et développer des applications sensibles au contexte.

2 Définitions et gestion du contexte

2.1 Introduction

L'étude de la sensibilité au contexte nous conduit à étudier les travaux de recherche effectués pour définir et modéliser le contexte.

Dans ce chapitre, nous étudions les approches de modélisation du contexte qui tentent d'intégrer différents modèles et différents types de raisonnement afin d'obtenir une représentation expressive du contexte et simplifier son utilisation.

Nous entamons ce chapitre par présenter une synthése des définitions données pour le contexte, la sensibilité au contexte en informatique ubiquitaire et nous décrivons les caractéristiques des informations contextuelles. Nous présentons ensuite l'architecture générale d'un système sensible au contexte tirée d'une synthèse des travaux existants. De plus, nous dressons un état de l'art des travaux de modélisation du contexte puis nous analysons ces différentes approches.

2.2 Définitions

2.2.1 Contexte

Le dictionnaire encyclopédie Larousse considère le contexte comme :*"l'ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours"*; ou encore :*"un ensemble de circonstances dans lesquelles se produit un événement, se situe une action"*.

2.2.2 Le contexte en informatique ubiquitaire

Le contexte étant une notion complexe et abstraite, que sa définition diffère d'une équipe de recherche à une autre. Les définitions générales sont les plus nombreuses ; nous listons ici les principales :

Parmi les premiers à essayer de définir le contexte, se trouvent *Schilit et Theimer*[11], 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.

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.

2.2.3 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'autres termes, du point de vue de l'application tous les contextes observables ne sont pas pertinents pour l'application.

2.2.4 Sensibilité au contexte

Schilit and al, 1994 ; [17] disent d'un système qu'il est sensible au contexte s'il peut tirer, interpréter et utiliser des informations issues du contexte et adapter sa réponse en fonction du

contexte d'utilisation.

Salbert and al ;[18] définissent la sensibilité au contexte comme étant la capacité d'un système à agir en temps réel avec des données provenant du contexte.

Brown, 1997 ;[12] dit d'une application sensible au contexte qu'elle doit automatiquement extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par les capteurs.

Dey et al, 2001 ; [19] proposent qu'un système soit sensible au contexte s'il utilise le contexte pour offrir des informations ou des services pertinents pour l'utilisateur.

2.2.5 Entités observables, observables

Entité observable : une entité est un élément représentant un phénomène logique ou physique (personne, concept, etc.) qui peut être une unité indépendante ou membre d'une catégorie et auquel des observables peuvent être attachés. Une personne, une machine, une pièce, un ensemble de personnes sont des exemples d'entités présentes dans l'environnement et qui peuvent être observées.

Observable : un observable est une abstraction qui définit un élément à surveiller (ou observer).

Observation : une observation représente l'état de l'observable à un instant donné.

Exemples

- Une personne est une entité observable.

  Les observables associés : sa localisation, sa préférence langue.
- Une salle est une entité observable.

  Les observables associés : le nombre de personnes présentes, la luminosité.

2.3 Les premières applications sensibles au contexte

- **Shopping Assistant guide :**[20]utilise la localisation de l'utilisateur pour le guider dans un magasin.
- **CyberGuide :**[21]offre des informations touristiques sur une carte interactive (sites a visiter selon (1) localisation utilisateur et (2) historique des visites).
- **Conference Assistant :**[22]assiste les participants a une conference ; suggestion de présentations a laquelle assister ; affichage automatique des présentations en cours.

- **Adaptive GSM phone and PDA :**[23]assiste les utilisateurs de terminaux mobiles ; change la taille d'écriture en fonction de l'activité de l'utilisateur ; sélection automatique des profils de téléphones portables en fonction du contexte (sonner, vibrer ou rester silencieux).
- **Call Forwarding :**[24] grâce au système de localisation d'Active Badge, le réceptionniste utilise la localisation de l'utilisateur pour faire suivre les appels téléphoniques vers le téléphone le plus proche de l'utilisateur.

2.4 Types d'informations contextuelles

Vu la diversité des sources de contexte, les informations contextuelles existantes sont nombreuses. Chen[25] a identifié quatre types :

- **Le contexte physique :** Toutes les données pouvant être mesurées dans le monde physique, comme la luminosité, le bruit, les embouteillages, la température, la distance, la vitesse...etc.
- **Le contexte utilisateur :** Tout ce qui concerne l'utilisateur et ce qui l'entoure : son profil, sa localisation, son humeur, ses intentions... etc.
- **Le contexte d'exécution :** Toutes les données virtuelles ou logicielles : connectivité réseau, bande passante, ressources informatiques disponibles ou à proximité, activité des périphériques...etc.
- **Le contexte temporel :** Tout ce qui a une relation avec le temps : dates, heure, durée, agenda, saison, événements périodiques...etc.

2.5 Caractéristiques des informations du contexte

Les informations de contexte sont des informations collectées à partir de plusieurs sources hétérogènes. De ce fait, elles ont des caractéristiques variables[3].

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 des sujets

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.

Afin d'utiliser les informations de contexte dans des applications informatiques, il est nécessaire de connaître pour chaque observable les caractéristiques citées ci-dessus. De plus, il est très important d'avoir une description abstraite de ces informations afin de pouvoir les utiliser.

2.6 Architecture d'un système sensible au contexte

Un système sensible au contexte est représenté par cinq couches : couche acquisition(capture) du contexte, couche d'Interprétation et d'agrégation du contexte,couche de stockage et historique du contexte, couche dissémination du contexte et la couche application.

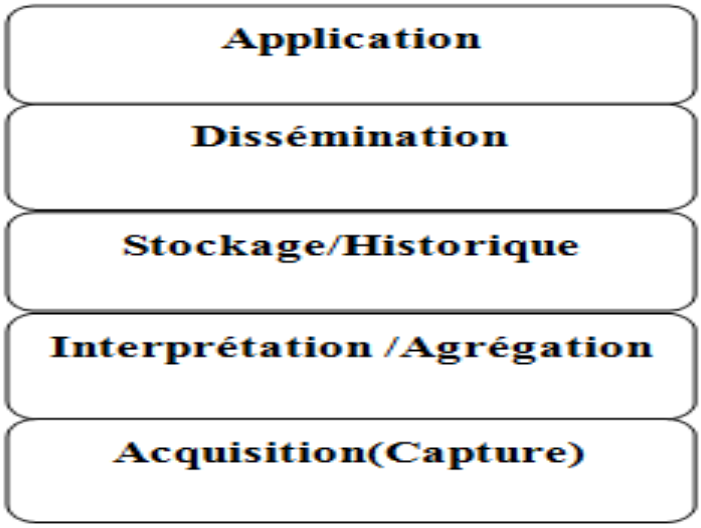


FIGURE 2.1 – Architecture générale d'un système sensible au contexte[2]

2.6.1 Couche acquisition(capture)du contexte

La première couche d'une architecture sensible au contexte est composée d'une collection de capteurs. Un capteur est une source matérielle ou logicielle qui peut générer une information contextuelle. Nous distinguons trois types de capteurs : physiques, virtuels et logiques [26].

Capteurs physiques

Les capteurs physiques sont des dispositifs matériels qui sont capables de fournir des données de contexte. La table 2.1 donne quelques exemples de capteurs physiques selon le type d'information qu'ils fournissent.

Type d'information fournie	**Capteurs disponibles**
Lumière	capteurs de couleurs, capteurs d'infrarouge et d'ultra violet...
Contexte visuel	Caméras numériques
Audio	Microphones
Localisation	GPS (Global Positionning System), GSM (Global System for Mobile Communications), Active Badge System[24]...
Mouvements et accélérations	capteurs d'angles, accéléromètres, détecteurs de mouvement, champs magnétiques...
Température	Thermométres numériques
Caractéristiques biologiques	Capteurs Biométriques (Tension, résistance de peau...)

TABLE 2.1 – Exemples de capteurs de contexte

Capteurs virtuels

Les capteurs virtuels fournissent des informations contextuelles à partir d'applications ou services logiciels. Par exemple, il est possible de détecter l'emplacement d'un livreur de marchandise en consultant son carnet électronique de rendez-vous sans avoir recours à des capteurs physiques. Les capteurs virtuels sont beaucoup moins coûteux que les capteurs physiques puisqu'ils sont basés sur des composants logiciels qui sont généralement moins chers que des appareils électroniques.

Capteurs logiques

Ce type de capteurs utilise généralement plusieurs sources d'information contextuelles pour fournir une autre information de synthèse plus précise. Ces capteurs peuvent réutiliser des

capteurs physiques et virtuels pour fournir un contexte de plus haut niveau.

2.6.2 Couche d'interprétation et d'agrégation du contexte

Cette couche offre des moyens d'interprétation des données contextuelles fournies par les capteurs du contexte. Elle sert à l'analyse et à la transformation des données brutes fournies par la couche de capture du contexte dans d'autres formats de haut niveau qui sont plus faciles à manipuler et à utiliser. En effet, les capteurs fournissent généralement des données techniques qui ne sont pas appropriées pour une utilisation directe par l'application. Les transformations effectuées sur les données brutes fournies par les capteurs peuvent être réalisées par plusieurs opérations : extraction, quantification, raisonnement, agrégation...etc. Par exemple, les coordonnées GPS d'une personne peuvent être moins significatives qu'une adresse physique sous forme de numéro de rue et de ville. [27].

La complexité des interprétations de contexte peut varier d'une simple agrégation de valeurs qui proviennent de plusieurs capteurs à des raisonnements ou analyses statistiques complexes. Par exemple, la localisation de plusieurs personnes dans une seule salle peut inférer le fait qu'ils sont en réunion. Dans ce cas, le niveau de bruit peut aussi être une information importante pour savoir s'ils sont en réunion de travail ou de loisir.

Cette couche doit aussi assurer la résolution de conflits causés par l'utilisation de plusieurs sources de contexte. En effet, ces sources peuvent donner des résultats contradictoires ou peuvent aboutir à des situations imprécises. Cette couche doit donc avoir une certaine intelligence d'interprétation pour résoudre ces conflits [27],[28].

2.6.3 Couche de stockage et historique du contexte

La troisième couche "stockage et historique du contexte" organise les données capturées et interprétées et les stocke pour une utilisation ultérieure. Ce stockage peut être centralisé ou distribué. La solution centralisée est l'option la plus répandue et la plus utilisée puisqu'elle facilite la gestion des mises à jours et des variations des valeurs du contexte. La gestion distribuée du contexte est beaucoup plus complexe puisqu'elle inflige des fonctions additionnelles de découvertes de ressources et d'actualisation des valeurs du contexte. De plus, cette gestion distribuée alourdit la tâche de l'application qui doit gérer la collecte des différentes informations contextuelles d'une façon interne.

2.6.4 Couche dissémination du contexte

Cette couche assure la transmission des différentes informations contextuelles à l'application. Ces informations sont distribuées sur différents lieux géographiques et proviennent de plusieurs types de dispositifs. Elle assure une transparence totale de la communication avec l'application. En conséquence, le développement de l'application devient plus simple. Sans cette couche, on serait amené à développer des protocoles de communications avec les différentes sources de contexte. La couche de dissémination du contexte offre des moyens de communication standards pour notifier l'application des changements de contextes et leur transmission à l'application. Plusieurs systèmes offrent des mécanismes de gestion d'évènements qui se basent essentiellement sur les fonctions de gestion de requêtes directes ou de notification [29]. L'application peut demander un accès direct à une information contextuelle précise mais elle peut aussi s'abonner pour recevoir tous les changements des valeurs de cette information. Ces deux fonctions principales assurent un moyen de communication transparent et efficace pour la dissémination des valeurs de contexte à l'application[30]. L'implantation de ces types de communication est nécessaire pour garantir la sensibilité au contexte dans une application.

2.6.5 Couche application

La couche application dans les systèmes sensibles au contexte existant [31] est représentée par l'application qui offre ses services aux différents clients concernés. Elle est responsable de l'extraction des informations des différentes sources de données attachées à l'application. Elle doit aussi implémenter les réactions nécessaires aux changements du contexte. Chaque application s'abonne à la couche de dissémination du contexte pour accéder aux différentes informations contextuelles et être informée de leurs changements. Une application peut accéder à ces informations de deux façons différentes : synchrone et asynchrone. Dans le cas d'un accès synchrone, l'application demande à la couche de dissémination de lui fournir une information contextuelle précise. Ceci est réalisé généralement par des appels directs de fonctions au niveau de la couche dissémination. Dans le cas de la communication asynchrone, l'application s'abonne à des évènements spécifiques qui correspondent à des changements de valeurs de contexte. Dès qu'un événement est déclenché, l'application est simplement notifiée ou bien l'un de ses services est directement invoqué en utilisant des fonctions de callback implémentées dans la couche dissémination.

2.7 Exigences des gestionnaires du contexte

Plusieurs conditions doivent être prises en compte lors de la modélisation du contexte d'informations, Bittini et al. Dans [32] ont présenté certains d'entre eux dans les points suivants :

a) L'hétérogénéité et la mobilité : Les modèles de contextes doivent faire face à une grande variété de sources de contexte qui différent suivant leurs taux de mis à jour et leur niveau sémantique, les modèles de contextes doivent ainsi être capable d'exprimer différents types d'information et le système de gestion de contexte doit fournir une gestion de ces informations suivant leurs types.

b) Relations et dépendances : Il existe différentes relations entre les types d'informations contextuelles capturés pour assurer un comportement correct des applications. Une telle relation est la dépendance où les entités d'information de contexte peuvent dépendre d'autres entités d'informations de contexte : par exemple, un changement de la valeur d'une propriété (exemple, le réseau bande passante) peut influer sur les valeurs d'autres propriétés (par exemple, autonomie de la batterie puissance).

c) Raisonnement : Les applications sensibles au contexte utilisent des informations de contexte pour évaluer s'il y a un changement de contexte de l'utilisateur et/ou la situation de l'environnement ; de prendre une décision si une adaptation à ce changement est nécessaire, qui, souvent requièrent des capacités de raisonnement. Les Techniques de raisonnement peuvent également être adoptées pour obtenir plus d'informations de contexte. Par conséquent, il est important que les techniques de modélisation du contexte soient en mesure de soutenir à la fois la vérification de cohérence, et le raisonnement sur des situations complexes.

d) Timeliness : Les applications sensibles au contexte peuvent nécessiter l'accès à des états précédant et états futurs. Ainsi, histoires de contexte est une autre caractéristique des informations de contexte qui doit être capturé par les modèles contextuels. La gestion de l'historique du contexte est difficile si le nombre de mises à jour est très élevé.

e) Imperfection (incomplétude) : En raison de sa nature dynamique et hétérogène, des informations de contexte peut être de qualité variable ou même incorrect. La plus part des capteurs disposent d'une inexactitude inhérente (par exemple, de quelques mètres pour des positions GPS), les informations de contexte peuvent être incomplètes : un capteur qui détecte le nombre de personnes dans une pièce peut manquer quelqu'un. ainsi, une bonne approche de modélisation de contexte doit prendre en considération le paramètre de qualité des informations du contexte.

2.8 Approches de modélisation 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. Nous proposons dans cette section deux types d'approches : approches classiques et approches hybrides.

2.8.1 Modèles classiques

Nous résumons les approches de modélisation de contexte les plus connues dans la littérature, ils sont classifiés en quatre types d'approches[3] : l'approche clés/valeurs, l'approche orientée modèle, modèles basée sur la logique et l'approche orientée ontologie.

2.8.1.1 Approches clés/valeurs

Plusieurs architectures présentent le contexte sous forme de paires (attribut, valeur). Où l'attribut représente un nom d'une information contextuelle et la valeur représente la valeur actuelle de cette information. Par exemple, *(Name = "context1", User = "doctorEH102", Localisation = "Edouard Herriot Hospital", Time = "Mon Jul 09 16 :51 :20 CEST 2007")*. Le contexte *context1* est défini par " l'utilisateur x est localisé dans un emplacement y à un temps t".

Discussion

La modélisation clés-valeur est la modélisation la plus simple, elle présente l'avantage de la facilité d'implantation. En effet, la gestion du contexte revient à parcourir la liste des contextes disponibles. Cependant, cette modélisation manque d'expression de relations et de complétude. En effet, sa structure trop "plate" ne permet pas de définir tous les aspects contextuels de l'application, mais qu'une seule correspondance exacte. Ce genre de modéle est aussi une source de conflits. Par exemple, si nous définissons un nouveau contexte : *(Name = "context2", User = "x", Localisation = "z", Time = "t")* avec l'emplacement z est dans l'hôpital Edouard Herriot (par exemple la chambre 220 de l'hôpital), on ne peut pas dire que *context2* est un sous contexte de *context1* et que toutes les fonctionnalités offertes par l'application dans *context1* doivent aussi exister dans *context2*.

2.8.1.2 Approches orientées modèle

L'approche orientée modèle utilise des modèles formels pour modéliser les informations de contexte. Son objectif principal est d'offrir la possibilité d'encapsuler le contexte et de permettre sa réutilisation. Nous décrivons quelques approches appartenant à cette catégorie de modélisation, les informations qui peuvent être décrites avec ces modèles, et les outils utilisés par chaque modèle pour décrire les informations de contexte.

- **Unified Modeling Language (UML)**

  Bauer[33] a utilisé le langage UML afin de modéliser le contexte auquel une application de gestion de trafic aérien est sensible. Mais, le modèle proposé est spécifique à l'application et ne peut être utilisé dans d'autres applications sans modification.

  Sheng et Benatallah [34] 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*.Comme l'illustre la figure 2.2, ce méta-modèle est composé de plusieurs classes qui permettent de créer des services sensibles au contexte. Une classe permet de décrire un contexte observable. Ce dernier peut être un contexte de bas niveau ou bien un contexte interprété. Pour chaque contexte de niveau bas, le modèle permet de spécifier la source à partir de laquelle il a été collecté. Le méta-modèle *ContextUML* permet aussi la description des actions d'adaptation et les situations pertinentes qui permettent de les déclencher.

  Discussion

  Le méta-modèle *ContextUML* est un modèle basé sur l'UML, il est caractérisé par sa généricité, il permet de décrire le contexte de manière plus riche que les approches clés/valeurs. Mais, il ne prend pas en considération la description des relations de dérivation et de dépendance entre les informations de contexte. De plus, ce méta-modèle n'offre pas le moyen de décrire la qualité des informations de contexte ni leur validité temporelle.

