N° d'ordre: 2007-ISAL-0108 Année 2007

Thèse

Services Pervasifs Contextualisés : Modélisation et Mise en Œuvre

Présentée devant
L'Institut National des Sciences Appliquées de Lyon
(INSA de Lyon)

Pour obtenir
Le grade de Docteur
École doctorale Informatique et Information pour la Société
Spécialité : Informatique

Par

Dejene Ejigu Dedefa

Soutenue le 12 Décembre 2007 devant la Commission d'examen

Jury

Prof. Bruno DEFUDE	INT Paris	Rapporteur
Prof. Lionel BRUNIE	INSA de Lyon	Directeur de thèse
Dr. Marian SCUTURICI	INSA de Lyon	Co-Directeur de thèse
Prof. Aris OUKSEL	Université d'Illinois à Chicago	Examinateur
Prof. Jean-Marc PETIT	INSA de Lyon	Examinateur
Dr. Thierry DELOT	Université de Valenciennes	Examinateur
Dr. Richard CHBEIR	Université de Bourgogne	Examinateur

Ordering N°: 2007-ISAL-0108 Year 2007

Thesis

Context Modeling and Collaborative Context-Aware Services for Pervasive Computing

Submitted to the
National Institute of Applied Sciences
(INSA de Lyon)

In fulfillment of the requirement for
Doctoral Degree

Doctoral School of Computer and Information Sciences (EDIIS)
Affiliated Area: Computer Science

Prepared by
Dejene Ejigu Dedefa

Defended on 2 December 2007 in front of the Examination Committee

Committee Members

Prof. Bruno DEFUDE	INT Paris	Reviewer
Prof. Lionel BRUNIE	INSA de Lyon	Supervisor
Dr. Marian SCUTURICI	INSA de Lyon	Co-Supervisor
Prof. Aris OUKSEL	University of Illinois at Chicago	Examiner
Prof. Jean-Marc PETIT	INSA de Lyon	Examiner
Dr. Thierry DELOT	University of Valenciennes	Examiner
Dr. Richard CHBEIR	University of Bourgogne	Examiner

Acknowledgement

During my doctoral study, I have been accompanied and supported by many people. It is a pleasure that I have now the opportunity to express my gratitude to all of them.

I owe my most sincere gratitude to my supervisor Prof. Lionel Brunie. His sympathetic personality, enthusiasm and integral view on research with a mission for providing “only high-quality work and not less” have made a deep impression on me. Besides being an excellent advisor, he was as close as a relative and a good friend to me and all his students. I should also mention the hospitality of his wife Benedict and their children. I am really glad that I have come to know Lionel and his family in my life.

I also feel honoured and grateful to thank my co-supervisor Dr. Marian Scuturici, who has always kept an eye on the progress of my work and was always available when I needed his advice. His encouragement, guidances and critiques were valuable at each step of my work.

I am deeply indebted to Prof. B. Defude for reviewing my thesis and being a member of my Ph.D. examination committee, Prof. M. Parashar for reviewing my thesis. I would also like to extend my thanks to the other members of the examination committee, Prof. A. Ouksel, Prof. J. PETIT, Dr. T. Delot and Dr. R. Chbeir.

I warmly thank all colleagues in the LIRIS laboratory, INSA de Lyon, and specifically those in the SIP research group, Dr. Nadia Bennani, Addisalem Negash, Atechian Talar, Julien Gossa, Lyes Limam, Ny-Haingo Handrianarisoa, Omar Hasan, Rachid Saadi, Yaser Fawaz, Yonny Cardenas, Zeina Torbey and Françoise Conil. Similar thanks to the secretaries, Mrs Mabrouka Gheriassa and Mrs Christiane Ripon who were always available to make our lab a convivial place to work.

My sincere thanks go to the French Embassy in Addis Ababa for sponsoring my study, Dr. Dawit Bekele for encouraging me to decide to accept the offer and facilitating the process with the embassy, the department of Computer Science, AAU, for allowing me to use this opportunity. I am also grateful to the members of the Department of Computer Science and the Department of Mathematics, AAU, for their hospitality and care to my family while I am away.

My special thanks go to all members of the Ethiopian community and students in and around Lyon who, one way or another, have contributed to the success of my work. I thank Araya-Yohannes Bekele for his support to translate and edit the French version of the extended summary of the thesis. Similar thanks go to Girma Berhe with whom we share our daily dinner for three years and whose encouragement has been with me after he left Lyon.

I am also grateful to my friends, Girma Taye, Mulugeta Dinka, Sori Ararsa, Tadesse Degefu and their family who have always been a constant source of encouragement both for me and my family. Furthermore I extend my sincere thanks to all friends and relatives in Ethiopia, all over the world whose encouragement to me and my family has contributed to the success of this work.

Finally, I am thankful to my mother in-law, w/o Asegedech, on whose constant care and love for my family I have relied throughout my stay. I am grateful to my wife Rahel for the very special person she is and for her love and patience during my study. Similar thanks go to my children Eden, Robenus and Abigiya who suffered a lot by my absence from home and whose patience and love enabled me to complete my study successfully. « *Hey kids, dad is back-home!* » I am deeply grateful to my mother, w/o Nuritu, who sacrificed part of her life for that of mine and who formed part of my vision and taught me the good things that really matter in life.

Last but not least, thanks God, may your name be honored and glorified !

Dejene Ejigu, December 12, 2007 Lyon, France

Remerciements

Pendant mes études doctorales, j'ai été accompagné et soutenu par plusieurs personnes. C'est un plaisir pour moi de saisir cette occasion pour exprimer ma gratitude à tous.

Je dois ma gratitude la plus sincère à mon directeur de thèse, Prof. Lionel Brunie. Sa sympathique personnalité, son enthousiasme et sa vue intégrale sur la recherche avec une mission pour fournir ‘seulement un travail de haute qualité et pas moins’ m'ont profondément impressionné. En plus d’être un encadrant excellent, il était proche comme un parent et un bon ami à tous ses étudiants. Je devrais également mentionner l'hospitalité de son épouse Benedict et de leurs enfants. Je suis vraiment heureux de connaître Lionel et sa famille dans ma vie.

Je tiens également à remercier mon co-directeur de thèse, Dr. Marian Scuturici, qui a toujours gardé un œil sur le progrès de mon travail et qui était disponible quand j'avais besoin de ses conseils. Son encouragement, ses conseils et ses critiques étaient toujours très important à chaque étape de mon travail.

Je suis profondément endetté à Prof. B. Defude qui a passé en revue ma thèse et qui était aussi un membre du Jury lors de ma soutenance ainsi qu’au Prof. M. Parashar pour avoir passé en revue ma thèse. Je voudrais également remercier les autres membres du Jury, Prof. A. Ouksel, Prof. J. PETIT, Dr. T. Delot et Dr. R. Chbeir.

Je remercie fortement tous les collègues dans le laboratoire de LIRIS à l’INSA de Lyon et spécifiquement ceux du groupe de recherche de SIP, Dr. Nadia Bennani, Addisalem Negash, Atechian Talar, Julien Gossa, Lyes Limam, Ny-Haingo Handrianarisoa, Omar Hasan, Rachid Saadi, Yaser Fawaz, Yonny Cardenas, Zeina Torbey et Françoise Conil. Je voudrais également remercier les secrétaires, Mme Mabrouka Gheriassa et Mme Christiane Ripon, qui étaient toujours disponibles pour rendre notre laboratoire un endroit fonctionnel et convivial.