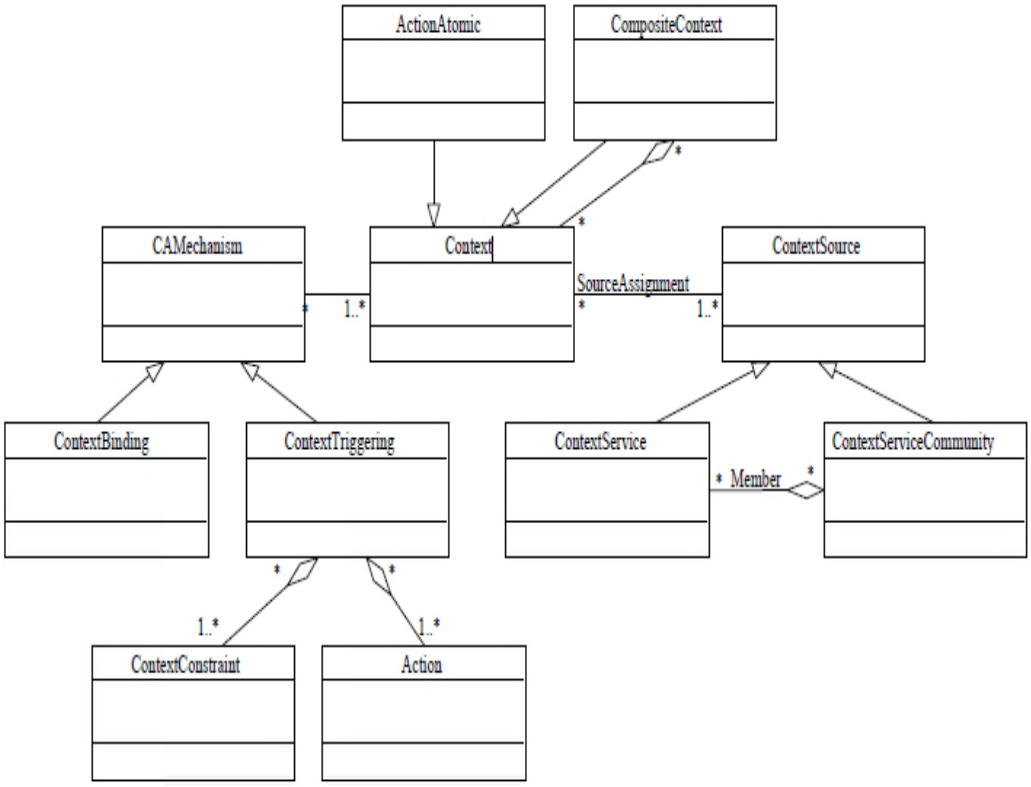


FIGURE 2.2 – Un méta-modèle de ContextUML[3]

- **Context Modeling Langage (CML)**

  Afin de pouvoir modéliser les caractéristiques des informations de contexte et leurs propriétés, *Henricksen et al.*[35][36] ont proposé un langage orienté objet dérivé de l'ORM *(Object Role Modeling)*[37]. Ce langage appelé CML *(Context Modeling Language)*[38] 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 profil) et les dépendances entre ces informations. CML permet aussi de spécifier la qualité de chaque information observée et sa validité temporelle.

  Dans cette modélisation, les informations de contexte sont groupées en un ensemble d'entités, chacune d'elle décrit un objet conceptuel ou physique tel qu'une personne, un dispositif ou un réseau. Les propriétés des entités telles que le nom de la personne, le nom du dispositif ou l'identificateur du réseau sont représentées par des attributs. Les entités sont liées à leurs attributs ou à d'autres entités à travers des relations. À l'aide de ces relations, le concepteur peut classifier les informations de contexte associées à chaque entité. S'il

utilise une association de type *sensed* entre une entité et son attribut, cela veut dire que cet attribut est un contexte obtenu à partir de capteurs matériels ou logiciels, comme la localisation d'une personne.

La figure 2.3 illustre un exemple de modélisation de contexte avec loutil CML en utilisant différents types dassociations entre les informations modélisées.

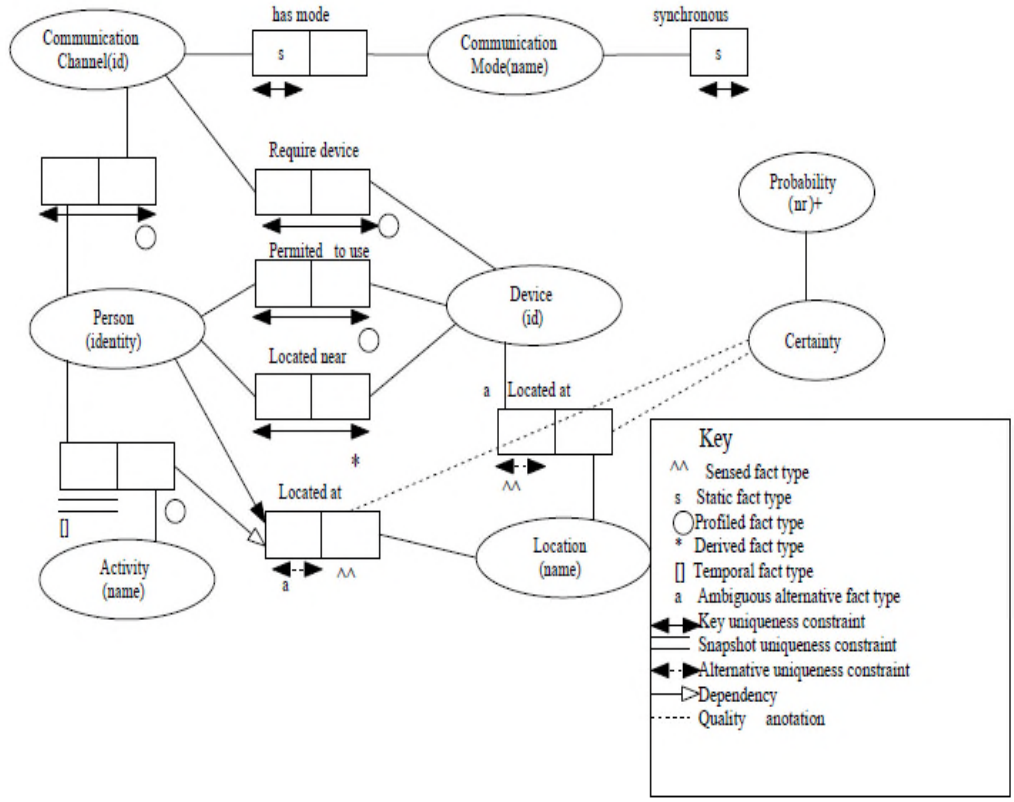


FIGURE 2.3 – Exemple de modélisation avec CML[3]

Discussion

CML est venu remédier à certains des manques de l'UML en proposant un modèle avec visualisation graphique qui permet de décrire un ensemble de relations entre plusieurs contextes observables et de typer le contexte, ce qui entraine une gestion plus complexe . Les approches orientées modèles permettent de décrire le contexte de manière plus pratique que les approches clés/valeurs et elles offrent aux concepteurs d'applications la possibilité de réutiliser le modèle pour d'autres applications.

2.8.1.3 Approches basées sur la logique

Les modèles basés sur la logique sont caractérisés par un très grand degré de formalité. Ils utilisent l'algèbre booléenne[1] 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.

La première approche de modélisation du contexte en utilisant la logique a été publiée en 1993 par *McCarthy et son équipe* [39],[40]. *McCarthy* définit le contexte comme une entité mathématique abstraite ayant des propriétés. Cette formalisation logique est fondée sur une réification du contexte et un méta-prédicat *ist ; ist(p,c)* signifie que l'assertion p est vraie dans le contexte c. Par exemple la formalisation c : *(ist(contextof("Histoire de Sherlock Holmes"), "Sherlock Holmes est un détective")* considère que le personnage Sherlock Holmes est un détective dans l'histoire de Sherlock Holmes. Ce type de modélisation est utilisé dans le domaine de l'intelligence artificielle où les connaissances sont regroupées en micro-théories [41] ; selon les valeurs du contexte, on se place dans ou hors d'une micro-théorie.

Discussion

Les approches basées sur la logique utilisent l'algèbre booléenne ou la logique du premier ordre pour modéliser le contexte dans le but de raisonner sur les informations collectées. 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.

2.8.1.4 Approches orientées ontologie

Une ontologie est une description sémantique, structurée et formelle des concepts d'un domaine et de leurs inters-relations [42]. En informatique, une ontologie est définie comme un ensemble structuré de savoirs dans un domaine particulier de la connaissance ou un ensemble

1. Algèbre booléenne ou calcul booléen, est la partie des mathématiques, de la logique et de l'électronique qui s'intéresse aux opérations et aux fonctions sur les variables logiques. Elle permet de modéliser des raisonnements logiques, en exprimant un "état" en fonction de conditions. Par exemple : communication= Emetteur ET Récepteur Communication est "VRAI" si Emetteur actif et Récepteur actif.

de concepts organisés en graphe dont les relations peuvent être sémantiques, de composition ou d'héritage.

Une ontologie est un ensemble de classes et de relations existant entre ces classes, de propriétés attachées aux classes et d'axiomes [43]. La création d'une ontologie se fait avec un langage logique, de façon à ce que l'on puisse faire des distinctions détaillées, précises, cohérentes et logiques entre les classes, les propriétés et les relations. Les ontologies sont caractérisées par la possibilité de partager des connaissances entre plusieurs systèmes. De plus, elles sont ouvertes et extensibles, ce qui permet à chaque système de les enrichir et dexploiter les notions qui y sont déjà définies. En effet, les langages d'ontologies offrent le moyen de publier, d'étendre des ontologies existantes et d'employer diverses ontologies existantes pour compléter une nouvelle ontologie.

À chaque ontologie, on peut associer un moteur d'inférence [2][44] qui permet de raisonner sur les informations de contexte en exécutant des règles d'inférence. Les moteurs d'inférence fournissent un ensemble d'opérations basiques prédéfinies (opérations de comparaison et d'ajout d'instances dans l'ontologie... etc) et offrent aux développeurs la possibilité de définir leurs propres opérations.

La Figure 2.4 représente un exemple d'ontologie,dans cette ontologie, il y a 8 classes : Personne, TravailleurManuel, Plombier, Politicien, Métier, MétierManuel, Plomberie et Politique. Les flèches is-a représentent les relations de super-classe/sous-classe. La relation aMétier relie Personne à Métier, ce qui veut dire que les personnes peuvent avoir un métier. Les relations père, oncle et frère relient la classe Personne à elle-même, car le père, l'oncle ou le frère d'une personne sont aussi des personnes.

Discussion

Les approches orientées ontologie sont des approches formelles qui tirent parti des caractéristiques des ontologies pour modéliser le contexte. En effet, les caractéristiques de partage et de distribution des données ont été exploitées afin de définir des méta-modèles de description du contexte. De plus, les moteurs d'inférence fournis par les ontologies ont été utilisés pour déduire des contextes de haut niveau à partir des données collectées.

2. Un programme qui effectue des déductions logiques d'un système à partir d'une base de connaissance et d'une base de règles. Les règles sont utilisées pour manipuler les connaissances et aboutir à des conclusions

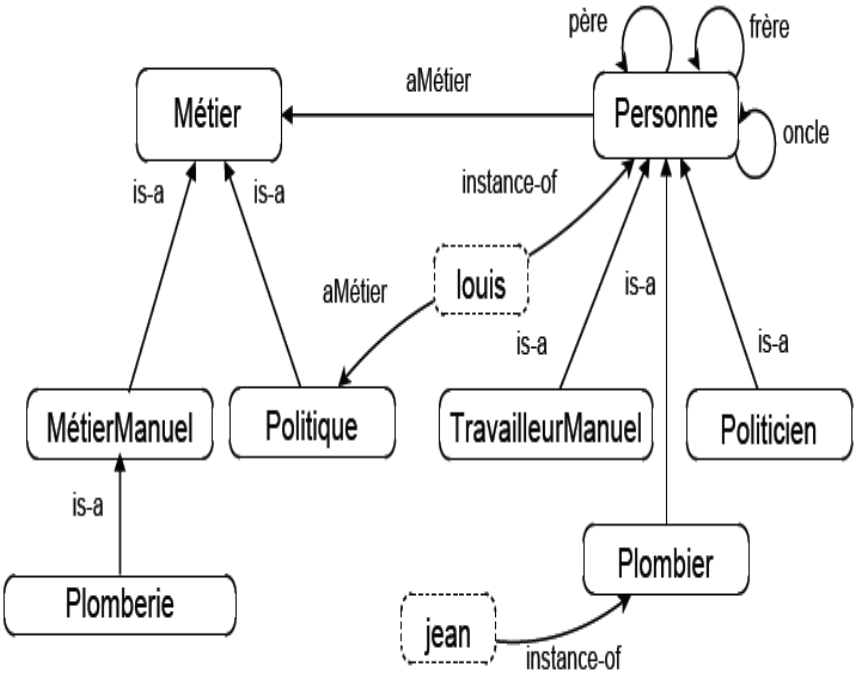


FIGURE 2.4 – Exemple d'ontologie[4]

Synthèse

Le modèle clés-valeurs est plus simple et facile à implémenter. Cependant, cette modélisation manque d'expression de relations et de complétude. De plus elle ne permet pas de définir tous les aspects contextuels de l'application, elle est aussi une source de conflits.

Les approches orientées modèle sont caractérisées par leurs possibilité de réutilisation dans d'autres applications. Néanmoins elles ne permettent pas de raisonner sur des informations contextuelles ni la publication de ces dernières.

Pour ce qui consterne les approches basées sur la logique, elles sont très efficaces pour raisonner sur le contexte et déduire des actions de réaction si une situation pertinente est détectée. Cependant elle ne permet pas de décrire la validité temporelle des informations ni les relations qui peuvent exister entre les informations de contexte.

Enfin, les modèles ontologiques ont des avantages évidents en termes d'expressivité et de l'interopérabilité, pour la représentation des relations complexes et les dépendances entre les données de contexte. Toute fois ces modèles restent souvent difficiles à implémenter dans des cas réels.

Modèle	Clés/Valeurs	Orienté Modèle	Basée sur la logique	Basée sur l'ontologie
Caractéristiques	Simplicité d'utilisation, pauvreté d'expression.	Utilisation d'un modèle global	Très formel	Difficile à implémenter dans le cas réel
Relations et dépendance	Non	Oui	Non	Oui
Raisonnement	Non	Non	Oui	Oui
Publication des données	Non	Non	Non	Oui

TABLE 2.2 – Comparaison entre les modèles classiques existants

2.8.2 Modèles hybrides

Un modèle hybride est une combinaison entre les modèles classiques, en tenant en compte des avantages de chacun pour la modélisation du contexte qui tente d'intégrer différents modèles et différents types de raisonnement afin d'obtenir des systèmes plus souples et généraux. Nous présentons dans ce qui suit les deux approches hybrides[32] : Modèle hybride fait/ontologie et le modèle balisage/ontologie.

2.8.2.1 Modèle hybride faits/ontologie

Henricksen et al.[45]] proposent une approche de modélisation du contexte hybride, combinant les ontologies et l'approche basée sur les faits fournis par le langage CML. L'objectif est de combiner les avantages de modèles CML (en particulier le traitement des ambiguïtés et des informations de contexte imparfaites) et ceux du modèle ontologique notamment l'interopérabilité et les divers types de raisonnement. L'approche hybride est basée sur une cartographie de CML pour modélisation les classes et les relations OWL-DL.

Discussion

En raison de certaines limitations d'expressivité d'OWL-DL, une cartographie complète entre CML et OWL-DL ne peut pas être obtenue. En ce qui concerne les questions d'interopérabilité, les avantages acquis par une représentation ontologique du contexte sont clairement reconnaissables. Cependant, par rapport à la dérivation de nouvelles données de contexte, les

expériences avec le 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.

2.8.2.2 Le modèle hybride balisage/ontologie

CARE[46] adopte pour la sensibilité au contexte une approche de modélisation du contexte basée sur une interaction entre un modèle de balisage et un modèle ontologique. L'interaction entre ces modèles est réalisée à travers la représentation des données de contexte par l'intermédiaire CC/PP qui contiennent une référence aux classes OWL-DL et leurs relations. Afin de préserver l'efficacité. Chaque fois que de nouvelles données pertinentes du contexte 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) dérivées par un raisonnement ontologique peuvent être utilisées comme pré-conditions de la règle pour dériver de nouvelles données de contexte telles que les préférences de l'utilisateur.

Discussion

Ce modèle a comme intermédiaire CC/PP qui est basé sur RDF et qui utilise une structure de profils, c'est un modèle dont le vocabulaire est pauvre, il est restreint à la description de profil. Cependant, il est réutilisable et interopérable.

Synthèse

Modèle	Faits/ Ontologie	Balisage/Ontologie
Caractéristiques	Basé sur CML	Intermédiaire CC/PP
Relations et dépendance	Oui	Oui
Raisonnement	Faible	Faible
Publication des données	Oui	Oui
Expressivité	Aucun avantage	Aucun avantage

TABLE 2.3 – Comparaison entre les modèles hybrides existants

Le modèle hybride faits/ontologie a l'avantage d'être interopérable, grâce à une représentation ontologique du modèle du contexte. Cependant, il ne permet pas la dérivation de nouvelles données de contexte.

Le modèle hybride Balisage/Ontologie assure l'interopérabilité grâce à la caractéristique de l'ontologie, mais il n'exprime pas la validité temporelle.

2.9 Conclusion

L'objectif de l'étude que nous avons effectuée consiste à montrer l'apport de chaque approche de modélisation de contexte. Notre choix d'une approche de modélisation comporte plusieurs critères : l'expressivité du modèle, l'interopérabilité,le raisonnement sur le contexte,la capacité d'exprimer les relations qui existent entre les informations contextuelles, la possibilité de sa réutilisation, de son extension et la publication des informations qu'elle permet de décrire.

Dans le prochain chapitre, nous allons étudier le modèle logique Event Calculus et le langage ontologique OWL.

3

Etude du modèle loqique Event Calculus et du modèle ontologique OWL

3.1 Introduction

Dans ce chapitre, nous exposerons l'approche de modélisation Event calculus, ses différents concepts ainsi que ses différentes versions et nous présenterons la logique de description et pour finir nous donnerons un aperçu du langage ontologique OWL.

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.

3.2 Event calculus

Event calculus est introduit par Kowalski et Sergot[47] 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.

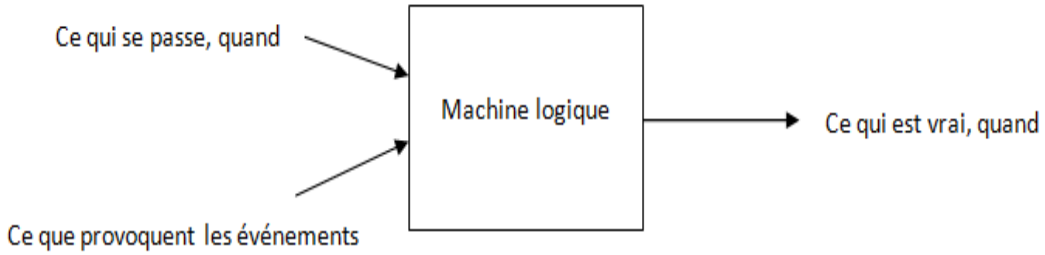


FIGURE 3.1 – Le fonctionnement de Event Calculus

3.2.1 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.
  E (variables e,e_1,...)
- ***Fluent*** : défini 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.
  F variables (f,f_1,...)
- ***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 :$<, >, >=, <=, =$.
  T (variables t, t_1,...)
- ***Prédicats*** : Les prédicats permettent
  - D'exprimer quels événements se produisent, et quand ils vont se produire.
  - De décrire les effets de ces événements.
  - De 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.

3.3 Les capacités représentationnelles de EC

3.3.1 La circonscription

Dans EC, la circonscription est le fait de ne considérer que les événements connus (il n'y a pas d'autres types d'events), et le raisonnement ne s'effectuera que sur ces événements connus [48].

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

3.3.2 Le raisonnement révisable

Le raisonnement révisable est un type particulier de raisonnement non démonstratif, où le raisonnement ne produit pas une démonstration complète ou définitive d'une assertion [49].

Dans un formalisme à raisonnement révisable [50], un argument est utilisé comme révisable pour appuyer des conclusions. Une conclusion sera considérée comme valide uniquement lorsque l'argument qui l'appuie 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.

3.3.3 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.

3.3.4 Le changement continu

La nécessité de représenter les changements continus est une exigence bien établie dans les schémas de représentation en IA [51].

Dans EC, ce problème est résolu en intégrant, les prédicats ***Trajectory*** et ***Antitrajectory***. ***Trajectory*** apparait dans les premiers travaux de shanahan. ***Antitrajectory*** est ajouté dans EC par la suite par *Miller* et *Shanahan* [52],[53],[54].

cond $\Rightarrow$ ***Trajectory (f1, t1, f2, t2)***

f1, ***f2*** représentent les fluents qui varient dans l'intervalle [t1, t2[.

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

3.3.5 La loi d'inertie

Le concept de la loi d'inertie est d'abord défini par référence au calcul de situation par *Lifschitz* [55] 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. Cela se traduit par EC dans les prédicats ***Release*** et ***ReleasedAt***, présentés par *Miller* et *Shanahan* [54]. 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***.

3.3.6 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***.

3.4 Caractéristiques de EC

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

3.4.1 Parcimonie de la représentation

EC permet de résoudre le problème de cadre (frame problem : concerne la représentation des non effets 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 [56].

3.4.2 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 continu (countinous change).

3.4.3 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 [57].

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.

3.5 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.

3.5.1 Original Event Calculus (OEC)

OEC est introduit par Kowalski et Sergot [47]. 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 3.1 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 : $After(e1, f) = Before(e2, f) \supset End(After(e1, f), e2)$.

OEC7 : $Holds(After(e1, f)) \wedge Holds(Before(e2, f)) \wedge e1 < e2 \wedge \neg Broken(e1, f, e2) \supset$
$After(e1, f) = Before(e2, f)$.

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

OEC est la conjonction entre ***OEC1*** et ***OEC8***

Où :

$\supset$: représente une implication.

$\equiv$: représente une bi-implication.

Sachant que (e, e1, e2 = occurrences d'event, f, f1, f2 = fluents, p = période de temps)

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 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.

3.5.2 Simplified Event Calculus (SEC)

Simplified Event Calculus(SEC) est proposé par Kowalski en 1986[58] et développé par Sadri[59], Eshghi[60], et Shanahan[61].

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 3.2. Les axiomes de SEC sont les suivants :

SEC1 :$((Initially(f) \wedge \neg StoppedIn(0, f, t)) \vee \exists e, t1(Happens(e, t1) \wedge Initiates(e, f, t1) \wedge t1 < t \wedge \neg StoppedIn(t1, f, t))) \equiv HoldsAt(f, t).$

SEC2 :$StoppedIn(t1, f, t2) \equiv \exists e, t(Happens(e, t) \wedge t1 < t < t2 \wedge Terminates(e, f, t)).$

SEC est la conjonction entre ***SEC1*** et ***SEC2***.

Sachant que (e, e1, e2 = type d'event, f, f1, f2 = fluents, t = instants de temps)

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

3.5.3 Basic Event Calculus (BEC)

Shanahan [56],[62] 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 3.3. 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 : $Happens(e, t1) \wedge Terminates(e, f, t1) \wedge t1 < t2 \wedge \neg StartedIn(t1, f, t2) \supset \neg HoldsAt(f, t2)$.

BEC est la conjonction ***BEC1*** entre ***BEC7***.

Sachant que (e, = type d'event, f, f1, f2 = fluents, t1,t2 = instants de temps)

Prédicat ou fonction	Le sens
InitiallyN(f)	f est faux à l'insatant 0
InitiallyP(f)	f est vrai à l' instant t
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

3.5.4 Event Calculus (EC)

Miller et Shanahan [54],[48] ont introduit plusieurs formulations alternatives de BEC. Un certain nombre de leurs axiomes peuvent être combinés [63] pour produire ce que nous appelons Event Calculus EC, qui diffère d'event calculus de base dans les points suivants :

- Il permet le temps négatif. Les points temporels 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 table3.4. Les axiomes de EC sont les suivants :

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

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

EC3 : $StoppedIn(t1, f, t2) \equiv \exists e, t(Happens(e, t) \wedge t1 < t < t2 \wedge Terminates(e, f, t))$.

EC4 $: StartedIn(t1, f, t2) \equiv \exists e, t(Happens(e, t) \wedge t1 < t < t2 \wedge Initiates(e, f, t))$.

EC5 $: Happens(e, t1) \wedge Initiates(e, f1, t1) \wedge 0 < t2 \wedge Trajectory(f1, t1, f2, t2) \wedge \neg StoppedIn(t1, f1, t1 + t2) \supset HoldsAt(f2, t1 + t2)$.

EC6 $: Happens(e, t1) \wedge Terminates(e, f1, t1) \wedge 0 < t2 \wedge AntiTrajectory(f1, t1, f2, t2) \wedge StartedIn(t1, f1, t1 + t2) \supset HoldsAt(f2, t1 + t2)$.

EC7 $: PersistsBetween(t1, f, t2) \equiv \neg\exists t(ReleasedAt(f, t) \wedge t1 < t \leq t2)$.

EC8 $: ReleasedBetween(t1, f, t2) \equiv \exists e, t(Happens(e, t) \wedge t1 \leq t < t2 \wedge Releases(e, f, t))$.

EC9 $: HoldsAt(f, t1) \wedge t1 < t2 \wedge PersistsBetween(t1, f, t2) \wedge \neg Clipped(t1, f, t2) \supset HoldsAt(f, t2)$.

EC10 $: \neg HoldsAt(f, t1) \wedge t1 < t2 \wedge PersistsBetween(t1, f, t2) \wedge \neg Declipped(t1, f, t2) \supset \neg HoldsAt(f, t2)$.

EC11 $: ReleasedAt(f, t1) \wedge t1 < t2 \wedge \neg Clipped(t1, f, t2) \wedge \neg Declipped(t1, f, t2) \supset ReleasedAt(f, t2)$.

EC12 $: \neg ReleasedAt(f, t1) \wedge t1 < t2 \wedge \neg ReleasedBetween(t1, f, t2) \supset \neg ReleasedAt(f, t2)$.

EC13 $: ReleasedIn(t1, f, t2) \equiv \exists e, t(Happens(e, t) \wedge t1 < t < t2 \wedge Releases(e, f, t))$.

EC14 $: Happens(e, t1) \wedge Initiates(e, f, t1) \wedge t1 < t2 \wedge \neg StoppedIn(t1, f, t2) \wedge \neg ReleasedIn(t1, f, t2) \supset HoldsAt(f, t2)$.

EC15 $: Happens(e, t1) \wedge Terminates(e, f, t1) \wedge t1 < t2 \wedge \neg StartedIn(t1, f, t2) \wedge \neg ReleasedIn(t1, f, t2) \supset \neg HoldsAt(f, t2)$.

EC16 $: Happens(e, t1) \wedge Releases(e, f, t1) \wedge t1 < t2 \wedge \neg StoppedIn(t1, f, t2) \wedge \neg StartedIn(t1, f, t2) \supset ReleasedAt(f, t2)$.

EC17 $: Happens(e, t1) \wedge (Initiates(e, f, t1) \vee Terminates(e, f, t1)) \wedge t1 < t2 \wedge \neg ReleasedIn(t1, f, t2) \supset \neg ReleasedAt(f, t2)$.

EC est la conjonction entre ***EC5***, ***EC6***, ***EC9***, ***EC10***, ***EC11***, ***EC12***, ***EC14***, ***EC15***, ***EC16***, et ***EC17*** .

Sachant que (e = type d'event, f, f1, f2 = fluents, t1,t2 = instants de temps).

3.5.5 Discrete Event Calculus (DEC)

Mueller [63],[64] 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 3.4. Les axiomes et les définitions de DEC sont les suivants :

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

Prédicat ou fonction	Le sens
HoldsAt(f, t)	f 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 est libéré de la loi d'inertie après t.
Trajectory(f1, t1, f2, t2) if f1 is initiated	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.
Clipped(t1, f, t2)	le fluent f est terminé entre l'instant t1 et t2.

TABLE 3.4 – Prédicats et fonctions de EC.

i.e : Un fluent f est arrêté entre l'instant t1 et t2 s'il y a un event qui termine f après t1 et avant t2.

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

i.e : Un f fluent est commencé entre l'instant t1 et t2 s'il y a un event qui lance f après t1 et avant t2.

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

i.e : Si un event e se produit pour lancer le fluent f1 à l'instant t1 et s'il y a un Trajectory qui fait à ce déclenchement fluent un autre f2 après une période le t2, alors le fluent f2 est vrai à t1+t2, supposant que f1 n'est pas arrêté entre t1 et t1+t2.

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

i.e : Ceci définit l'équivalent au DEC3 pour un Antitrajectory, ainsi si un événement se produit pour terminer le fluent f1 à t1 et un Antitrajectory fait que f1 déclenche f2 à t1+t2, alors le f2 est vrai à t1+t2, supposant que f1 n'est pas remis en marche entre le t1 et t1+t2.

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

i.e : Si un fluent f est vrai à l'instant t et n'est pas libéré de la loi d'inertie à l'instant t et un event ne se produit pas à t pour terminer fluent f à l'instant t, alors le fluent f se produira t+1.

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

i.e : C'est identique au DEC5, sauf qu'il traite des cas où fluent f ne se produit pas à t et implicitement à t+1.

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

i.e : Ceci déclare que si un fluent f est libéré de la loi d'inertie à t et un événement ne se produit pas pour terminer fluent f à t, alors le fluent f sera toujours libéré de la loi d'inertie à t+1.

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

i.e : C'est identique au DEC7, sauf qu'il traite des cas où f n'est pas libéré à t et implicitement à t+1.

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

i.e : Si un event e initié un fluent f à l'instant t, alors f est vrai à t+1.

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

i.e : Si l'event e termine f à t, alors f ne se produira pas à t+1.

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

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

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

i.e : Si un event e initié ou termine un fluent f à t, alors f ne sera pas libéré de la loi d'inertie t+1.

DEC est la conjonction entre ***DEC1*** et ***DEC2***.

3.6 Tableau comparatif des différentes versions de EC

Version	Temps	Changement continu	Concurrence	Circonscription	Contradictions
OEC	Période	Non	OUI	OUI	NON
SEC	Instant Réel > 0 Entier> 0	NON	OUI	OUI	OUI
SEC	Instant Réel > 0 Entier> 0	NON	OUI	OUI	OUI
BEC	Instant Réel > 0 Entier> 0	OUI	OUI	OUI	OUI
EC	Instant Réel ou Entier	OUI	OUI	OUI	OUI
DEC	Instant Entier	OUI	OUI	OUI	OUI

TABLE 3.5 – Tableau comparatif des différentes versions de event calculus

Nous avons choisi la version DEC parmi les autres versions pour sa capacité de représenter le changement continu, détecter la contradiction si elle existe dans la base de connaissance ($HoldsAt(f,t)$ et $\neg HoldsAt(f,t)$) , les événements concurrents, la circonscription de plus il limite le temps à des entiers.De plus, c'est la seul version qui a un raisoneur implémenté (DEKT Reasoner implémenté en 2010)

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 lesquelles nous avons opté pour EC.

3.7.1 Situation Calculus

Situation Calculus est proposé par *Mc Carthy* et *Hayes* en 1961[65]. Il donne une définition logique des concepts fluents 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 [66] 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 longs que EC. Par exemple : quand il s'agit 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.

Disscussion

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'état 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).

3.8 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[67].

3.8.1 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.

3.8.2 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		aEnfant, 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	Femme$\equiv$Personne $\sqcap$ Feminin Une Femme est une Personne et Feminin
$\neg$	Négation	$\neg$A	$\neg$Humain Les individus qui ne sont pas Humain

3.8.3 La ABox et la TBox

Dans une base de connaissances en logique descriptive, on distingue deux composantes : Terminological Box (TBox) et Assertion Box (ABox). La première contient tous les axiomes

$\bot$	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
$\forall$	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.$\top$	$\exists$aEnfant.$\top$ 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
I	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 ($\leq$ ou$\geq$)	$\geq$n R $\leq$n R	Femme$\sqcap\geq$0 aEnfant (Femme qui n'a pas d'enfants) FemmeMarié$\equiv$Femme $\sqcap\exists$mariéAvec.$\top\sqcap\leq$1 mariéAvec (Femme mariée avec au plus 1 seul homme)
Q	Restriction qualifiée($\leq\geq$)	$\geq$n R.C $\leq$n R.C	FemmeMarié$\equiv$Femme$\sqcap$ $\exists$mariéAvec.T$\sqcap\leq$1 mariéAvec.Homme

TABLE 3.6 – Axiomes et concepts de base de AL

définissant les concepts du domaine, comme la définition du concept mère qui est défini comme étant une femme qui est le parent d'au moins un humain . La ABox 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.
Exemple : voici un exemple d'ontologie.

TBox

$Clibataire \equiv Personne \sqcap \leq 0 mariAvec$

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

$Homme \sqsubseteq Personne$

$Femme \sqsubseteq Personne$

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

$PreDeFilles \equiv Pre \sqcap aEnfant.Femme$

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

ABox

Homme(Bernard)

age(Bernard,56)

Femme(Sabine)

age(Sabine,46)

Homme(Valentin)

age(Valentin,10)

mariéAvec(Bernard,Sabine)

aEnfant(Bernard,Valentin)

3.8.4 Inférence

3.8.4.1 Inférence au niveau TBox

En logique descriptive, il y a quatre propriétés qu'on peut être intéressé à prouver pour une TBox T :

- **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 :

$$C1 \equiv Homme \sqcap Femme$$

- **Subsomption**

  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 :

  $$Homme \sqsubseteq Humain$$

- **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 :

  $$Humain \equiv Personne$$

- **Disjonction**

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

  $$Homme \sqcap Femme \sqsubseteq \bot$$

3.8.4.2 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 :

- **Vérifécation d'instance**

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

- **Vérification de consistance**

  Une ABox A est consistante par rapport à une TBox T, s'il existe une interprétation [1]qui est un modèle des deux, A et T.

  Exemple : si on définit une classe (concept) comme étant à la fois une sous-classe des classes Homme et Femme, et que la TBox spécifie aussi que ces 2 classes sont disjointes (aucun individu ne peut à la fois être un Homme et une Femme) : ce nouveau concept est alors inconsistant.

3.9 Les ontologies et le langage ontologique OWL

Comme nous l'avons définie dans le chapitre précédent, une ontologie est un ensemble structuré de consente représentant le sens d'un domaine de connaissance dont l'objectif en informatique ubiquitaire est de représenter le contexte des situations. Elle comprend : les concepts de

1. Une interprétation revient à donner une situation concrète. C'est-à-dire que l'on va affecter à chaque constante un personnage et indiquer ce qui est vrai ou faux pour ceux-ci.

l'ontologie, les instances des concepts (bases de connaissance) et la relation entre les instances.

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 2001 un groupe de travail qui a abouti à la recommandation OWL (Ontology Web Language) en février 2004 [43]. OWL définit une syntaxe basée sur RDF/XML pour décrire et construire des vocabulaires afin de créer des ontologies.

Le langage OWL offre trois sous-langages d'expression croissante conçus pour des communautés de développeurs et d'utilisateurs spécifiques : OWL-Lite, OWL-DL et OWL-Full.

- Le langage OWL-Lite concerne les utilisateurs ayant principalement besoin d'une hiérarchie de classification et de mécanismes de contraintes simples. Par exemple, les contraintes de cardinalité, en effet il ne permet que des valeurs de cardinalité de 0 ou 1.
- Le langage OWL-DL concerne les utilisateurs souhaitant une expressivité maximum sans sacrifier la complétude de calcul et la décidabilité des systèmes de raisonnement. Le langage OWL-DL comprend toutes les structures de langage OWL avec des restrictions comme la séparation des types (une classe ne peut pas être en même temps un individu ou une propriété). Le langage OWL-DL, conçu pour gérer le secteur existant de la logique de description.
- Le langage OWL-Full est la version la plus complexe de OWL, mais également celle qui permet le plus haut niveau d'expressivité. Toutefois, il ne garantit pas la décidabilité des calculs liés à l'ontologie (tous les calculs se font dans un intervalle de temps infini).