Mes remercîments sincères vont à l'Ambassade de France à Addis Ababa pour le financement de mes études, à Dr. Dawit Bekele pour m'avoir encouragé à décider d'accepter cette bourse de recherche et pour avoir facilité le processus avec l'Ambassade de France, et au département de l'informatique de l’Université d’Addis Abeba (AAU). Je suis également reconnaissant aux membres du département de l'informatique et du département de mathématiques d’AAU pour leur hospitalité et avoir veillé sur le bien-être de ma famille lors de mon absence.

Mes remercîments spéciaux vont à tous les membres de la communauté et des étudiants éthiopiens de Lyon qui ont contribué au succès de mon travail. Je remercie Araya-Yohannes Bekele pour la traduction et édition de la version française du résumé étendu de la thèse. Je remercie également Girma Berhe avec qui nous avons partagé nos diners quotidiens pendant trois années et dont l'encouragement a été toujours avec moi après son départ de Lyon.

Je suis également reconnaissant à mes amis, Girma Taye, Mulugeta Dinka, Sori Ararsa, Tadesse Degefu et leur famille qui ont toujours été une source constante d'encouragement pour moi et ma famille. En outre, mes remerciements sincères vont aux amis et parents qui sont en Ethiopie et par tout dans le monde dont l'encouragement à moi et à ma famille a contribué au succès de ce travail.

En conclusion, je suis reconnaissant à ma belle-mère, w/o Asegedech, sur qui j'ai compté pour le soin et l'amour constants qu’elle apportait à ma famille pendant tout mon séjour. Je suis très reconnaissant à mon épouse Rahel, pour la personne très spéciale qu'elle est et pour son amour et patience pendant mes études. Je remercie également mes enfants Éden, Robenus et Abigiya qui ont souffert beaucoup de mon absence de la maison et dont la patience et l'amour m'a permis d'achever mes études avec succès. « *Hé les enfants, papa est à la maison!* » Je suis profondément reconnaissant à ma mère, w/o Nuritu, qui a sacrifié une partie de sa vie pour la mienne et qui a contribué à ma vision en m’apprenant de bonnes choses qui sont vraiment importantes dans la vie.

Enfin, et pas des moindres, merci mon Dieu, que ton nom soit honoré et glorifié !

Dejene Ejigu, Décembre 12, 2007 Lyon, France

Résumé

Les systèmes pervasifs visent à intégrer des services fournis par des dispositifs répartis communicants. De tels environnements ont comme objectif d'optimiser l'interaction de l'utilisateur avec les dispositifs intégrés, par exemple en permettant à l'utilisateur d'accéder à l'ensemble des informations disponibles et en adaptant celles-ci aux conditions matérielles effectives (qualité de service réseau, caractéristiques du matériel de connexion). Cela impose aux applications d'adapter dynamiquement leur fonctionnement aux caractéristiques de l'environnement (notion de "contexte d'exécution").

Pour réaliser cette adaptation il est important de disposer d'un mécanisme efficace de capture et gestion du contexte et d'un mécanisme de raisonnement approprié. La gestion du contexte comprend la représentation, l'agrégation, l'interprétation, le stockage et le traitement des données contextuelles. Le raisonnement est le processus de déductions des nouveaux faits à partir des données contextuelles observées.

Dans cette thèse, nous proposons un modèle sémantiquement riche pour la collaboration, la représentation et la gestion du contexte. Nous utilisons un modèle de représentation du contexte fondé sur une approche hybride utilisant des ontologies et des bases de données relationnelles (nommè modèle HCoM : Hybrid Context Management model). Le modèle HCoM utilise l'ontologie pour la modélisation et la gestion des métadonnées riches en sémantique du contexte, et le schéma de la base de données relationnelles pour la modélisation et la gestion des données brutes du contexte. Les deux sont liés à travers des relations sémantiques construits dans l'ontologie. La séparation des ces deux éléments de modélisation nous permet d'extraire, charger, partager et utiliser seulement les données du contexte relevant afin des limiter la quantité de données dans l'espace de raisonnement.

Les éléments constitutifs du modèle HCoM sont les données contextuelles brutes, l'ontologie et les règles de inférence. Ces éléments sont organisés dans un modèle que nous appelons EHRAM: *Entité, Hiérarchie, Relation, Axiome* et *Métadonnée*. EHRAM est mappé à un schéma de base de données relationnelle pour la représentation des données contextuelles et permet une représentation compatible avec les langages à bas de balises pour son ontologie et ses règles d'inférence.

Cette richesse de modélisation nous permet de sélectionner de manière efficace les informations contextuelles pertinentes et ainsi d'améliorer les performances du processus de raisonnement mis en oeuvre dans l'analyse du contexte d'exécution.

Nous présentons également la plateforme logicielle d'intégration de services pervasifs que nous avons développée (nommé CoCA : *Collaborative Context-Aware service Plateform*). Cette plateforme s'appuie sur la méthodologie et les modèles de représentation et de gestion du contexte proposés dans la thèse. Elle permet une interaction "contextualisée" des services fournis par les dispositifs participants, offrant en particulier des mécanismes d'adaptation au contexte et de déclenchement proactif ou réactif de services en réponse à une évolution du contexte. Cette plateforme implémente le protocole JXTA dans ses composants de collaboration et utilise la librairie JENA pour le raisonnement (déclaration et interprétation des règles d'analyse du contexte).

Des démonstrateurs ont été développés et testés illustrant l'utilisation de la plate-forme dans trois cas d'utilisation liés à des domaines applicatifs variés : les réseaux sociaux, l'hôpital intelligent, l'adaptation d'IHM au contexte.

Les résultats obtenus illustrent la performance, la robustesse et l'extensibilité de l'approche proposée.

Les Mot clé : Informatique Pervasif, Réactivité au Contexte, Contexte Modélisation, Raisonnement Sémantique, Ontologie du Contexte, Informatique Collaborative.

Abstract

Pervasive or ubiquitous computing aims to integrate computing and computing appliances into the environment rather than having computers as distinct objects. This can be realized through applications that adapt their behavior to every changing environment. Such systems need to ensure that the adaptive behavior experienced is useful, relevant, non-distracting and consistent with individual and organizational goals. Such adaptation needs proper capturing, management and reasoning of constantly changing context. Context capturing involves extracting relevant context data about selected entities in the environment. Context management deals with representation, aggregation, interpretation, storage and processing of context data. Context reasoning is the process of drawing inferences or conclusions (unknowns) from known facts using information from the various sources of context.

The computationally intensive characteristics of context reasoning process, the presence of handheld or wearable, tiny and resource hungry computing devices, and the lack of a semantically rich context model have been a bottleneck for the development of such applications. Moreover, most of the current context-aware systems are based on ad-hoc models of context, which causes lack of the desired formality and expressiveness. They do not separate processing of context semantics from processing and representation of context data and structure.

In this thesis, we propose a semantically rich and a collaborative context representation and management model that uses a hybrid of ontology and database management approaches (called HCoM model: Hybrid Context Management model). HCoM model uses ontology for modeling and management of context semantics and relational database schema for modeling and management of context data. These two modeling elements are linked to each other through the semantic relations built in the ontology. Separation of the two context modeling elements allows us to extract, load, share and use only relevant context data into the reasoner in order to limit the amount of context data in the reasoning space. By doing this, we considerably improve the performance of the reasoning process

The building blocks of the HCoM model are context data, context ontology, and deduction rules. These data elements are organized into a context representation structure (called EHRAM: Entity, Hierarchy, Relation, Axiom and metadata). EHRAM is a graphical

context representation structure that serves as a context conceptualization model. EHRAM is mapped to a standard relational database schema for representation of its context component and is serialized to markup languages for representation of its ontology and rule component.