Tout document OWL est une ontologie, qui peut avoir un identificateur unique représenté par URI, et qui contient des faits et des axiomes. Les faits sont des descriptions d'individus, alors que les axiomes fournissent les descriptions de concepts. Un document OWL a donc la forme suivante :

ontologie : := Ontology([ontologieID]directive)

directive : := axiome | fait

Donc, l'ontologie OWL la plus simple que l'on peut écrire (et sûrement pas la plus intéressante) est celle-ci :

ontology()

Notons, finalement, que OWL comprend les deux classes pré-définies *owl :Thing* et *owl :Nothing* qui correspondent, respectivement, à $\top$ et $\bot$.

3.9.1 Définition des éléments d'une ontologie OWL

Nous fournissons des définitions nécessaires pour comprendre le principe des ontologies OWL[4].

- **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
- **Classe :** Une classe définit un groupe d'individus possédant des caractéristiques similaires. Dans toute ontologie OWL, il existe une superclasse, nommée *Thing*, dont toutes les autres classes définies par l'utilisateur sont implicitement des sous-classes. Ceci nous amène directement au concept d'héritage, disponible à l'aide de la propriété *subClassOf.*
- **Individu/instance** : Les individus ou instances sont les objets du domaine de discours que l'on représente dans une ontologie
- **Propriété :** Les propriétés permettent d'exprimer les faits au sujet des classes déclarées et de leurs instances. Le langage OWL distingue entre deux principaux types de propriétés :
  - Les propriétés d'objet qui relie des instances à d'autres instances.
  - Les propriétés de type de donnée qui relie des individus à des valeurs de données.

3.9.2 OWL-DL

3.9.2.1 Axiomes

En OWL-DL, les axiomes sont plus expressifs qu'en OWL-Lite. Contrairement à ce dernier langage, qui ne permet que des définitions contenant des concepts atomiques ou des restrictions limitées, OWL-DL permet d'utiliser des descriptions complexes dans une définition :

axiome::= *Class([* ***classeID*** *]* ***modalité*** **{** ***description*** **}** *)*

modalité::= *complete* | *partial*

description::= ***classeID***
| ***restriction***
|*unionOf(* **{*description*}** *)*
| *intersectionOf(* **{*description*}** *)*
| *complementOf(* ***description*** *)*
| *oneOf(* **{*individuID*}** *)*

Déclarer des classes disjointes

axiome::= DisjointClasses(**description description** {**description** })
| EquivalentClasses(**description** { **description** })
| SubClassOf(**description description**)

Définir une classe par énumération

axiome::= EnumeratedClass(**classeID** { **individuID**})

Les restrictions

restriction::=

*restriction(**propriétéIndividuID élémentRestrictionIndividu**)*

| *restriction(**propriétéLitteralID élémentRestrictionLittéral**)*

élémentRestrictionIndividu *: := allValuesFrom(**description**)*

| *someValuesFrom(**dedscription**)*

| *value(**individuID**)*

| ***cardinalité***

élémentRestrictionLittéral::= *allValuesFrom(**dataRange**)*

| *someValuesFrom(dataRange)*

| *value(**littéral**)*

| ***cardinalité***

cardinalité::= minCardinality(**entierNonNégatif**)
| maxCardinality(**entierNonNégatif**)
| cardinality(**entierNonNégatif**)

dataRange::= typeLittéralID | rdfs :Literal | oneOf({ **littéral** })

Les axiomes sur les propriétés

axiom::= *DatatypeProperty(**propriétéLittéralID***
{*super(**propriétéLittéralID**)*}
*[**Functional**]*
{*domain(**description**)*}
{*range(**dataRange**)*})
| *ObjectProperty(**propriétéIndividuID***
{*super(**propriétéIndividuID**)*}
{*inverseOf(**propriétéIndividuID**)*}
[Functional | InverseFunctional | Functional InverseFunctional | Transitive]

{*domain(* ***description*** *)}

{*range(* ***description*** *)}* *)*

axiom::=

EquivalentProperties(***propriétéLittéralID propriétéLittéralID*** {***propriétéLittéralID***}

)

| *SubPropertyOf(* ***propriétéLittéralID propriétéLittéralID*** *)*

| *EquivalentProperties(* ***propriétéIndividuID propriétéIndividuID***

{***propriétéIndividuID***} *)*

| *SubPropertyOf(* ***propriétéIndividuID propriétéIndividuID*** *)*

3.9.2.2 Faits

Les faits en OWL-DL permettent de fournir des informations sur des entités spécifiques (appelées individus). Essentiellement, on peut spécifier les classes auxquelles un individu appartient, ainsi que les valeurs des propriétés qui le concernent, qui sont des individus ou des littéraux, selon le type de propriété.

En OWL-DL, la classe d'un individu dans la ABox peut être n'importe quelle description complexe :

fait : := individu

| *SameIndividual(* ***individuID individuID*** { ***individuID*** }*)*

| *DifferentIndividuals(* ***individuID individuID*** { ***individuID*** }*)*

individu *: := Individual(* [individuID] { type(**description**) } { **valeur**})

valeur : := value(**propriétéIndividu IndividuID**)

| value(**propriétéIndividu Individu**)

| value(**propriétéLittéral Littéral**)

Exemple

En logique descriptive

$AnimalDeCompagnie \equiv AnimalDomestique \sqcap (Chien \sqcup Chat)$

$(Chien \sqcap Chat) \sqsubseteq \bot$

$AnimalDomestique \equiv Animal \sqcup \neg AnimalSauvage$

En OWL-DL

Namespace(local=<http ://www.polymtl.ca>)
Ontology(
Class(local :Animal partial)
Class(local :AnimalSauvage partial)
Class(local :AnimalDeCompagnie complete
local :AnimalDomestique
unionOf(
local :Chien
l*ocal :chat*))
DisjointClasses(local :Chien local :Chat)
Class(local :AnimalDomestique complete
local :Animal
completementOf(local :AnimalSauvage)))

3.9.3 Raisonnement avec OWL

Comme nous l'avons mentionné, le fait qu'OWL possède une base théorique formelle (la logique de description) permet l'implantation de logiciels appelés moteurs d'inférence ou raisonneurs, qui sont capables de traiter une ontologie OWL pour déduire des faits qui ne sont pas explicitement déclarés [68]. C'est-à-dire, ils peuvent trouver des informations qui sont implicitement contenues dans l'ontologie pour les rendre explicites. Ce processus s'appelle inférence ou raisonnement.

Parmi les caractéristiques importantes du raisonnement avec OWL est la monotonicité. Le fait qu'OWL soit monotone implique que tout fait présent dans une ontologie ne peut pas être supprimé[69]après l'ajout d'une nouvelle information, même si l'information ajoutée contredit celle qui existait précédemment.

Le raisonnement dans OWL applique le principe connu comme hypothèse du monde ouvert, cette caractéristique offre la possibilité d'étendre l'ontologie.

OWL-DL est capable d'exprimer certains types d'implications (Exp : héritage, propriété

transitive.), il peut être utilisé si son expressivité est suffisante. Si une expressivité plus importante est souhaitée, alors l'intégration de SWRL est nécessaire, car il permet l'usage des variables ainsi que des propriétés et des classes dans les règles.

3.9.4 Règles SWRL

Le Semantic Web Rule Langage (SWRL)[70] est un langage de règles basé sur la logique descriptive. Il est proposé par le W3C qui combine OWL-DL avec le Rule Markup Langage (RuleML) [2] pour fournir plus d'expressivité au langage OWL, cette expressivité implique la perte de la décidabilité[71].

Une règle est composée de façon générale d'une tête et d'un corps sous la forme d'une implication : $A_1 \wedge ... \wedge A_n \rightarrow B_1 \wedge ... \wedge B_n$

Où $A_1 \wedge ... \wedge A_n$ est le corps ou antécédent de la règle et $B_1 \wedge ... \wedge B_n$ est l'entête ou conséquent.

Les termes $A_1 ... A_n, B_1 ..., B_n$ sont des atomes SWRL.

Un atome A_i ou B_j est une expression de la forme $p(arg_1, ..., arg_n)$ où p est un prédicat et arg_i est un terme ou un argument de l'expression.

L'interprétation de la règle est la suivante : si les conditions spécifiées dans l'antécédent sont vérifiées, alors on peut déduire que les propositions spécifiées dans le conséquent sont vérifiées aussi.

Disscussion

Après l'étude, nous avons conclu que le langage de règles SWRL apporte une solution au manque d'expressivité du langage OWL, en permettant d'écrire des règles basées sur les concepts d'OWL-DL. Cependant, le manque de négation dans les règles de SWRL est un problème sérieux.

L'autre contrainte liée à SWRL est le recouvrement d'une propriété [5] effectivement les valeurs d'une propriété ne peuvent pas changer avec le temps, elle n'exprime pas comment les états peuvent changer au fil du temps ce qui implique le manque de la sensibilité au contexte dans les ontologies OWL.

Caractéristiques / Modèle	Relation et dépendance	Raisonnement	Négation	Point temporel
SWRL	Oui	faible	Non	Non
EC	Non	Fort	Oui	Oui

TABLE 3.7 – Comparaison entre SWRL et DEC

3.10 Synthèse

SWRL irrite des propriétés d'OWL, dont la capacité d'exprimer les relations et les dépendances entre les classes, mais le fait qu'il ne représente pas la négation des propriétés ni les points temporels le rend faible en raisonnement.

EC permet la négation, ainsi que la représentation des donnes contextuelles et leur variation dans le temps. Par conséquent amélioration de la performance du raisonnement. Cependant, il n'exprime pas relation et dépendance entres les informations du contexte.

3.11 Conclusion

L'étude des différents articles nous a permis de noter et analyser les diverses approches de modélisations et de raisonnements sur les informations contextuelles. Ainsi, ils ont contribué à l'élaboration de notre proposition. Dans le prochain chapitre, nous exposerons en détail notre contribution à la problématique.

2. Est un lanquage de règles utilisant des variables

4 Problématique et proposition

4.1 Introduction

Les caractéristiques de l'environnement ubiquitaire telles que la mobilité et l'hétérogénéité au niveau matériel et logiciel rendent la satisfaction des besoins de l'utilisateur difficile. En effet, les applications doivent être capables doffrir aux utilisateurs nomades des services adaptés par tous et à tous moments. Afin de mieux répondre aux attentes des utilisateurs, il est nécessaire de disposer dinformations contextuelles qui serviront à adapter les applications au contexte perçu.

Notre travail consiste à modéliser les informations contextuelles capturées à partir de différentes sources et raisonner sur ces informations dans le but de sélectionner le meilleur service.

4.2 Problématique

Le choix d'approche de modélisation se base sur ses capacités de représenter et raisonner sur les informations contextuelles telles que l'expressivité, possibilité de réutiliser le modèle et le partage des données. Après l'étude effectuée dans les chapitres précédents, nous avons conclu que chaque modèle a des limites et ne répond pas à toutes les exigences.

Le modèle clé - valeur est un modèle simple et facile à implémenter, mais il manque d'expression de relation. Les approches orientées modèles sont réutilisables, cependant elles ne permettent pas de raisonner sur les informations contextuelles. Les approches logiques sont très efficaces pour raisonner sur le contexte, mais elles n'expriment pas les relations qui peuvent exister entre les informations du contexte. Finalement, l'approche ontologique est expressive et représente les relations et les dépendances entre les données du contexte.

Aucune de ces approches ne peut répondre simultanément à toutes les exigences illustrées précédemment. Ce qui a incité les chercheurs à combiner entre plusieurs approches et proposer

des modèles hybrides pour plus d'expressivité et de raisonnement.

Le modèle hybride fait/ontologie qui est basé sur CML et OWL-DL n'apporte aucun avantage par rapport au raisonnement ontologique. Mais il hérite la caractéristique dinteropérable de OWL et la possibilité de réutiliser le modèle.

Le modèle hybride balisage/ontologie c'est un modèle dont le vocabulaire est pauvre, car il est restreint à la description de profil. Cependant, il est réutilisable et interopérable.

Après l'analyse des modèles cités précédemment, nous avons conclu qu'aucun d'eux ne remédie au problème de négation dans OWL/SWRL.

OWL offre un moyen de faire des descriptions sémantiques, en effet, il définit la nature des éléments qui composent le domaine de l'ontologie et les relations existantes entre ses concepts, ainsi que celles existantes entre les instances de ses classes. Il permet aussi de réutiliser, d'enrichir plus facilement l'ontologie et de publier les données décrites à travers le réseau. Cependant, les résultats expérimentaux avec une ontologie complexe (ayant plus de 500 classes) prouvent que l'exécution des tâches de raisonnement prend plus de temps par rapport au raisonnement sur une ontologie simple.

L'utilisation de SWRL permet la définition des règles d'implication, en liant des variables aux éléments dans l'ontologie. En revanche, il est limité par certaines contraintes : il ne comporte aucun opérateur de disjonction, ne soutient pas la négation et il n'exprime pas le changement des valeurs des propriétés dans le temps.

Il existe deux types de négations : négation de classes et négation de propriétés.

Pour remédier au problème de la négation d'une classe, il est possible de définir son complément à l'aide de l'opérateur *complementOf*, ce qui signifie qu'elle décrit l'ensemble de tous les individus qui ne sont pas membres de la classe. Par exemple, *owl : Class X owl : complementOf (owl : Class Y)* signifie que la classe X comprend chaque individu possible qui est introduit dans la base de connaissances, quelle que soit sa catégorie, à condition qu'elle ne soit pas membre de Y. Mais cette solution n'est pas pratique, car il est nécessaire de créer les compléments de toutes les classes ce qui augmente la complexité de l'ontologie. Par contre, il n'existe aucun moyen pour remédier au problème de la négation d'une propriété dans SWRL.

EC est fondée sur un raisonnement non monotone, ce qui suppose un monde de faits connaissables limités. Ainsi, quelques règles dans l'EC peuvent défaire d'autres et les événements peuvent déterminer les valeurs de vérité des fluents. EC inclut également les observations qui enregistrent les valeurs de vérité des fluents à des moments différents, et les historiques, qui enregistrent la séquence d'opérations.

En outre, il définit des axiomes généraux au sujet des événements et les rapports temporels,

il inclut également les axiomes qui décrivent comment les événements et les fluents agissent l'un sur l'autre.

EC est l'idée que des événements, leurs conséquences et leurs conditions peuvent tous être représentés avec le sous-ensemble de clause de Horn [1]. En effet, il supporte la négation en utilisant l'atome $\neg HoldsAt(f, t)$, par contre il n'ya aucun moyen d'exprimer son équivalent en SWRL.

4.3 Proposition

Notre travail consiste particulièrement à résoudre le problème de négation dans SWRL. Nous proposons une plateforme permettant une modélisation hybride basée sur une combinaison du modèle ontologique OWL-DL et la version DEC du formalisme logique temporel EC. Cette plateforme comprant trois étapes.

Dans la première étape, nous allons modéliser les différentes informations caractérisant notre domaine d'étude sous forme d'une ontologie. Ces informations seront enregistrées dans un fichier XML [2], formant ainsi une base de connaissances.

Dans la deuxième étape, les données seront transmises à Event Manager pour les traduire en règles de DEC compréhensibles par le raisonneur d'Event Calculus.

La troisième étape consiste à raisonner sur les évenements reçus et produire des actions de réaction à exécuter par les agents concernés, après qu'EC Reasoner a raisonné sur les événements reçus de la part d'Event Manager.

4.4 Architecture de la plateforme

L'architecture générale correspondant à la proposition est la suivante :

1. Une clause de Horn est une clause contenant au plus un littéral positif. Exemples :[$\neg A$, $\neg B$, C] est une clause de Horn

2. (eXtensible Markup Language) est un langage de balisages, les balises sont définies dans une DTD (Document Type Definition).

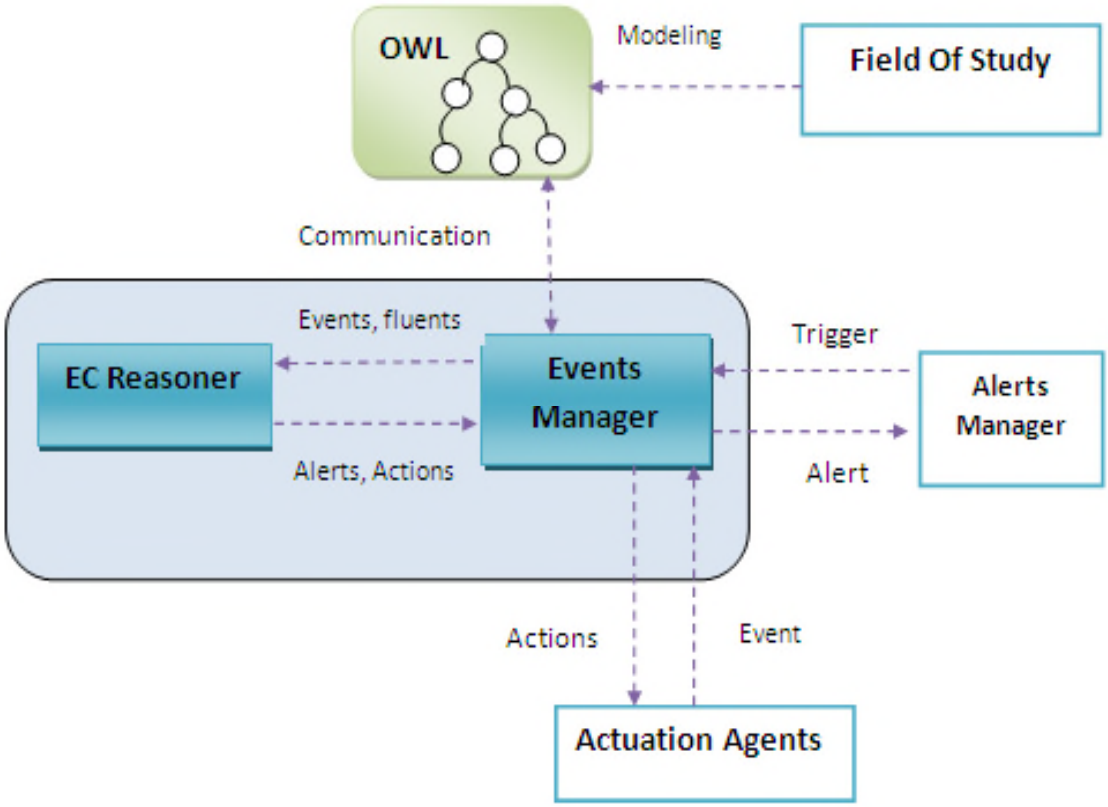


FIGURE 4.1 – Architecture générale de la proposition

- **EC Reasoner :** le Cerveau de l'infrastructure, c'est l'agent responsable des décisions des différents événements qu'il reçoit d'autres agents, l'environnement est décrit en utilisant des prédicats et fluents de Event Calculus, la prise de décision est réalisée en envoyant différentes actions à exécuter aux *Actuation Agents* ou par des déclenchements d'alertes en les envoyant aux *Alerts Manager*.
- **Events Manager :** Coeur de l'infrastructure, il joue le rôle de passerelle pour relier les différents agents. Cet agent est très important parce 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 doit passer par *Event Manager*. De plus, il traduit :
  - Les informations fournies en entrée par les différents agents sous forme d'événements et fluents de EC.
  - Les prédicats et fluent reçues de *EC Reasoner* en actions à exécuter par *Actuation Agents*.
  - Les prédicats et fluent reçues de *EC Reasoner* en alertes à déclencher par *Alerts Manager*.
  - Les informations communiquées par l'ontologie en Event, Fluent et Sort.

- **OWL :** contient la base de connaissance obtenue en modélisant le domaine d'étude.
- **Field Of Study :** domaine d'étude, 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.
- **Alerts Manager :** Cet agent est très particulier, il contrôle le système d'alerte. Lors de réception d'une alerte du Reasoner, il produit un déclenchement qui sera envoyé au Reasoner après un moment défini par ce dernier. Il peut également désactiver quelques alertes actives par ordre du Reasoner.
- **Actuation Agents :** Exécute les actions reçues du Reasoner qui provoque un événement, ce dernier sera envoyé au Reasoner.

Fonctionnement

4.5 Conclusion

Dans ce chapitre, nous avons défini notre proposition, ainsi que son architecture. Pour la valider, nous proposerons lexemple d'application de l'autosurveillance glycémique que nous détaillerons dans le prochain chapitre.

5 Validation

5.1 Introduction

Dans ce chapitre, nous développons l'exemple d'application de la proposition qui est l'autosurveillance glycémique, nous le modélisons avec l'éditeur d'ontologie protégé et l'implémenté avec Event Calculus.

5.2 Exemple d'application

5.2.1 Autosurveillance glycémique

L'autosurveillance glycémique consiste à contrôler le taux de glycémie par soi-même quotidiennement à l'aide d'un lecteur de glycémie capillaire (équipé d'un moyen de communication), au minimum le matin à jeun et 2 h après le début de chacun des trois principaux repas. C'est sur les valeurs de glycémie qu'une décision de modifications thérapeutiques peut être prise.

L'environnement du patient est équipé de différents capteurs (capteur de mouvement, de bruit...etc.) et de multiples dispositifs de communication (IPhone, Tablet, ordinateur...etc.)

L'application sensible au contexte serra installée sur l'un des dispositifs, dans notre cas, nous avons choisi IPhone.

Quand l'heure de contrôle glycémique arrive, un message écrit ou sonore annonçant l'heure de la prise serra affiché sur l'un des dispositifs, selon le contexte du patient.

À l'heure venue -> afficher un message sur le dispositif selon le contexte du patient "c'est le moment de la prise" (1)

Si non prise (2*) (non-réception du taux par le dispositif) alors rappel (3*)

Si réception du résultat (2, 3) alors - l'envoyer au service médical (4)

- Analyse du résultat (4)

Si (5) valeur normale alors afficher "normal"

Sinon

Si hypoglycémie alors afficher le traitement qui convient

Si hyperglycémie alors afficher le traitement à suivre

Si service médical reçoit la nouvelle valeur du taux concernant le patient alors

- Mise à jour (6)
- Intervention si c'est nécessaire (7)

La figure 5.1 illustre le scénario de L'autosurveillance glycémique.

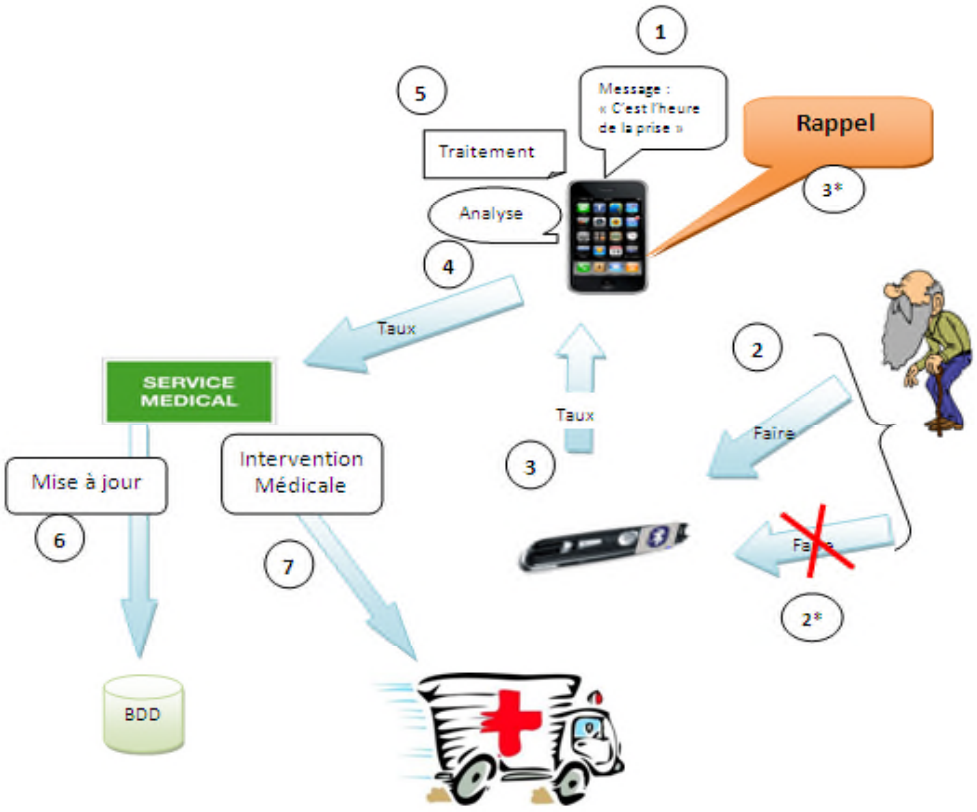


FIGURE 5.1 – Scénario Autosurveillance glycémique

5.2.2 L'ontologie de l'exemple

L'ontologie de l'exemple d'application contient les concepts de base suivants :

PATIENT : personne concerné par l'auto-surveillance glycémique.

GLYCEMIA-READER : lecteur de taux de glycémie(Stylet).

DEVICE : Dispositif utilisé par le patient (IPhone).

READING : événement qui se produit suite à la détection du sang au niveau de lecteur.

RATE-ANALYZES, SEND-MESSAGE : événements qui se produisent suite à la réception du taux.

POSTING-TRAITEMENT : événement qui se produit suite à l'analyse du taux (glycémie normale ou anormale).

TEST : situation pertinente, le fait qu'un patient effectue une prise de sang, provoque des événements.

MEDICAL-SERVICE : service médical consulté par le patient.

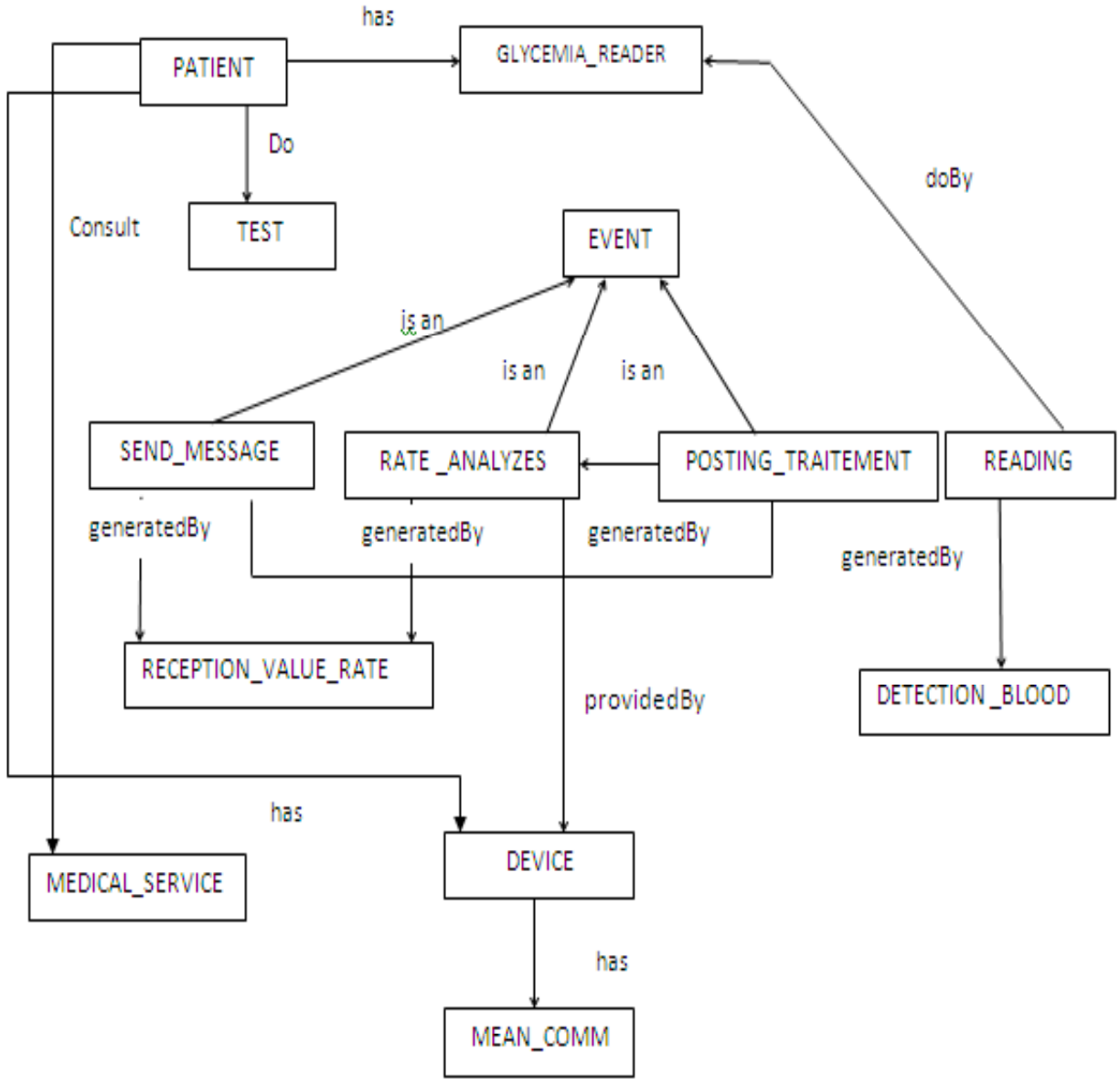


FIGURE 5.2 – L'ontologie du scenario

5.2.3 L'exemple en logique descriptive

TBox

RATE-ANALYZES, SEND-MESSAGE et POSTING-TRAITEMENT sont des sous classes de la classe EVENT

DETECTION-BLOOD $\equiv generatedByREADING$

PATIENT $\equiv \exists has(DEVICE \sqcap GLYCEMIA-READER) \sqcap consulteMEDICAL-SERVICE \sqcap faitTEST$

$GLYCEMIA - READER \equiv \exists id : String \sqcap MOY - COM.$

$POSTING-TRAITEMENT \subseteq EVENT \sqcap \exists provideByDEVICE \sqcap \exists generatedByRATE-ANALYZE$

$RATE-ANALYZES \subseteq EVENT \sqcap \exists provideByDEVICE \sqcap \exists generatedByRECEPTION-VALUE-RATE$

$SEND-MESSAGE \subseteq EVENT \sqcap \exists provideByDEVICE \sqcap \exists generatedByRECEPTION-VALUE-RATE$

ABox

$DEVICE(iphone).$

$GLYCEMIA - READER(stylet).$

$MEAN - COMM(bluetooth).$

$PATIENT(person)$

$MEDICAL - SERVICE(service - madical).$

$TEST(prise1).$

$do(person, prise1).$

$consult(person, service - medical).$

$has(person, stylet).$

5.3 Outils et langages

5.3.1 Protégé

Protégé est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Il n'est pas spécialement dédié à OWL, mais un éditeur hautement extensible, capable de manipuler des formats très divers.

C'est un outil employé par les développeurs et des experts de domaine pour développer des systèmes basés sur les connaissances (Ontologies). L'outil Protégé possède une interface

Outils/ Langage	Choix
Création/édition d'ontologies	Protégé 3.5
Création/édition de règles SWRL	SWRLTab pour Protégé
Moteur d'inférence	Pellet 1.5.2
Accès par programme aux ontologies	Protégé-OWL API

TABLE 5.1 – Outils et langages pour la création et manipulation de l'ontologie

utilisateur graphique (GUI) lui permettant de manipuler facilement tous les éléments d'une ontologie : classe, méta-classe, propriété, instance...etc. Protégé peut être utilisé dans n'importe quel domaine où les concepts peuvent être modélisés en une hiérarchie des classes.

Après avoir défini les classes de notre domaine et les relations existantes entre les classes sous Protégé, nous avons obtenu l'ontologie ci-dessous.

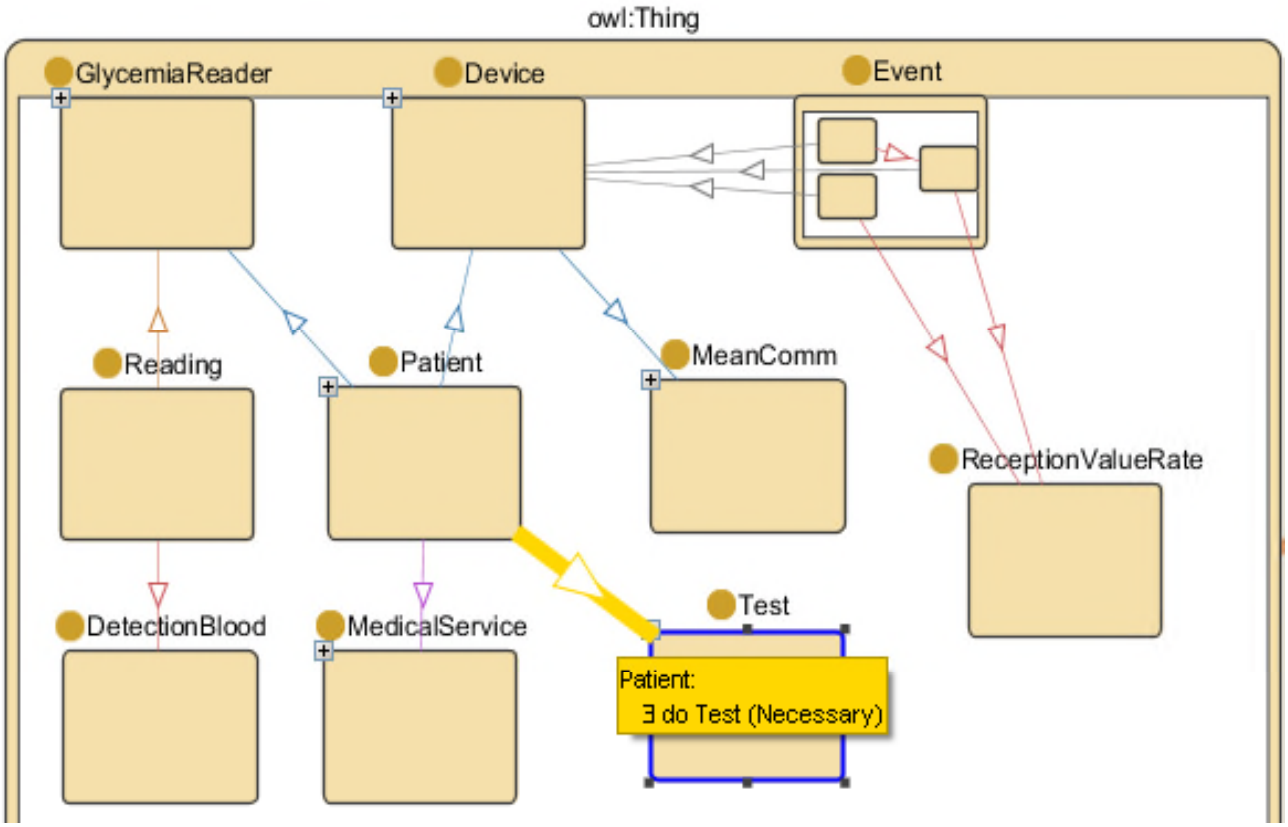


FIGURE 5.3 – L'ontologie sous protégé

5.3.2 SWRL Tab

SWRLTab est un outil pratique pour éditer des règles de SWRL et vérifier la syntaxe des règles écrites. Ces règles sont exécutées par le langage de requête SQWRL, il ne permet aucun

changement à l'information qu'elles pourraient extraire à partir de l'ontologie.

La figure 5.4 montre les étapes d'exécution des règles de SWRL sur un ensemble de données de Protégé. Les règles doivent être traduites et présentées dans le moteur de règle (1). Après, l'ontologie et la base de connaissance doivent être traduites et présentées dans le moteur de règle (2). Après le raisonnement (3), les résultats du raisonnement devraient être traduits de nouveau dans le format de Protégé (4).