We also present a domain independent context-aware middleware platform (called CoCA: Collaborative Context-Aware service platform) under which our proposed context management model is implemented and used. CoCA uses data organized into the HCoM model as its data source and provides reasoning and decision services based on changing contexts. It triggers proactive and/or reactive actions and provides a collaboration interface between the pervasive peers. CoCA collaboration is based on JXTA protocols and its reasoning is based on Jena framework.

To evaluate the scalability and extensibility of the proposed model, reusability of the platform and performance of the collaboration process, we have developed a test case of the use of our context model in the platform using data from multiple scenarios: Community based network in a campus, smart hospital and adaptation of HCI to context.

Results obtained from our experiment show that compared to other related works in the domain, our approach gives a robust, extensible and scalable model and platform for the development of context-aware applications in pervasive environment.

Keywords: Pervasive Computing, Context-Aware Computing, Context Modeling, Semantic Reasoning, Context Ontology, Collaborative Computing.

Table of Contents

List of Figures

List of Tables

Résumé Etendu

Services Pervasifs Contextualisés
(Modélisation et Mise en Œuvre)

1. Introduction

Le nombre croissant d'ordinateurs et d'utilisateurs d'Internet a conduit au développement de structures de coopération telles que la technologie pair-à-pair en informatique qui a un grand potentiel d'évolutivité. L'intégration des clients mobiles dans un environnement distribué et la mise en réseau ad-hoc des composants dynamiques deviennent de plus en plus importantes dans tous les domaines d'application. [Mattern, 2003] a indiqué que, vu le progrès technique continu en informatique et en communication, nous nous dirigeons vers l'utilisation de réseaux informatiques tout englobant nommée l'informatique pervasif ou ubiquitaire.

Les systèmes de l'informatique pervasif visent à adapter leur comportement en vue de répondre aux besoins des utilisateurs suivant le changement du contexte de l'environnement et de ses composants. Les dispositifs pervasifs établissent une connexion ad-hoc entre eux et peuvent être connectés aux différents types d'appareils capturant les changements dans l'environnement. La Figure 1 montre le flux dans la chaîne de l'évolution de l'informatique centralisée à l'informatique pervasif tel que présenté par [Satyanarayanan, 2001] et [Strang, 2004]. Cette classification présente le problème de réactivité au contexte (context awareness) au cœur du problème de l'informatique pervasif. Avec l'addition de nouvelles composantes de la chaîne, la complexité de ces problèmes augmente de façon multiplicative (⊗) plutôt que de la façon additive (⊕).

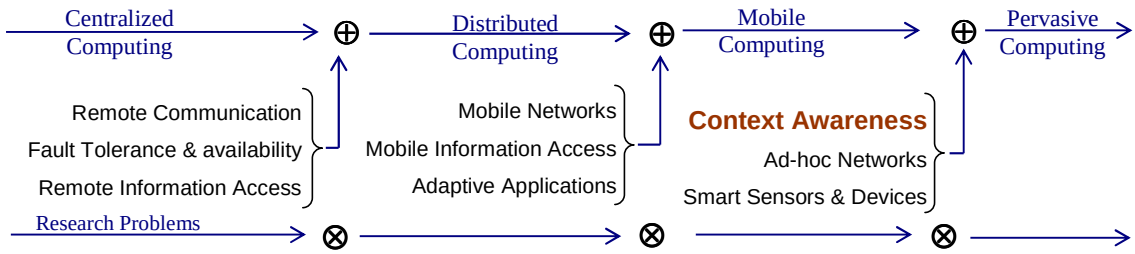


Figure 1: Réactivité au contexte dans la chaîne d'évolution de l'informatique

1.1 Motivation

En voici un scénario pour motiver l'informatique pervasive réactive au contexte : le scénario de l'hôpital intelligent. Considérons un hôpital intelligent où des patients, des infirmières, des médecins, etc. sont impliqués. Supposons que l'hôpital est équipé de technologies (matériel et logiciel) de capteur de contexte dans les chambres, les couloirs et

le jardin à la disposition des individus impliqués. Un système de l'informatique pervasive réactive au contexte adapté pour la surveillance et le suivi des patients dans les hôpitaux aide à minimiser l'engagement des spécialistes aux activités moins importantes. L'intervention humaine pourrait être nécessaire uniquement lors d'une alerte par le système.

La Figure 2 illustre un cas d'utilisation spécifique du scénario de l'hôpital intelligent. *Un matin, le docteur Pascal et ses collègues sont en réunion de consultation hebdomadaire (1). Michel, le patient, est dans un jardin pour profiter du soleil matinal (2). Soudain, Michel se sent épuiser et tombe par terre (3). Les portables et les insignes qu'il porte remettre immédiatement tous les informations nécessaires au dispositif informatique se trouvant à proximité (4). Ensuite le système envoie (5) un message d'alerte à Ada (6), l'infirmier, sur son téléphone intelligent (7). Sachant également les horaires du docteur Pascal, de son agenda, qu'il est dans la réunion, le système lui informe (8) par message SMS sur son téléphone cellulaire qui clignote la lumière rouge à la réception de ce type de message d'urgence quand il est en réunion. La camera face au jardin (9), où Michel est situé, est activé et son image est envoyée (10) à l'ordinateur central pour l'adapter et la diffuser (11, 12, 13) à tous les terminaux concernés (14, 15, 16). Les secouristes (17) sont également informés (18) de la situation. À la suite, Michel est amené (19, 20) au chambre de traitement (21) par les secouristes d'urgence. Le docteur Pascal, qui a déjà été au courant de la situation actuelle de son patient, a fait toutes les consultations nécessaires (22, 23, 24), la préparation de médicaments appropriés et il est déjà dans la salle de traitement (25). Ada est également dans la salle de traitement (26) pour fournir l'aide nécessaire au patient.*

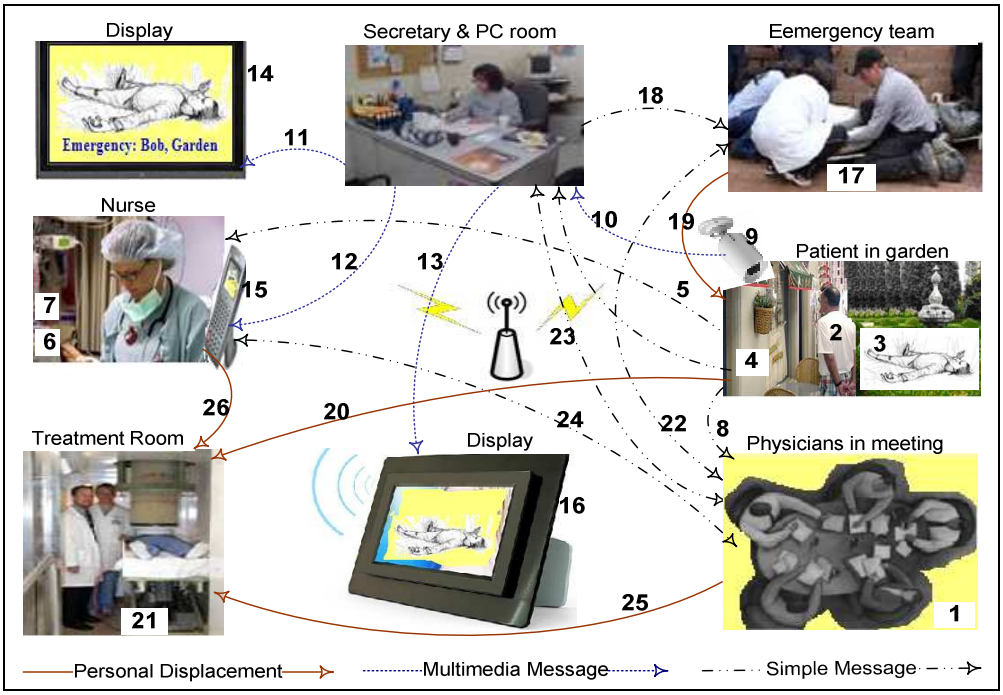


Figure 2: Scénario use case dans l'hôpital intelligent

Ce travail est donc *motivé* par deux observations principales que nous avons relevées au sujet de l'environnement de l'informatique pervasive et le système réactif au contexte : la nécessité d'un modèle de gestion du contexte dans un cadre générique et la nécessité d'une plateforme pour développement des systèmes réactive au contexte dans l'informatique pervasive.