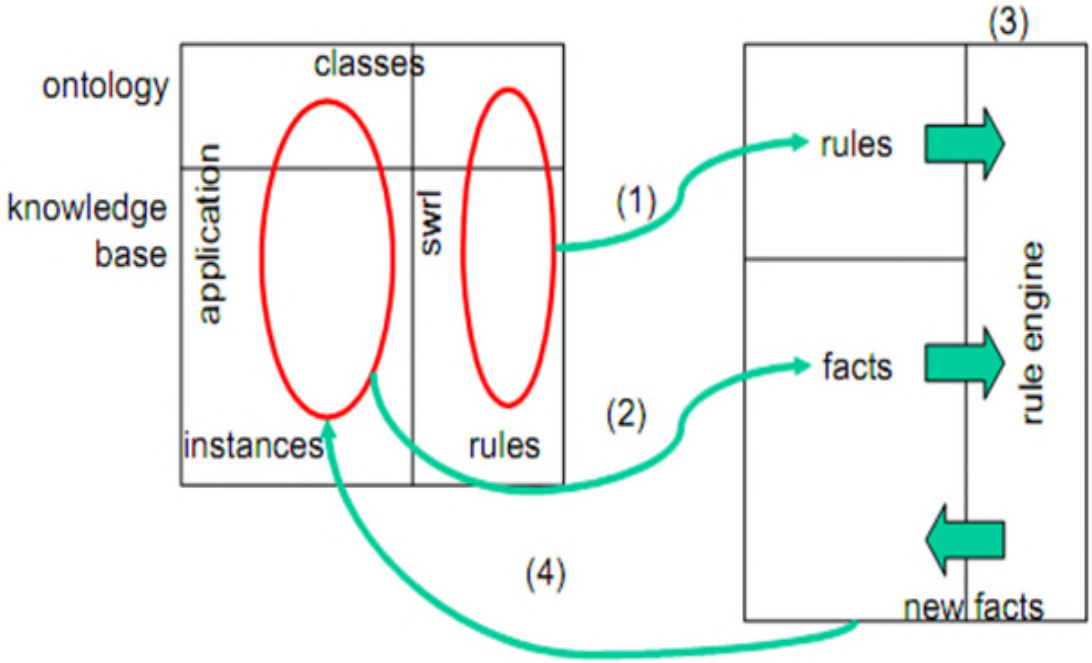


FIGURE 5.4 – Etapes d'exécution des règles SWRL sous Protégé[5]

Exemple de règle

1. **Cas où la personne a fait la prise**

   Afficher le nom des femmes qui ont fait leurs prises, ainsi que les dates de chaque prise. Women(?x)$\wedge name(?x, ?nom) \wedge Test(?y) \wedge date(?y, ?d) \wedge rote(?y, ?taux) \wedge do(?x, ?y) \longrightarrow sqwrl : select(?nom, ?d, ?taux)$

Le résultat de l'exécution de la règle précédente est affiché sur la figure 5.5 :

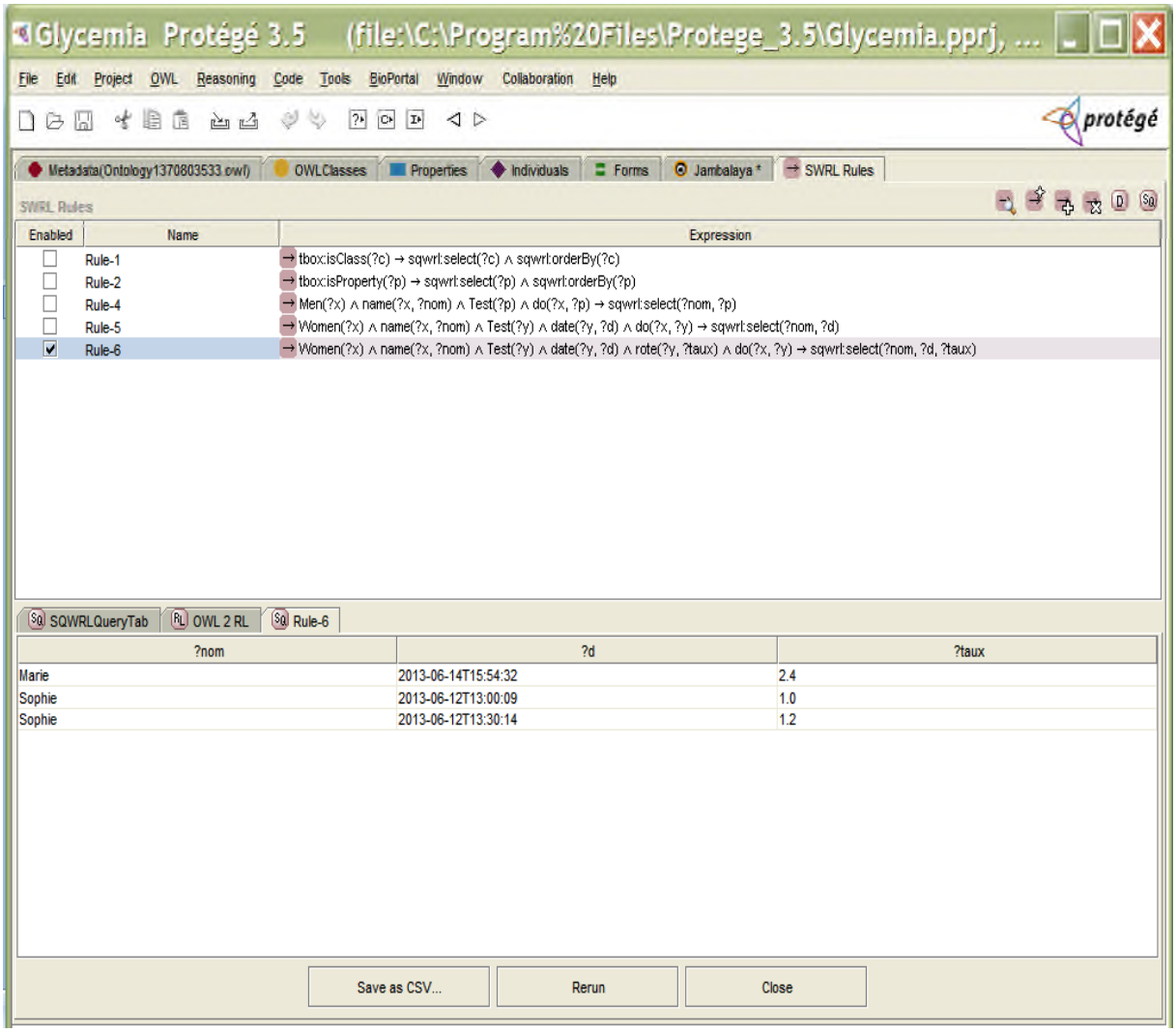


FIGURE 5.5 – Execution d'une règle SWRL sous protégé

2. **Cas où la personne n'a pas fait la prise**

   Il est impossible d'exprimer la négation de la propriété "do" en utilisant une règle SWRL.En effet, il n'est pas permis d'insérer un symbole de négation.

   La figure 5.6 prouve qu'on ne peut pas présenter la négation de la propriété " do "

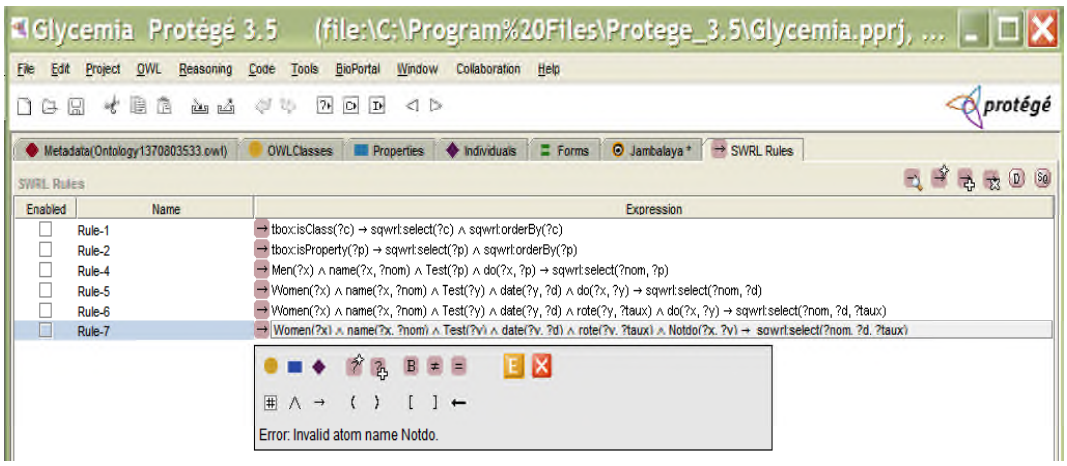


FIGURE 5.6 – Exécution d'une règle SWRL contenant la négation de la propriété "do"

5.3.3 Pellet

Le raisonneur Pellet a été construit pour raisonner sur des ontologies d'OWL-DL[72], c'est le raisonneur par défaut dans Protégé 3.x.

5.3.4 Protégé-OWL API pour le traitement des ontologies OWL

Protégé fournit une interface de programmation d'application (API) écrite en JAVA. 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"* fourni avec Protégé. Cette classe possède une méthode *getKnowledgeBase()* permettant d'accéder au contenu de la base de connaissances.

5.4 Implémentation du scénario d'autosurveillance glycémique avec DEC

5.4.1 Cas où le patient fait sa prise

Les prédicats correspondants sont :

Domain Aximozation

$Happens(TimeActivity(?MobilDevice), ?t) => Happens(PlayActivityMessage(?Patient,$

$?MobilDevice), ?t).$

$Happens(PlayActivityMessage(?Patient, ?MobilDevice), ?t) \wedge HoldsAt(PriseDone($
$?Patient), ?t) \wedge HoldsAt(RecevedValue(?GlycemiaReader, ?MobilDevice, ?Value), ?t) =>$
$Happens(SendInformation(?MobilDevice, ?MedicalService), ?t).$

$Happens(PlayActivityMessage(?Patient, ?MobilDevice), ?t) \wedge HoldsAt(PriseDone($
$?Patient), ?t) \wedge HoldsAt(RecevedValue(?GlycemiaReader, ?MobilDevice, ?Value), ?t)$
$=> Happens(AnalyzesValue(?Value, ?MobilDevice), ?t).$

$Happens(AnalyzesValue(?Value, ?MobilDevice), ?t)) => Happens(PostingTraitement($
$?Patient, ?MobilDevice), ?t).$

Partie narrative(informations reçus)
$Happens(TimeActivity(IPhone), 1).$
$HoldsAt(PriseDone(person), 1).$
$HoldsAt(RecevedValue(stylet, IPhone, taux), 1).$
Le résultat du raisonnement avec EC est présenté dans la figure ci-dessous :

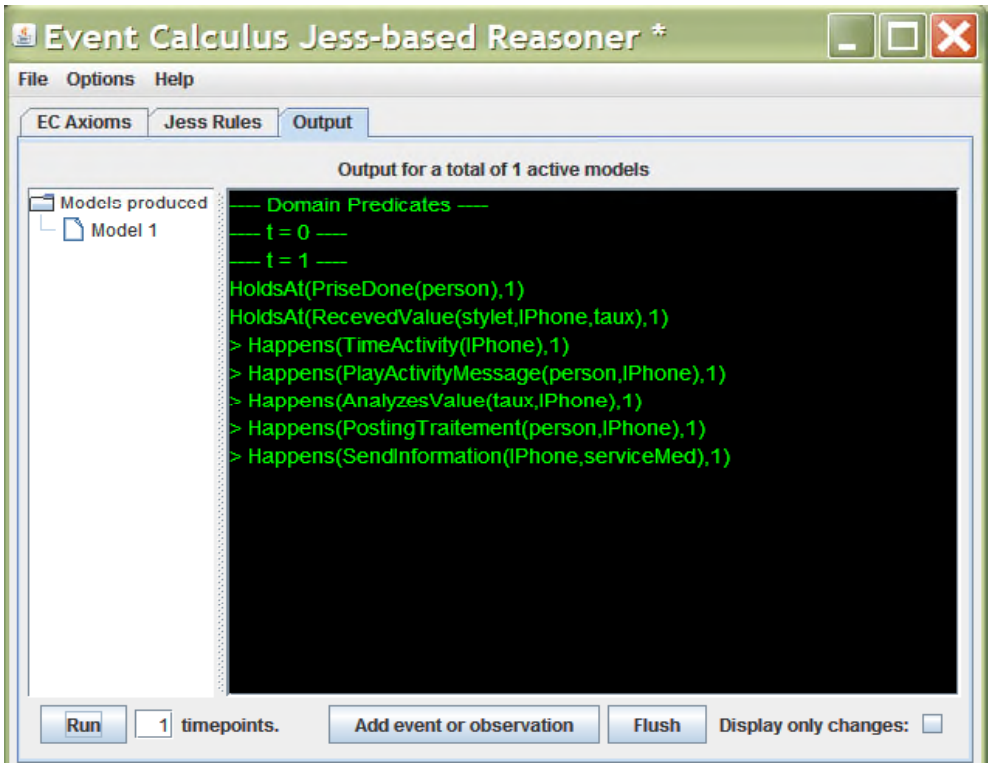


FIGURE 5.7 – Résultat d'execution des prédicats dans le cas de prise sous EC

5.4.2 Cas où le patient ne fait pas sa prise

Les prédicats associés sont :

Domain Aximozation

$Happens(TimeActivity(?MobilDevice), ?t) => Happens(PlayActivityMessage(?Patient, ?MobilDevice), ?t)$.

$Happens(PlayActivityMessage(?Patient, ?MobilDevice), ?t) \wedge \sim HoldsAt(PriseDone(?Patient), ?t+3) => Happens(PlayRemainingMessage(?Patient, ?MobilDevice), ?t+3)$.

Partie narrative

$Happens(TimeActivity(IPhone), 1)$.

$\sim HoldsAt(PriseDone(person), 1)$.

Le résultat du raisonnement avec EC est présenté dans la figure ci-dessous :

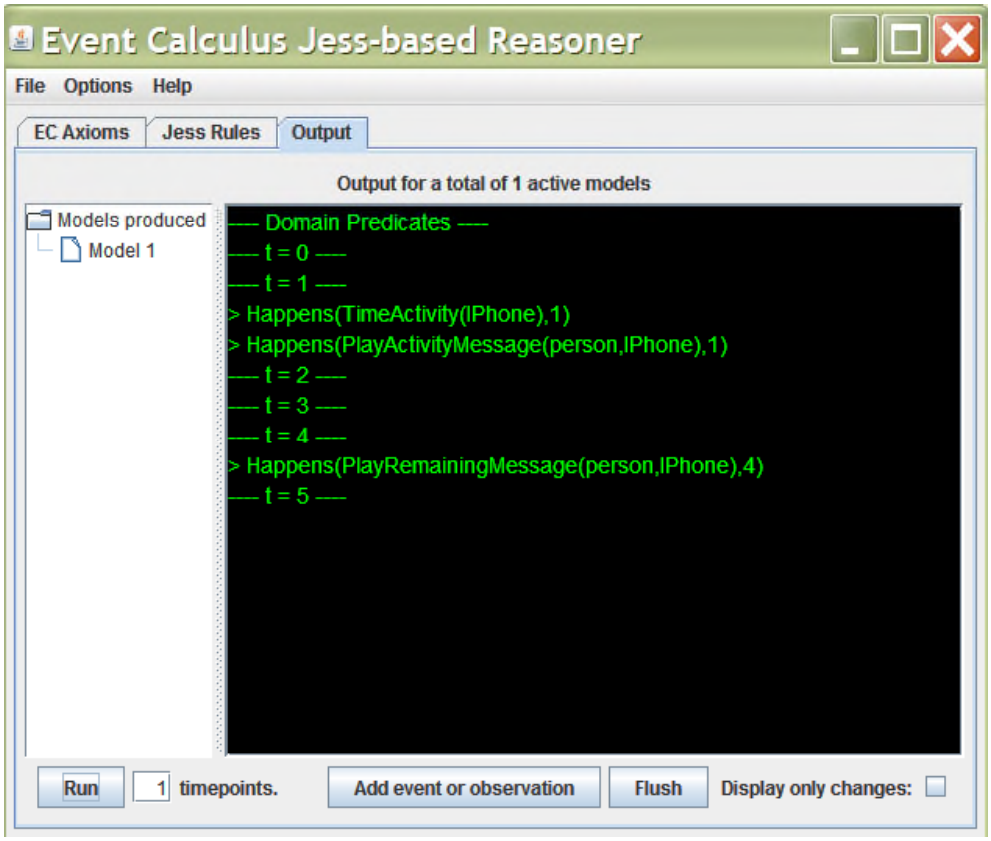


FIGURE 5.8 – Résultat d'execution des prédicats dans le cas non prise sous EC

5.5 Synthèse

Les caractéristiques de notre proposition sont les suivantes :

- L'architecture proposée permet de modéliser tous types de contexte collectés à partir des différents capteurs en utilisant le langage ontologique OWL-DL, donc elle assure l'interopérabilité et le partage des données.
- Elle exprime les relations et les dépendances qui existent entre les informations contextuelles et nous offre la possibilité d'enrichir et étendre le modèle.
- De plus, le raisonnement avec DEC permet une forte expressivité, car DEC exprime le déroulement des événements en fonction du temps.
- Elle apporte une solution au problème de négation dans les ontologies (OWL-DL).
- Event-Manager et EC-Reasoner sont des programmes qui peuvent être appliqué non seulement sur l'exemple proposé, mais aussi, il est réutilisable pour d'autres applications.

Néanmoins, la solution proposée est complexe, car il est nécessaire de disposer d'un programme (Events-Manager) qui permet de présenter les concepts et les relations de OWL en sort et prédicats. De plus, il faut maintenir une communication entre Event-Manager et OWL.

5.6 Conclusion

Dans ce chapitre, nous avons pu montrer la possibilité d'exprimer la négation avec Event Calculus, et pour ce faire : l'étape de modélisation prend en entrée les informations collectées par les différentes sources d'informations, qu'elles soient matérielles ou logicielles, afin d'établir les relations qui existent entre ces informations, cette tache revient à OWL. Puis, vient l'étape d'interprétation et présentation des informations sous forme de prédicats et sorts, qui sont effectués par Event-Manager. Pour finir, l'étape de raisonnement qui est effectué par EC-Reasoner.

Conclusion générale et perspective

Les technologies mobiles et leur large adoption par les nouvelles générations, sont appelées à transformer notre mode de vie par l'invention de nouveaux services adaptables au contexte utilisateur, facilement accessible en mobilité, et très pratiques pour une utilisation courante dans la vie quotidienne moderne.

La diversité des informations de contexte et leur utilisation dans divers domaines engendrent différentes façons de les modéliser. En effet, la modélisation du contexte est la première étape dans le processus de création d'applications sensibles au contexte. Cette modélisation permet à l'application de faciliter l'interaction avec le contexte en fournissant une description abstraite des observables. C'est dans le cadre de cette problématique que notre travail de recherche s'est effectué.

Après les différentes études des travaux menés sur le langage ontologique OWL/SWRL, nous avons pu déterminer certaines limites dans la description des informations en utilisant SWRL, mais il comporte aussi l'avantage de représenter les relations et les dépendances entre elles. Nous avons approfondi notre étude sur le problème de négation que SWRL ne peut pas représenter.

D'autres études ont démontré que l'approche de modélisation logique Event Calculus est forte en raisonnement, il permet d'exprimer la négation. Cependant, il ne décrit pas les relations et les dépendances entre les données contextuelles.

Dans le but de remédier au problème de la négation dans les ontologies tout en conservant le raisonnement et les interrelations, nous avons proposé une architecture hybride qui se base sur OWL et EC en utilisant Event-Manager. De ce fait, nous avons illustré notre proposition par l'exemple de l'autosurveillance glycémique.

Cependant, notre travail n'a pas été finalisé, car le cur de l'architecture (Event-Manager) est à développer. De plus, nous n'avons pas pu établir la communication entre OWL et Event-

Manager.

À travers cette contribution, nous avons pu découvrir plusieurs domaines utilisant l'informatique ubiquitaire, comprendre le fonctionnement des systèmes ubiquitaires et leur nécessité dans la vie courante. De plus, ce projet nous a initié à la recherche et au découvert de nouveaux outils (Protégé, Event Calculus).

Pour terminer, nous tenons à souligner que nous n'avons nullement la prétention d'avoir présenté un travail parfait, car aucun travail scientifique ne peut l'être, ainsi nous laissons le soin à tous ceux qui le liront et qui sont du domaine de nous parvenir leur remarques et suggestions pour l'enrichir et l'amélioré.

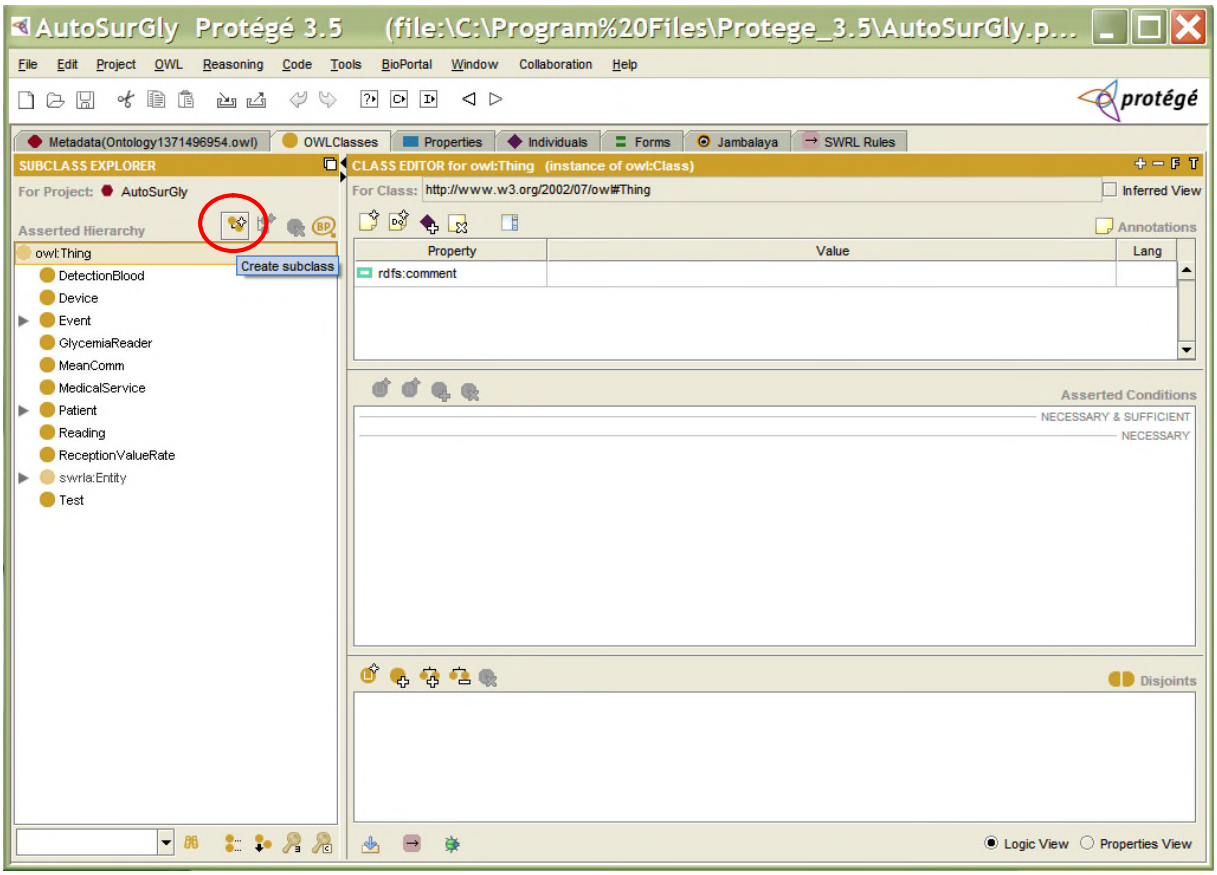
Annexe

Annexe A

Etapes de création de l'ontologie

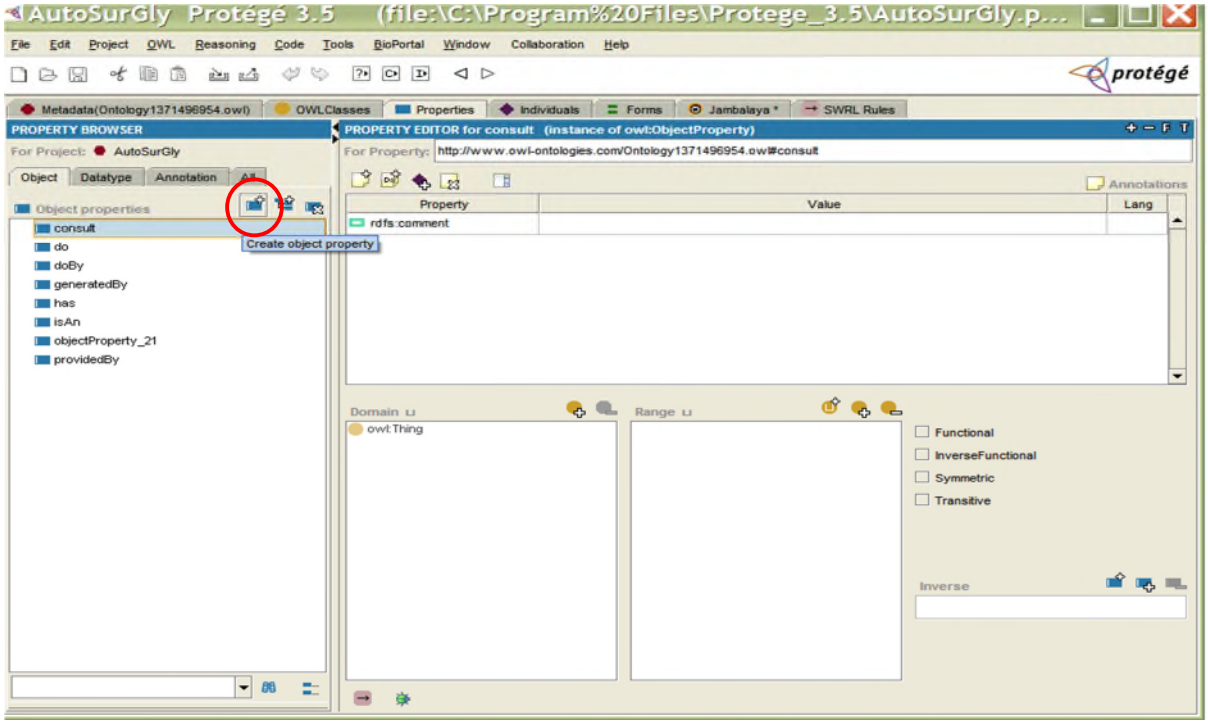
Définition des classes

Appuyer sur l'onglet *OWLClasses* et cliquer sur le mot *Thing* dans la vue *Asserted hierarchy* qui est la racine de toutes les classes que nous allons créer. *En appuyant sur le premier* bouton en haut à droite de cette vue, un dialogue apparait qui permet de créer une classe.



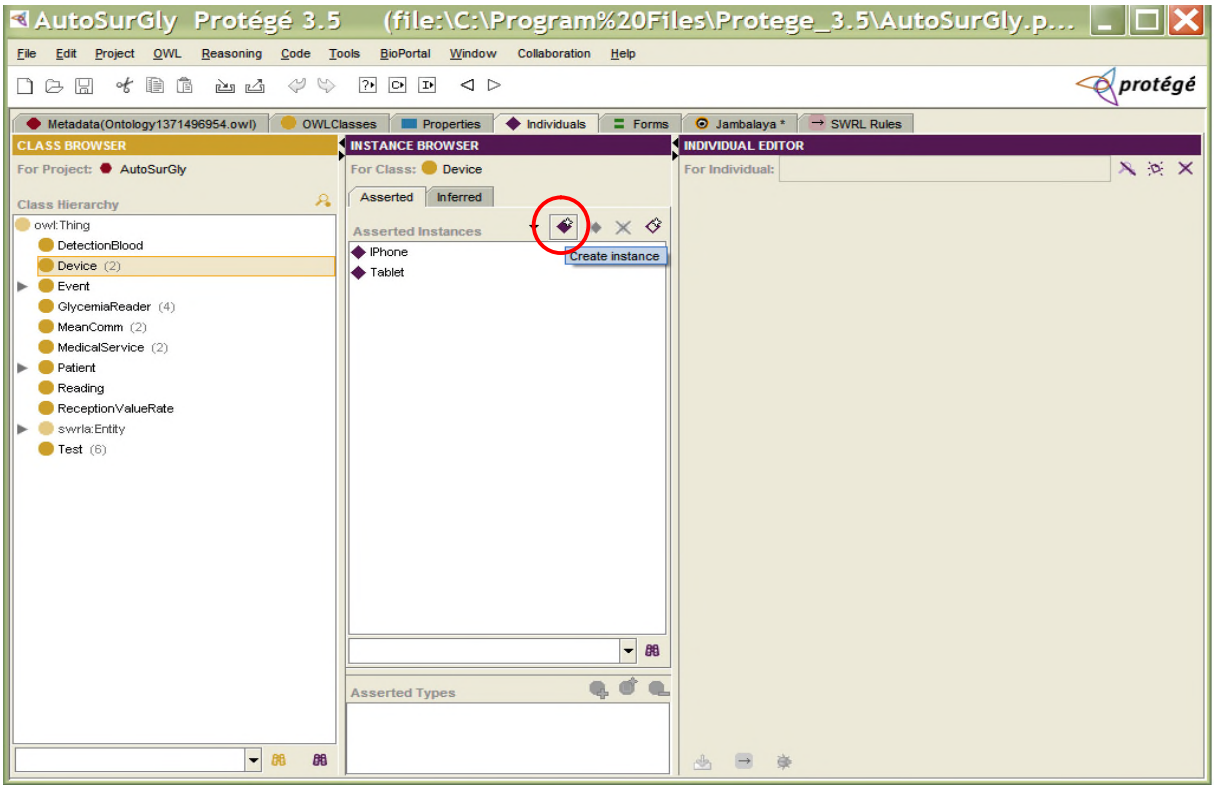
Définition des propriétés

Appuyer sur l'onglet *Properties* et, en utilisant le bouton en haut à droite dans la vue *Object properties* créer les propriétés dont vous avez besoin. Votre fenêtre ressemblera à ceci :



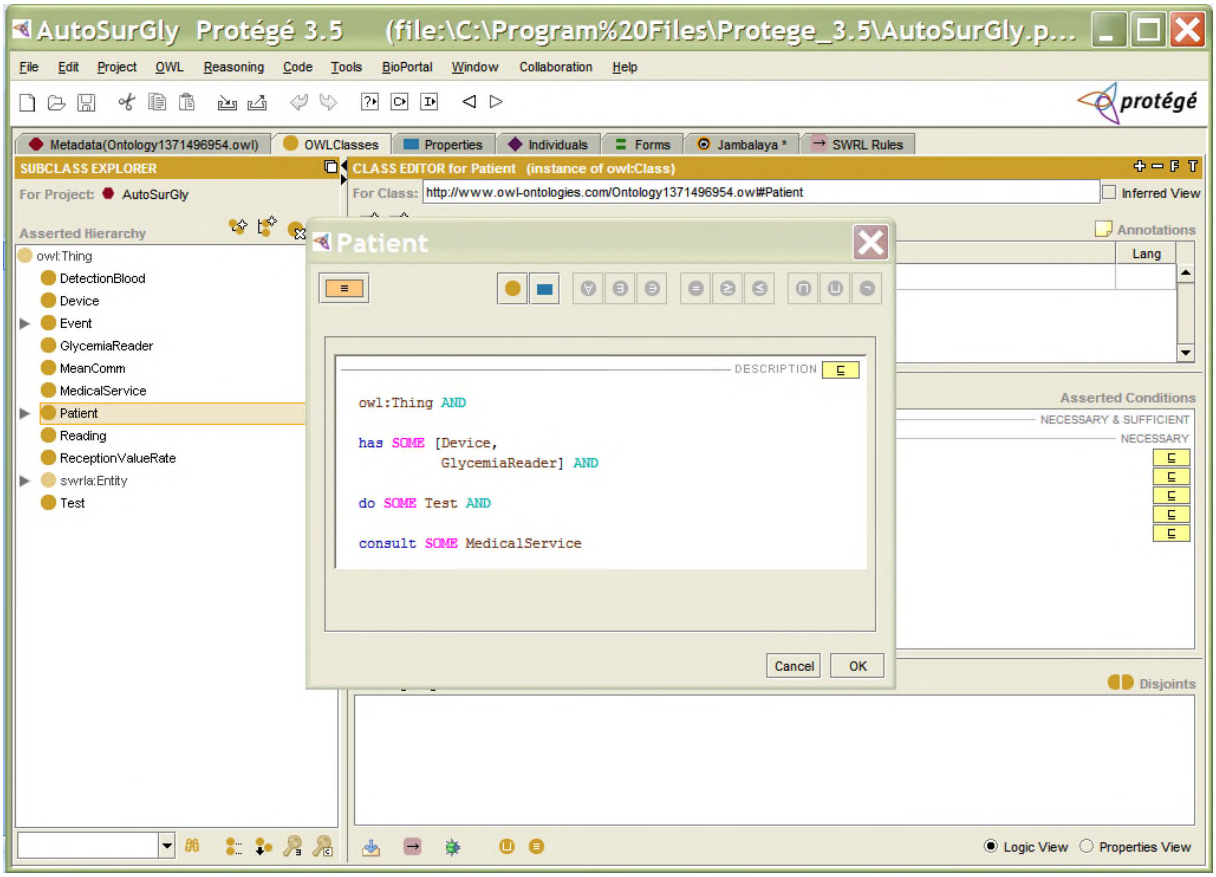
Ajout d'individus

Pour ajouter des instances de classes appuyer dans l'onglet *Individuals*, ensuite sélectionnant une classe dans la vue de gauche et ajouter un individu comme membre. Comme la figure si dessous.