2. Modélisation de contexte: Le modèle EHRAM

Nous considérons le contexte comme un terme opérationnel dont la définition dépend de l'intention pour laquelle il est recueilli et de l'interprétation des opérations sur une entité plutôt que les qualités intrinsèques de l'entité et les opérations eux mêmes. Une chose servant de contexte pour une personne pourrait ne pas l'être pour une autre. Aussi, un objet servant de contexte dans certaine situation pour une personne particulière ne pourrait pas être importante pour la même personne dans une autre situation. Donc, le contexte est une réponse à la gamme de questions comme: comment, où, quand, quoi, qui et quel sur l'entité descripteurs et leur interaction entre eux qui affectent des actions prises ou des actions admises par les entités.

2.1 Le modèle EHRAM

Nous présentons maintenant notre nouveau modèle de représentation de contexte nommé EHRAM. EHRAM est un méta modèle qui utilise l'ensemble d'entités (E), ensemble de la hiérarchie (H), l'ensemble de relations (R), l'ensemble de relations de axiomatique (A) et l'ensemble de métadonnées (M) pour représenter des données du contexte et son sémantique. Le nom EHRAM est composé des initiales des composantes d'EHRAM décrites ci-dessous.

- **E** est l'ensemble d'entités pour lesquelles on capture le contexte.
- **H** est l'ensemble de relations qui forment un graphe acyclique dirigé inversement (DAG inversé) sur les entités. Les *nœuds* du graphe représentent des entités et les *arcs* du graphe représentent des relations hiérarchiques. L'entité racine qui se trouve au sommet de la hiérarchie du graphe est une entité globale nommée *ContextEntity*.
- **R** signifie l'union de l'ensemble des relations R_e et R_a. R_e est l'ensemble des relations ayant leur domaine et leur range de l'ensemble E. R_a est l'ensemble des relations définies à partir des entités E aux littérales représentant les attributs de l'entité.
- **A** est l'ensemble des relations axiomatiques. Une relation axiomatique est une relation sur des relations. Par exemple, si nous définissons une relation r_1 comme une relation transitive, alors r_1 obéit la propriété (axiome) transitivité: (e_1, r_1, e_2) and (e_2, r_1, e_3) => (e_1, r_1, e_3).

- **M** est l'ensemble de métadonnées sur une instance de relation définie. Par exemple, si nous avons une déclaration qui dit: "*Bob a indiqué qu'Alice est située dans le jardin ce matin*". Les expressions qualificatives comme "*Bob a indiqué*" et "*ce matin*" sont des métadonnées de la déclaration faite au sujet d'Alice. Elles répondent aux questions *qui* et *quand* concernant le contexte de base.

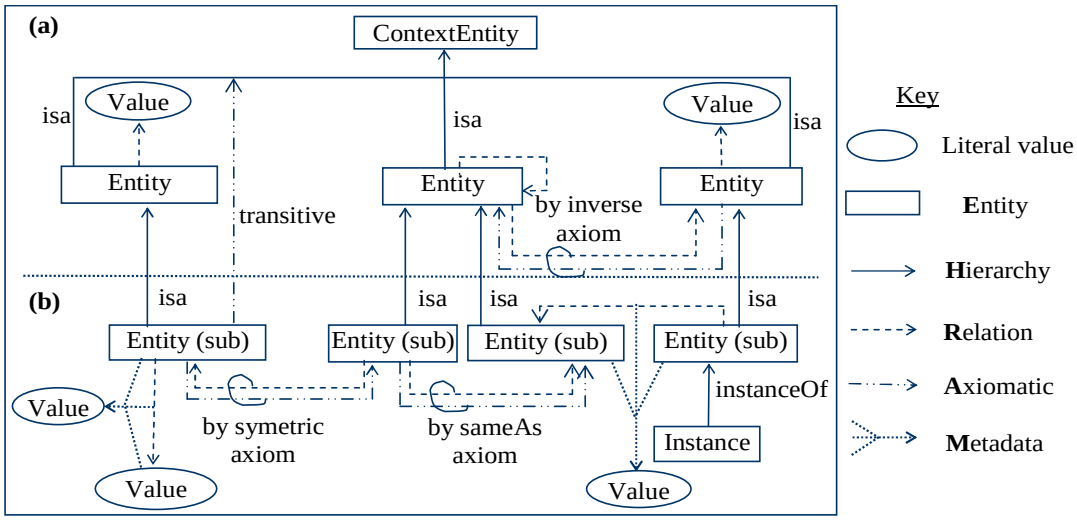


Figure 3: Représentation des composants de l'EHRAM

La figure 3 est une représentation graphique de la structure de l'EHRAM qui montre les hiérarchies, les entités, les relations d'entités, les relations d'attributs, les relations axiomatiques, les métadonnées et les couches où couche (a) est la couche générique et couche (b) est la couche de domaine.

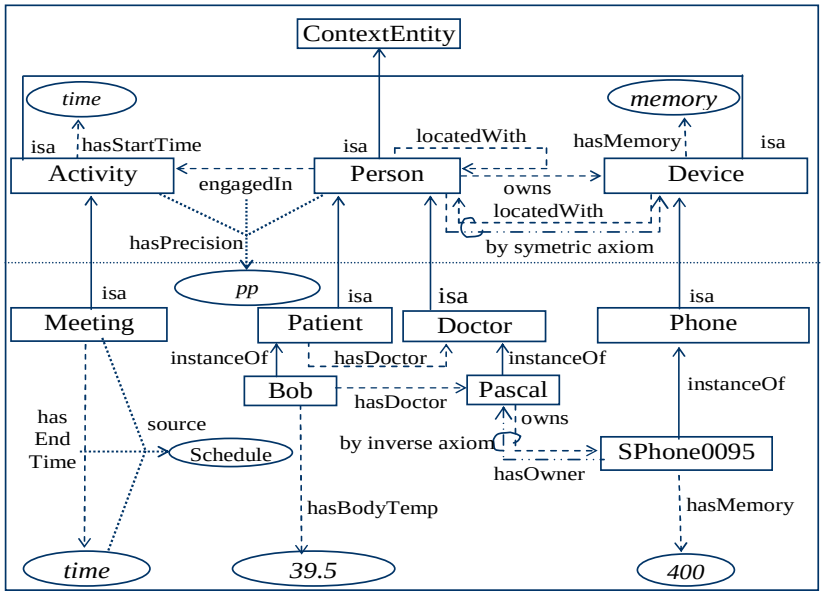


Figure 4: Un exemple des composants de l'EHRAM

Un exemple qui représente des composantes du modèle EHRAM en utilisant quelques données issues de l'application dans un domaine médicale est donné dans la Figure 4.