Définir les relations entre les classes

Appuyer sur l'onglet Classes pour compléter les descriptions de classes. Choisissez une classe la vue *Asserted hierarchy* et cliquer sur le bouton droit de la souris, un menu apparaitra. Choisisz *Object restriction creator* pour faire apparaître une fenêtre permettant de définir une expression comme dans la figure suivante :



Annexe B

Le code RDF entier de l'ontologie sous 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 abox "http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#" >
   <!ENTITY tbox "http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#" >
   <!ENTITY sqwrl "http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1371496954.owl#"
    xml:base="http://www.owl-ontologies.com/Ontology1371496954.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:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:tbox="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
    xmlns:abox="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#">
   <owl:Ontology rdf:about="">
      <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
      <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
      <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl"/>
      <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl"/>
   </owl:Ontology>
   <swrl:Variable rdf:ID="c"/>
   <owl:DatatypeProperty rdf:ID="adresse">
```

```
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MedicalService"/>
          <owl:Class rdf:about="#Patient"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="age">
    <rdfs:domain rdf:resource="#Patient"/>
    <rdfs:range rdf:resource="&xsd;int"/>
  </owl:DatatypeProperty>
  <MeanComm rdf:ID="bleutooth"/>
  <owl:ObjectProperty rdf:ID="consult"/>
  <owl:DatatypeProperty rdf:ID="date">
    <rdfs:domain rdf:resource="#Test"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="DetectionBlood"/>
  <owl:Class rdf:ID="Device"/>
  <owl:ObjectProperty rdf:ID="do"/>
  <owl:ObjectProperty rdf:ID="doBy"/>
  <owl:Class rdf:ID="Event"/>
  <Women rdf:ID="F1">
    <adresse rdf:datatype="&xsd;string">345REZ</adresse>
    <age rdf:datatype="&xsd;int">28</age>
    <consult rdf:resource="#ServiceMed1"/>
    <do rdf:resource="#Test4"/>
    <do rdf:resource="#Test5"/>
    <has rdf:resource="#L4"/>
    <has rdf:resource="#Tablet"/>
    <name rdf:datatype="&xsd;string">Sophie</name>
    <phoneNum rdf:datatype="&xsd;string">4357789</phoneNum>
  </Women>
  <Women rdf:ID="F2"/>
  <Women rdf:ID="F3">
    <adresse rdf:datatype="&xsd;string">rue234</adresse>
    <age rdf:datatype="&xsd;int">50</age>
    <consult rdf:resource="#Service2"/>
    <do rdf:resource="#Test6"/>
    <has rdf:resource="#L3"/>
    <has rdf:resource="#Tablet"/>
    <name rdf:datatype="&xsd;string">Marie</name>
```

```
    <phoneNum rdf:datatype="&xsd;string">8665545</phoneNum>
</Women>
<owl:ObjectProperty rdf:ID="generatedBy"/>
<owl:Class rdf:ID="GlycemiaReader">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#has"/>
            <owl:someValuesFrom rdf:resource="#MeanComm"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="has"/>
<MeanComm rdf:ID="infrarouge"/>
<Device rdf:ID="IPhone"/>
<owl:ObjectProperty rdf:ID="isAn"/>
<GlycemiaReader rdf:ID="L1">
    <has rdf:resource="#bleutooth"/>
</GlycemiaReader>
<GlycemiaReader rdf:ID="L2">
    <has rdf:resource="#bleutooth"/>
</GlycemiaReader>
<GlycemiaReader rdf:ID="L3">
    <has rdf:resource="#infrarouge"/>
</GlycemiaReader>
<GlycemiaReader rdf:ID="L4">
    <has rdf:resource="#bleutooth"/>
</GlycemiaReader>
<owl:Class rdf:ID="MeanComm"/>
<owl:Class rdf:ID="MedicalService"/>
<owl:Class rdf:ID="Men">
    <rdfs:subClassOf rdf:resource="#Patient"/>
</owl:Class>
<Men rdf:ID="Men2">
    <adresse rdf:datatype="&xsd;string">rue num 2</adresse>
    <age rdf:datatype="&xsd;int">50</age>
    <consult rdf:resource="#ServiceMed1"/>
    <do rdf:resource="#Test3"/>
    <has rdf:resource="#IPhone"/>
    <has rdf:resource="#L2"/>
    <name rdf:datatype="&xsd;string">David</name>
    <phoneNum rdf:datatype="&xsd;string">12345</phoneNum>
</Men>
<Men rdf:ID="Men_1">
    <adresse rdf:datatype="&xsd;string">rue num 1</adresse>
```

```
    <age rdf:datatype="&xsd;int">44</age>
    <consult rdf:resource="#Service2"/>
    <do rdf:resource="#Test1"/>
    <do rdf:resource="#Test2"/>
    <has rdf:resource="#L1"/>
    <has rdf:resource="#Tablet"/>
    <name rdf:datatype="&xsd;string">Pierre</name>
    <phoneNum rdf:datatype="&xsd;string">324567</phoneNum>
</Men>
<owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#Patient"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="objectProperty_21"/>
<owl:Class rdf:ID="Patient">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#has"/>
            <owl:someValuesFrom rdf:resource="#Device"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#has"/>
            <owl:someValuesFrom rdf:resource="#GlycemiaReader"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#do"/>
            <owl:someValuesFrom rdf:resource="#Test"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#consult"/>
            <owl:someValuesFrom rdf:resource="#MedicalService"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="phoneNum">
    <rdfs:domain rdf:resource="#Patient"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

```
<owl:Class rdf:ID="PostingTraitment">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#providedBy"/>
      <owl:someValuesFrom rdf:resource="#Device"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#generatedBy"/>
      <owl:someValuesFrom rdf:resource="#RateAnalyzes"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Event"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="providedBy"/>
<owl:Class rdf:ID="RateAnalyzes">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#providedBy"/>
      <owl:someValuesFrom rdf:resource="#Device"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#generatedBy"/>
      <owl:someValuesFrom rdf:resource="#ReceptionValueRate"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Event"/>
</owl:Class>
<owl:Class rdf:ID="Reading">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#doBy"/>
      <owl:someValuesFrom rdf:resource="#GlycemiaReader"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#generatedBy"/>
      <owl:someValuesFrom rdf:resource="#DetectionBlood"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
</owl:Class>
<owl:Class rdf:ID="ReceptionValueRate"/>
<owl:DatatypeProperty rdf:ID="rote">
  <rdfs:domain rdf:resource="#Test"/>
  <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="SendMessage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#providedBy"/>
      <owl:someValuesFrom rdf:resource="#Device"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#generatedBy"/>
      <owl:someValuesFrom rdf:resource="#ReceptionValueRate"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Event"/>
</owl:Class>
<MedicalService rdf:ID="Service2">
  <adresse rdf:datatype="&xsd;string">rue num 13</adresse>
</MedicalService>
<MedicalService rdf:ID="ServiceMed1">
  <adresse rdf:datatype="&xsd;string">rue num 12</adresse>
</MedicalService>
<Device rdf:ID="Tablet"/>
<owl:Class rdf:ID="Test"/>
<Test rdf:ID="Test1">
  <date rdf:datatype="&xsd;dateTime"
    >2013-06-24T08:00:51</date>
  <doBy rdf:resource="#Men_1"/>
  <rote rdf:datatype="&xsd;float">2.0</rote>
</Test>
<Test rdf:ID="Test2">
  <date rdf:datatype="&xsd;dateTime"
    >2013-06-24T10:10:39</date>
  <doBy rdf:resource="#Men_1"/>
  <rote rdf:datatype="&xsd;float">2.5</rote>
</Test>
<Test rdf:ID="Test3">
  <date rdf:datatype="&xsd;dateTime"
    >2013-06-24T13:00:16</date>
  <doBy rdf:resource="#Men2"/>
```

```
    <rote rdf:datatype="&xsd;float">0.9</rote>
  </Test>
  <Test rdf:ID="Test4">
    <date rdf:datatype="&xsd;dateTime"
      >2013-06-24T11:17:01</date>
    <rote rdf:datatype="&xsd;float">1.0</rote>
  </Test>
  <Test rdf:ID="Test5">
    <date rdf:datatype="&xsd;dateTime"
      >2013-06-24T13:29:34</date>
    <rote rdf:datatype="&xsd;float">1.2</rote>
  </Test>
  <Test rdf:ID="Test6">
    <date rdf:datatype="&xsd;dateTime"
      >2013-06-23T07:30:10</date>
    <rote rdf:datatype="&xsd;float">3.0</rote>
  </Test>
  <owl:Class rdf:ID="Women">
    <rdfs:subClassOf rdf:resource="#Patient"/>
  </owl:Class>
</rdf:RDF>
```

Annexe C

Code entier

```
sort: Patient(person).
sort: MobilDevice(IPhone).
sort: GlycemiaReader(stylet).
sort: MedicalService(serviceMed).
sort: Value(Taux).
event: TimeActivity(MobilDevice).
event: PlayActivityMessage(Patient,MobilDevice).
event: PlayRemainingMessage(Patient,MobilDevice).
event: SendInformation(MobilDevice,MedicalService).
event: AnalyzesValue(Value,MobilDevice).
event: PostingTraitement(Patient,MobilDevice).
fluent: PriseDone(Patient).
fluent: RecevedValue(GlycemiaReader,MobilDevice,Value).
```

Cas où le patient fait sa prise

```
Happens(TimeActivity(?MobilDevice),?t)=>
Happens(PlayActivityMessage(?Patient,?MobilDevice), ?t).

Happens(PlayActivityMessage(?Patient,?MobilDevice), ?t) ^ HoldsAt(PriseDone(?Patient),?t)^
HoldsAt(RecevedValue(?GlycemiaReader,?MobilDevice,?Value), ?t) =>
Happens(SendInformation(?MobilDevice,?MedicalService), ?t).

Happens(PlayActivityMessage(?Patient,?MobilDevice), ?t) ^ HoldsAt(PriseDone(?Patient),
?t)^HoldsAt(RecevedValue(?GlycemiaReader,?MobilDevice,?Value), ?t) => Happens(
AnalyzesValue(?Value,?MobilDevice), ?t).

Happens(AnalyzesValue(?Value, ?MobilDevice), ?t))=>
Happens(PostingTraitement(?Patient,?MobilDevice), ?t).


%%%%%%%%%%%%%Narrative %%%%%%%%%%%%%%%%%%
Happens (TimeActivity(IPhone), 1).
HoldsAt (PriseDone(person), 1).
HoldsAt(RecevedValue(stylet, IPhone, taux), 1).
```

Cas où le patient n'a pas fait sa prise

```
Happens(TimeActivity(?MobilDevice), ?t)=>
Happens(PlayActivityMessage(?Patient,?MobilDevice), ?t).
```

Happens(PlayActivityMessage(?Patient,?MobilDevice), ?t) ^ ~HoldsAt(PriseDone(?Patient), ?t+3) => Happens(PlayRemainingMessage(?Patient,?MobilDevice),?t+3).

%%%%%%%%%%%%%Narrative %%%%%%%%%%%%%%%%%%
Happens (TimeActivity(IPhone), 1).

Bibliographie

[1] I.Rebai. Informatique ubiquitaire et pervasive. Bretagne, février 2012.

[2] M.Baldauf and S.Dustdar. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing, forthcoming*, Vol.2(N°4) :263–277, 2007.

[3] N.Belhanafi Behlouli. *Ajout de mécanismes de réactivité au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades.* PhD thesis, Université dÉvry Val dEssonne, 2006.

[4] G.Sancho. *Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires. Contribution à l'interopérabilité par la sémantique.* PhD thesis, Université Toulouse 1 Capitole (UT1 Capitole), décembre 2010.

[5] D.Plas and al. *Manipulating context information with SWRL.* Ericsson Telecommunicatie B.V., March 7 2006.

[6] M.Weiser. The computer for the 21st century. *Scientific American*, 265(3) :94–104, septembre 1991.

[7] C.Linnhoff-Popien and Strang. A context modeling survey. *In UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management*, pages 34–41, 2004. Nottingham.

[8] Z.Drey. *Vers une méthodologie dédiée à l'orchestration d'entités communicantes.* Thèse de doctorat en informatique, Ecole doctorale EDMI, Bordeaux.

[9] M.Miraoui. *Architecture logicielle pour l'informatique diffuse : modélisation du contexte et adaptation dynamique des services.* Thèse de doctorat en informatique, école de technologie supérieure université du Québec, Montréal, Juillet 2009.

[10] S.Ghanem and Z.bouanani. Gestion de contexte et découverte de service sensible au contexte dans un environnement ubiquitaire. Master's thesis, Université A.Mira, Béjaia, 2012.

[11] B.Schilit and M.Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5) :22-32, 1994.

[12] P.Brown, J.Bovey, and X.Chen. Context-aware applications : from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5) :58–64, October 1997.

[13] N.Ryan, J.Pascoe, and D.Mors. *nhanced Reality Fieldwork : the Context aware Archaeological Assistant. In V. Gaffney, M. van Leusen et S. Exxon.* British Archaeological Reports, Oxford, tempus reparatum edition, october 1998.

[14] J.Pascoe. Adding generic contextual capabilities to wearable computers. *In Wearable Computers,Digest of Papers. Second International Symposium on*, pages 92–99, October 1998.

[15] G.Abowd and al. Towards a better understanding of context and context-awareness. *In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, 1999.

[16] T.Winograd. Architectures for context. human-computer interaction. *16 :401-419*, 2001.

[17] N.Schilit, N.Adams, and R.Want. Context-aware computing applications. *In Workshop on Mobile Computing Systems and Applications.Sta Cruz, EtatsUnis*, December 1994.

[18] D.Salber, A.Dey, and G.Abowd. Ubiquitous computing : Defining an hci research agenda for an emerging interaction paradigm. Technical report, In Georgia Tech GVU technical report, Janvier 1998.

[19] A.Dey, G.Abowd, and D.Salber. A conceptual framework and toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer Interaction*, 16(2-4 (special issue on context-aware computing)) :97–166, 2001.

[20] A.Asthana, M.Cravatts, and P.Krzyzanowski. An indoor wireless system for personalized shopping assistance. *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 69–74, December 1994. Santa Cruz, California.

[21] G.Abowd and al. A mobile context-aware tour guide. *Wireless Networks*, 3(5) :421–433, 1997.

[22] A.Dey and al. The conference assistant combining context-awareness with wearable computing. *In Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99)*, pages 21–28, October 1999.

[23] A.Schmidt and al. Advanced interaction in context. *In Proceedings of the First International Symposium on Handheld and Ubiquitous Computing,HUC'99*, pages 89–101, September 1999.

[24] R.Want and al. The active badge location system. *ACM Transactions on Information Systems*, 10(1) :91–102, January 1992.

[25] G.Chen and D.Kotz. A survey of context-aware mobile computing research. Technical report, Technical report TR2000-381, Dept. of Computer Science, Dartmouth college, 2000.

[26] J.Indulska and al. Experiences in using cc/pp in context-aware systems. *In Proceedings of the 4th international Conference on Mobile Data Management,Melbourne, Australia*, Vol. 2574 :247–261, January 21 - 24 2003.

[27] A.Dey, G.Abowd, and A.Wood. A framework for providing self- integrating context-aware services. *Knowledge Based Systems*, Vol.11(N°1) :3–13, 1998.

[28] E.Kiciman and A.Fox. Using dynamic mediation to integrate cots entities in a ubiquitous computing environment. *In Proceedings of the 2nd International Symposium on Handheld and Ubiquitious Computing (HUC2K). Heidelberg, Germany : Springer Verlag*, 2000.

[29] M.Bauer and al. A collaborative wearable system with remote sensing. *Proceedings of the 2nd International Symposium on Wearable Computers (ISWC98), CA : IEEE,Los Alamitos*, pages 10–17, 1998.

[30] T.Rodden and al. Exploiting context in hci design for mobile systems. *In Proceedings of the Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[31] H.Chen and al. An ontology for context-aware pervasive computing environments. *In Workshop on Ontologies and Distributed Systems (IJCAI 2003), Mexico*, August 2003.

[32] C.Bittini and al. A survey of context modelling and reasoning techniques. 27 March 2008.

[33] J.Bauer. dentification and modeling of context for different information in air traffic. Master's thesis, Université d'eléctronique et d'informatique de Berlin, Mars 2003.

[34] Q. Sheng and Benatallah. *ContextUML : A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services.* IEEE Computer Society, Sydney, Australia, July 11-13 2005.

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

[36] K.Henricksen, J.Indulska, and A.Rakotonirainy. Modeling context information in pervasivecomputing systems. *In the First International Conference on Pervasive Computing*, pages 167–180, 2002.

[37] K.Henricksen and T.McFadden. Modelling context information with orm. volume 3762 of Lecture Notes in Computer Science :626–635, 2005.

[38] K.Henricksen and J.Indulska. Modelling and using imperfect context information. *In PerCom Workshops*, pages 33–37, 2004.

[39] J.McCarthy. *Notes on Formalizing Contexts*. Proceedings of the Fifth National Conference on Artificial Intelligence, Los Altos, California, morgan kaufmann edition, 1993. 555-560.

[40] J.McCarthy and S.Buvac. Formalizing context (expanded notes). Technical report, Technical report, Stanford, CA, USA, 1994.

[41] R.Guha. *Contexts : a formalization and some applications*. PhD thesis, PhD thesis, Stanford, CA, USA, 1992.

[42] M.Uschold and M.Grüninger. Ontologies : principles, methods, and applications. *Knowledge Engineering Review*, 11(2) :93–155, 1996.

[43] W3c. owl web ontology language use cases and requirements, 10 February 2004.

[44] Jena2 Inference support. http ://jena.sourceforge.net/inference/index.html, 2006.

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

[46] A.Agostini, C.Bettini, and 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.

[47] R.Kowalski and Sergot. Alogic based calculus of events. *New Generation Computing*, vol 4 :67–95, 1986.

[48] M.Shanahan and R.Miller. Some alternative formulations of the event calculus. *in Computational Logic : Logic Programming and Beyond*, pages 95–111, 2002.

[49] D.Nute. Defeasible logic. INAP'01 Proceedings of the Applications of prolog 14th international conference on Web knowledge management and decision support.

[50] J.Pollock. Defeasible reasoning. *Cognitive Science journal*, volume 11, 1987.

[51] G.Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, vol. 4 :145–180, 1973.

[52] R.Miller and M.Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, vol. 4(no. 5) :513–530, 1994.

[53] M.Shanahan. A circumscriptive calculus of events. *Artificial Intelligence*, vol.77(no. 2) :249–284, 1995.

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

[55] V.Lifschitz. Formal theories of action (preliminary report). *in Proc. of IJCAI*, vol. 87 :966–972, 1987.

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

[57] J.McCarthy. Mathematical logic in artificial intelligence. *Daedalus*, vol. 117(no. 1) :297–311, Winter 1988.

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

[59] F.Sadri. *Three recent approaches to temporal reasoning.* Temporal Logics and their Applications, Academic Press, London, 1987.

[60] K.Eshghi. *Abductive planning with event calculus*, volume vol. 1. Logic Programming : Proceedings of the Fifth International Conference and Symposium, MIT Press, Cambridge, MA, 1988.

[61] M.Shanahan. *Prediction is deduction but explanation is abduction.* Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, San Mateo, CA, 1989.

[62] M.Shanahan. *Epresenting continuous change in the event calculus.* Pitman, London, 1990.

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

[64] E.Mueller. Commonsense reasoning. *Morgan Kaufmann, San Francisco*, 2006.

[65] J.McCarthy and al. Some philosophical problems from the standpoint of artificial intelligence. *Stanford University*, 1968.

[66] M.Thielscher. Introduction to the fluent calculus. *Electronic Transactions in Artificial Intelligence*, pages 179–192, 1998.

[67] F.Baader and W.Nutt. *Basic description logics.* The Description Logic Handbook : Theory, Implementation, and Applications, Cambridge University Press, 2003.

[68] D.Baader and al. The description logic handbook. Theory, Implementation, and Applications.

[69] M.Smith, C.Welty, and D.McGuinness. Owl web ontology language guide. w3c recommendation, février 2004.

[70] I.Horrocks and al. A semantic web rule language combining owl and ruleml. *W3C Member Submission*, 21 May 2004.

[71] B.Parsia and al. Cautiously approaching swrl. *Url :http ://www.mindswap.org/papers/CautiousSWRL.* 2005.

[72] E.Sirin and al. "pellet : A practical owl-dl reasoner". *Web Semantics : science, services and agents on the World Wide Web*, 5(2) :51–53, 2007.