Certaines relations dans le diagramme sont définies à avoir les axiomes associés et certaines ont des métadonnées. Des exemples de relations qui ont les axiomes associés sont : (*Person, locatedWith, Device*) et (*Pascal, owns, SPhone0095*). Dans le diagramme, « *locatedWith* » est défini comme une relation symétrique et obéit donc à la propriété (axiome) symétrique. De même, puisque « *owns* » est défini comme étant l'inverse de « *hasOwner* », il obéit à la propriété (axiome) inverse qui signifie que la relation *(SPhone0095, hasOwner, Pascal)* est automatiquement vraie. La relation *(Person, engagedIn, Activité)* a une métadonnée qui raconte sa précision représentée par *hasPrecision*.

De plus, un axiome est une phrase, une proposition ou une règle qui est considérée comme valide, et qui sert de point de départ nécessaire et la logique de l'expression formelle pour déduire et inférer logiquement. La description de certains axiomes génériques : *sameAs, inverse, symétriques* et *transitives* sont présenté comme :

$\forall r \in R$ symmetric(r) $\Leftrightarrow (\forall e_1, e_2 \in E,\ r(e_1,e_2) \Rightarrow r(e_2,e_1))$

$\forall r \in R$ transitive(r) $\Leftrightarrow (\forall e_1, e_2, e_3 \in E, r(e_1,e_2) \wedge r(e_2,e_3) \Rightarrow r(e_1,e_3))$

$\forall r_1, r_2 \in R$ inverse$(r_1,r_2) \Leftrightarrow (\forall e_1, e_2 \in E, r_1(e_1,e_2) \Rightarrow r_2(e_2,e_1))$

$\forall r_1, r_2 \in R$ sameAs$(r_1,r_2) \Leftrightarrow (\forall e_1, e_2 \in E, r_1(e_1,e_2) \Rightarrow r_2(e_1,e_2))$

De même, les relations axiomatiques basées sur le domaine d'application sont utilisées pour indiquer des axiomes et des règles qui sont utilisés pour en déduire d'autres connaissances utiles pour le raisonnement. Par exemple, nous pouvons créer des règles de déduction basées sur le domaine d'application et qui servent comme axiomes de raisonnement dans le domaine.

$\forall$d instanceOf Doctor, p instanceOf Patient: hasDoctor(p,d) $\wedge$ engagedInActivity(p, takeTreatement) $\Rightarrow$ engagedInActivity(d, giveTreatement)

Dans le modèle EHRAM, les métadonnées représentent une relation qui décrit l'instance d'une autre relation. Par exemple, si on nous donne l'information contextuelle "*Patient isLocatedIn Garden*", à partir de cette déclaration, nous pouvons alors faire d'autres déclarations pour répondre à des questions comme: Qui a donné cette information? Quel service est utilisé pour rapporter l'information? Quand s'est-il passé? L'information, est-elle exact? Pourquoi le sujet est-il dans cet état? Que va-t-il se passer après? etc.

2.2 EHRAM et l'UML

Une trace incrémentale des concepts en EHRAM à l'UML est donnée comme suit. Un schéma de la classe de l'UML basé sur cette trace est donné par la Figure 5.

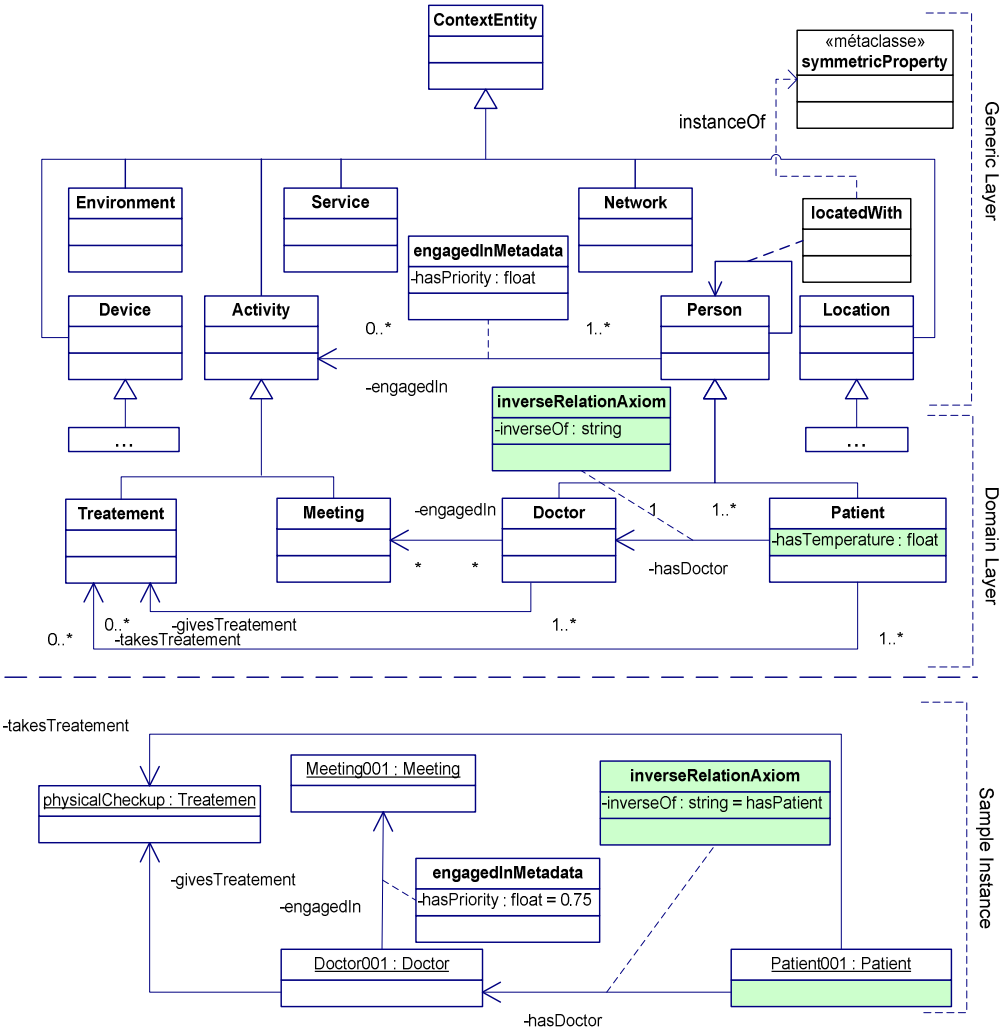


Figure 5: Représentation d'UML du modèle EHRAM

- *L'entité* dans le modèle EHRAM peut être représentée par une *classe* de l'UML.
- Le concept de la relation *hiérarchique* dans le modèle EHRAM peut être représenté par la relation de généralisation dans l'UML.
- *La relation de l'entité* dans le modèle EHRAM peut être représentée par *la relation d'association* en UML, et *la relation de l'attribut* dans le modèle EHRAM peut être représentée en utilisant les *attributs* de la classe de l'UML.
- *La relation axiomatique* dans le modèle EHRAM peut être représentée comme *une classe d'association* dans l'UML. Le concept de *méta-classe* peut également être utilisé pour représenter les propriétés axiomatiques.

- *La métadonnée* dans le modèle EHRAM peut être représentée par *une classe d'association* dans l'UML.

Certaines des limites de l'UML pour représenter le méta modèle EHRAM peuvent être surmontées par l'utilisation des *méta-classes* et des *classes d'association*. Cependant, nous avons encore un soutien limité pour la représentation de l'aspect sémantique des données contextuelles. Enfin, les outils de modélisation de l'UML ne peuvent être utilisés qu'en partie pour formaliser la représentation du modèle EHRAM. Nous allons donc continuer d'examiner d'autres méthodes.

2.3 EHRAM et les modèles relationnels

Étant donné l'ensemble d'entités E et l'ensemble de valeurs V (tires des entités et des valeurs littérales), la relation R est le sous-ensemble du produit cartésien de E et V.

$R \subseteq \{(e_i,v_j): e_i \in E, v_j \in V\}$

Nous sommes intéressés par tous les éléments qui ont un sens :

$(e_i,v_j) \in R$ qui peut aussi être représenté par :

$\{R(e_i,v_j): (e_i,v_j) \in R\}$ ou dans une forme plus linéaire

$\{(e_i,R,v_j): (e_i,v_j) \in R\}$

Ce triplet peut être utilisé pour définir un contexte (C), comme suit:

$C \equiv \{(e_{i,k}, r_k, v_{j,k}) : e_{i,k} \in E_k, r_k \in R, v_{i,k} \in V_k\}$

En utilisant le contexte C, des méta-données CM, des méta-relations RM et des méta-valeurs VM :

$CM \equiv \{(c_i, rm_k, vm_j) : c_i \in C, rm_k \in Rm, vm_j \in Vm\}$ ou
$CM \equiv \{((e_i, r_k, v_j), rm_l, vm_p): e_i \in E, r_k \in R, v_j \in V, rm_l \in RM, vm_p \in VM\}$

Exemple de contexte avec des métadonnées.

("Pascal, isEngagedIn, Meeting005", hasSource, Agenda)

("Pascal, isEngagedIn, Meeting005", hasPrecision, xx%)

Pour représenter l'aspect sémantique des données du contexte dans le modèle EHRAM, nous allons encore continuer à examiner d'autres méthodes.

2.4 EHRAM et le modèle RDF

La caractéristique principale des données du contexte c'est qu'il possède un acteur ou une entité nommée « *subject* ». La valeur de contexte définie sur le *subject* est exprimée en termes de propriétés multiples. Nous allons utiliser le nome « *prédicat* » et « *object*» pour représenter la situation de *subject* en ce qui concerne sa propriété spécifique. Cette convention va de pair avec le formalisme de représentation du triplet RDF <*subject, predicate, object*>.

Nous utilisons RDF et son extension appelée la réification de RDF afin de représenter la donnée du contexte, de comparer les assertions logiques de différents témoins et de déterminer leur crédibilité. Le message "*John is six inch tall*" est une affirmation de la vérité qui commet l'expéditeur à la réalité, alors que la déclaration réifiée, "*Marie reports that John is six inches tall*" attribue cet engagement à Marie.

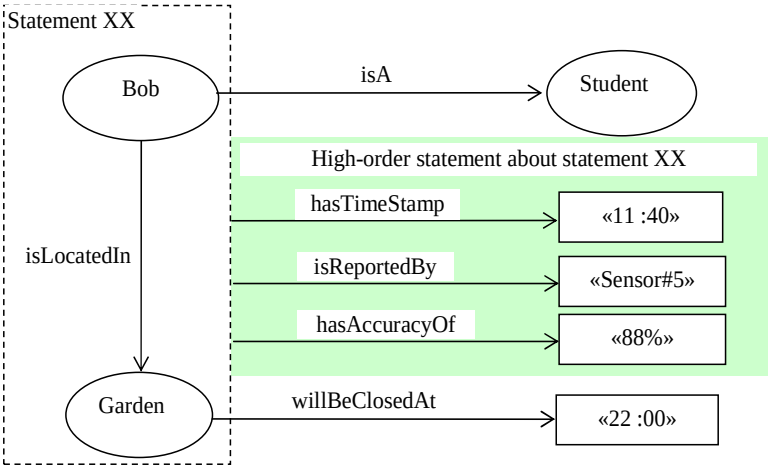


Figure 6: La représentation de la métadonnée de contexte en utilisant réification

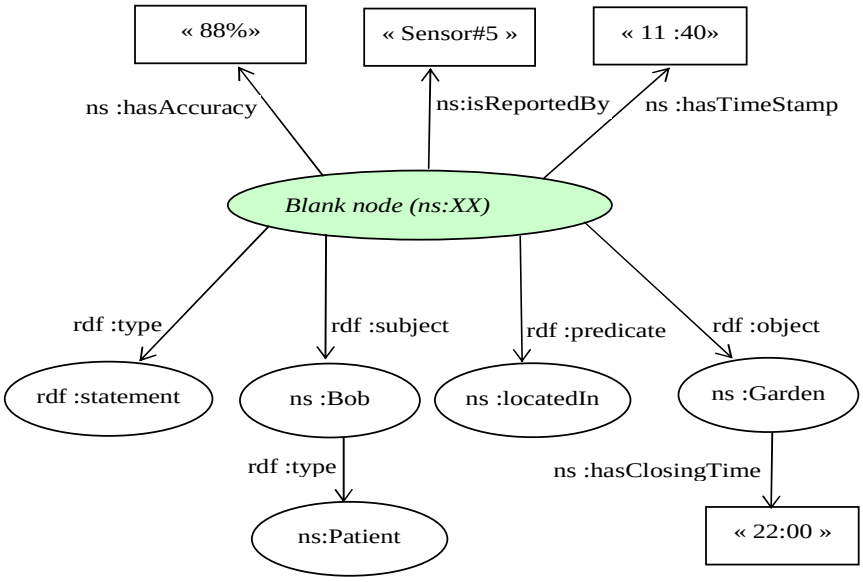


Figure 7: Modèle de RDF pour les métadonnées de contexte réifiées

De la même manière, des données réifiées de RDF contiennent chaque déclaration originale comme une ressource et les nouvelles déclarations faites à ce *subject*. Les quatre propriétés utilisées pour modéliser la déclaration originale de la ressource de RDF sont le *subject*, le *predicate*, l'*object* et le *type*. Une nouvelle ressource ayant ces quatre propriétés représente la déclaration originale et peut être utilisé comme *subject* ou *object* d'autres déclarations avec ses déclarations additionnelles. La figure 6 montre une démonstration de la représentation de métadonnées utilisant la déclaration da la réification du contexte. La figure montre un exemple de la déclaration triple: "*bob locatedIn library*". Cette déclaration peut être réifiée par méta informations supplémentaires comme "*isReported by sensor # 5*", "*hasAccuracy 88%*", "*hasTimeStamp 11:40 today*", etc. La Figure 7 illustre un modèle de données RDF équivalent aux données du contexte réifiées.

3. Gestion de Contexte: Le modèle HCoM

Dans ce chapitre, nous essayons d'étudier comment notre EHRAM, le modèle conceptuel de la représentation du contexte, peut être mappé au modèle de données relationnelles et au modèle de l'ontologie. Nous montrons les avantages et les inconvénients des deux approches. Nous proposons enfin une nouvelle approche hybride pour la modélisation de gestion du contexte (nommé: HCoM model).

Une base de données relationnelle est un modèle stable utilisée dans une large gamme d'applications pour la gestion de base de données. Dans le modèle EHRAM, le contexte est représenté par la combinaison des entités, des hiérarchies, des relations, des axiomes et des métadonnées. Dans le modèle de la base de données relationnelle, un modèle de relation d'entité (modèle ER) est utilisé pour représenter les entités, les attributs et les relations. Un algorithme de mappage étape par étape du modèle EHRAM au modèle relationnel offre un schéma de base de données relationnelle (RCDB) de la Figure 8.

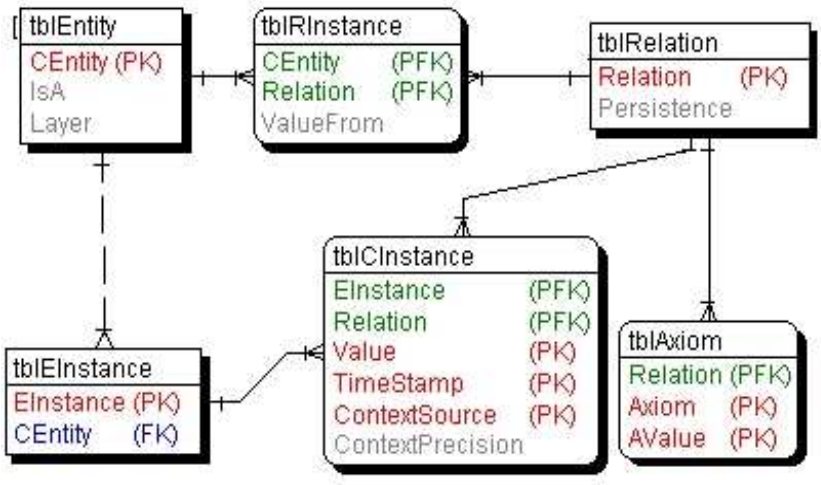


Figure 8: Schéma généralisé de la RCDB

Une ontologie peut être utilisée pour représenter la sémantique, le concept des relations et des axiomes dans les données du contexte.

Les outils de représentation ontologie fournissent une représentation largement accepté et formelle de sémantique de contexte dans le but d'interpréter et de raisonner sur les informations contextuelles. Les outils de l'ontologie, cependant, ne sont bons que pour la représentation statique des connaissances dans un domaine. Ils ne sont pas conçus pour la capture et le traitement des informations dans un environnement dynamique et évolutif. Par ailleurs, les langues de l'ontologie et leurs formats de sérialisation sont textuels (XML/RDF/RDFS/OWL) et ne sont donc pas conçus pour l'optimisation, le traitement et la récupération efficaces des requêtes de larges données de contexte.

Les modèles relationnels, d'autre part, fournissent des interfaces standard et l'optimisation des requêtes. Ils fournissent des outils pour la gestion de base de données du contexte et des outils pour recevoir et envoyer des notifications sur les changements de contexte. Le modèle relationnel, cependant, n'est pas conçu pour l'interprétation sémantique de données.

3.1 HCoM : Un modèle hybride pour la gestion de contexte

Nos rationnelles derrière la nécessité du modèle hybride de contexte sont : distinguer la gestion des données du contexte et la gestion de sa sémantique, les traiter séparément et rassembler les résultats pour un meilleur raisonnement et support décisionnel dans le system de informatique pervasive basé sur la réactivité au contexte. Nous utilisons l'approche ontologique pour gérer la sémantique de contexte et l'approche relationnelle pour gérer les données de contexte. Nous présentons un modèle hybride nommé le modèle HCoM (Hybride Contexte Management model). Le modèle HCoM vise à combiner les meilleurs des deux mondes.

Le module sélecteur/d'élagage (selector/pruning) dans HCoM fournit un moyen de choisir et de charger qu'une partie du contexte statique des données étant donné l'ensemble des données de contexte $T_c = \{c_1, c_2 \dots c_t\}$ et l'ensemble des données de contexte approprié comme $A_c=(c_1,c_2, \dots c_a)$. Par l'élagage de l'ensemble des données de contexte inappropriée de T_c, on obtient les données du contexte appropriée (sélectionnée) $S_c= \{c_1, c_2 \dots c_s\}$. Les deux mesures de la performance de l'algorithme d'élagage sont la qualité du raisonnement et la rapidité (des réponses aux requêtes). Ces mesures de performance dépendent de la différence entre les ensembles A_c et S_c.

Étant donné que tous les autres paramètres sont constants, la performance concernant la qualité du raisonnement d'un système qui utilise l'élagage (la sélection) des données est inversement proportionnelle à la valeur de |P|. Egalement, la performance de la vitesse est inversement proportionnelle à la valeur de |Q|. En supposant qu'un processus de raisonnement exige toujours un sous-ensemble de l'ensemble de contexte des données, c.-à-d. $A_c \subset T_c$ est toujours vrai, la plus grande perte de vitesse survient lorsque $T_c \equiv S_c$ mais, d'autre part, il garantit une qualité optimale. Un exemple d'un modèle avec une qualité optimale en termes de chargement des données complètes dans le reasoner est notre modèle GCoM. D'autre part, le modèle GCoM souffre d'une mauvaise qualité de la vitesse.

$$T_c \equiv S_c \Rightarrow A_c \subseteq S_c \Rightarrow P \equiv \varnothing \quad (quality \ \ is \ optimal \ )$$

Afin d'améliorer à la fois la qualité et la vitesse, l'algorithme d'élagage doit être choisi de telle sorte que les valeurs de |P| et |Q| s'approchent de zéro. En termes réels, il désigne que le critère de sélection/élagage est pertinent à l'intérêt de l'utilisateur.

Le chargement de données seulement pertinentes minimise la taille de l'espace de raisonnement et réduit la surcharge inutile du moteur d'inférence pour améliorer la performance globale du service réactivité au contexte. Cela aide à surmonter les limitations de l'absence d'évolutivité de la plupart des systèmes de raisonnement à l'augmentation constante du volume des ressources de raisonnement dans l'environnement pervasif.

3.2 Une architecture pour le modèle HCoM

La figure 9 montre les composants et les fonctionnalités qui représentent une architecture en couches du modèle HCoM. Les flèches dans le diagramme représentent le flux de l'ensemble des données différentes de ou vers chaque composante.

Context Filter : Reçoit une instance de données d'un nouveau contexte, qui peuvent être saisies à l'aide de capteurs matériels ou de capteurs logiciels, puis valide et crée le journal de contextes à partir duquel des copies des instances de contexte statique sont sélectionnées et ajoutées à la RCDB pour une utilisation future. La figure 10 montre un pseudo code d'un algorithme de filtrage du contexte. Cet algorithme est activé quand le système fonctionne afin de filtrer et décider si le contexte est utile en fonction de son facteur de fiabilité (lignes 9-17). Il vérifie également si le contexte est de catégorie statique (lignes 19-22) qui, si c'est le cas, doit être stocké pour une utilisation ultérieure ou utilisée une seule fois.

Context Selector: Utilise les informations historiques et actuelles de l'utilisateur, des dispositifs disponibles, des politiques institutionnelles etc. pour choisir et charger uniquement les données de contexte pertinentes. Cela permet de surmonter les limitations de l'absence d'évolutivité des systèmes de raisonnement. La figure 11 montre les étapes impliquées dans le processus de sélection.

Rules : Les règles en CoCA proviennent de trois sources différentes : des règles définies par l'utilisateur, des règles dérivées des politiques de l'entreprise et des règles dérivées de données de l'historique des décisions prises dans le passé.

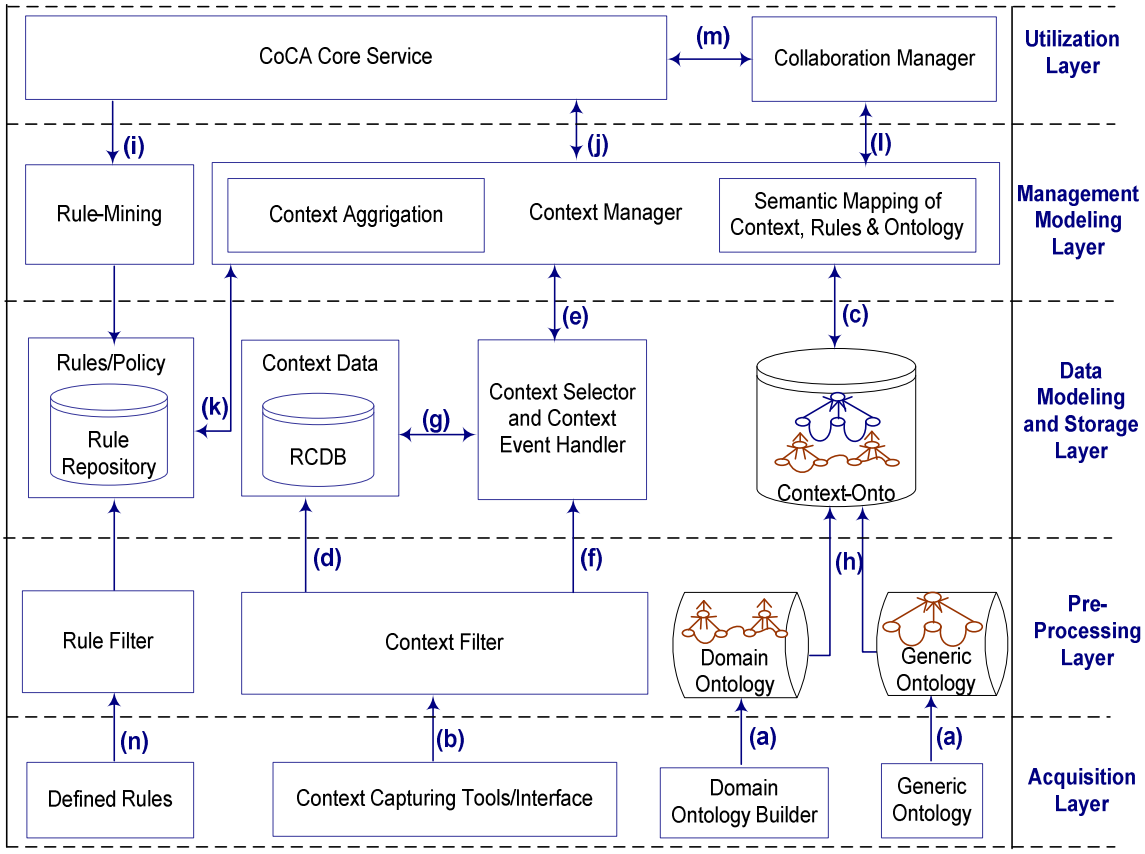


Figure 9: Architecture pour le modèle HCoM

Contexte-Onto est créée à partir des couches de domaine génériques et du modèle EHRAM et sert d'une ontologie référentielle.

RCDB est créé à partir du contexte statique du modèle EHRAM en utilisant le schéma des données décrites dans la section de la modélisation du contexte relationnel. Cette information est actualisée avec de nouvelles données du contexte statique qui sont capturées au cours de l'exécution. RCDB peuvent être stockés en utilisant n'importe quel système de gestion de base de données standard. Nous utilisons MySQL Comme un back-end pour stocker notre base de données de contexte et ainsi que son schéma.

Context Manager : Intègre et envoie les ressources de raisonnement nécessaires du modèle HCoM au moteur d'action de CoCA nommé RAID en mode-push chaque fois qu'un nouveau contexte est acquis. Cette action est basée sur l'‘information de déclenchement dans la notification d'événement de contexte qui est créé par le module de filtre quand un nouveau contexte est capturé.

Collaboration Manager: Basé sur les décisions du sélecteur de contexte, s'il n'y a pas suffisamment de données pertinentes dans l'appareil concerné, le gestionnaire de collaboration *(collaboration manager)* initie l'échange du contexte entre voisins proches. Ce module utilise le principe de la superposition d'un réseau virtuel qui utilise le protocole pair-à-pair de Jxta [JXTA, 2007].

```
1.  While (System_Is_Running)
2.  {
3.    newContext=null
4.    ContextBlock=new ContextClass()//context and all related data
5.    repeate until newContext!=null
6.    {
7.      newContext=ContextBlock.getNewContext()
8.    }
9.    reliabilityFactor=0
10.   if (ContextBlock.hasReliableSource()
11.   {
12.     reliabilityFactor=1
13.   }else
14.   {
15.     reliabilityFactor=ContextBlock.estimateSourceReliability()
16.   }
17.   if (reliabilityFactor> ContextBlock.reliabilityThereshold())
18.   {
19.     if (ContextBlock.hasStaticCategory())
20.     {
21.       ContextBlock.addContextToRCDB()
22.     }
23.     ContextBlock.addContextToContextOnto()
24.     ContextBlock.sendNoticeToCoCA()
25.   }else
26.   {
27.     ContextBlock.inValidContextError()
28.   }
29. }
```

Figure 10: Algorithme pour le filtre du contexte

```
//Algorithm for pruning non relevant entities (selecting relevant entities)
Input:  T (set of all entities in the EHRAM hierarchy graph)
        E1 (set of entities identified at the time of initialization)
        E2 (set of entities specified at the time of initialization)
        getProbablity() (a function to predict the probability of occurrence based on history data)
        getTH(), estimateTH() (functions that set threshold value)
Output: S (set of relevant entities selected from the set T) => (S ⊆ T)

Let τ ← 0 (threshold probability for selection of entity, value between 0 and 1 inclusive)
Let S1, S2, S3, S4, S5, S', SS ← ∅
Let δ ← DefaultDepth // Depth of search space for searching related entities   (δ>=1)
For every εi ∈ E1 do
        S1 ← S1 ∪ {εi}
End do
If (user has preferences)
        For every εi ∈ E2 do
                S2 ← S2 ∪ {εi}
        End do
        τ=getUserTH() // user specified threshold cf. Fig. 4.12
End if
S3 ← S1 ∪ S2
If τ=0 then τ = getApplicationTH() // default user threshold cf. Fig. 4.12
If τ=0 then τ = getDefaultTH() // default application threshold cf. Fig. 4.12
For every εi ∈ (T \S3) do
        ρ←getProbability(εi, S3) // conditional probability Fig. 4.12
        If (ρ ≥ τ)
                S4 ← S4 ∪  {εi}
        End if
End Do
SS ← S3 ∪ S4
S←SS
Depth←1
Repeat
        For every λi ∈ SS do // SPARQL/SQL
                S' ← S' ∪ {θi | hasRelation (λi, θi)}
                S' ← S' ∪ {θj | hasRelation (θj, λi)}
        End do
        S ← S ∪ S'
        SS ← S'
        S' ← ∅'
        Depth++
Until Depth>δ
Return S
```

Figure 11: Algorithme pour la sélection du contexte pertinent

3.3 Résumé

Dans ce chapitre, nous avons présenté notre nouvelle approche sémantiquement riche pour la modélisation de la gestion du contexte. Elle utilise l'hybride de l'ontologie et des principes de base de données pour la modélisation de gestion des données du contexte et de

la sémantique du contexte. L'ontologie représente l'aspect sémantique des données du contexte et le schéma relationnel représente les données du contexte elles-mêmes.

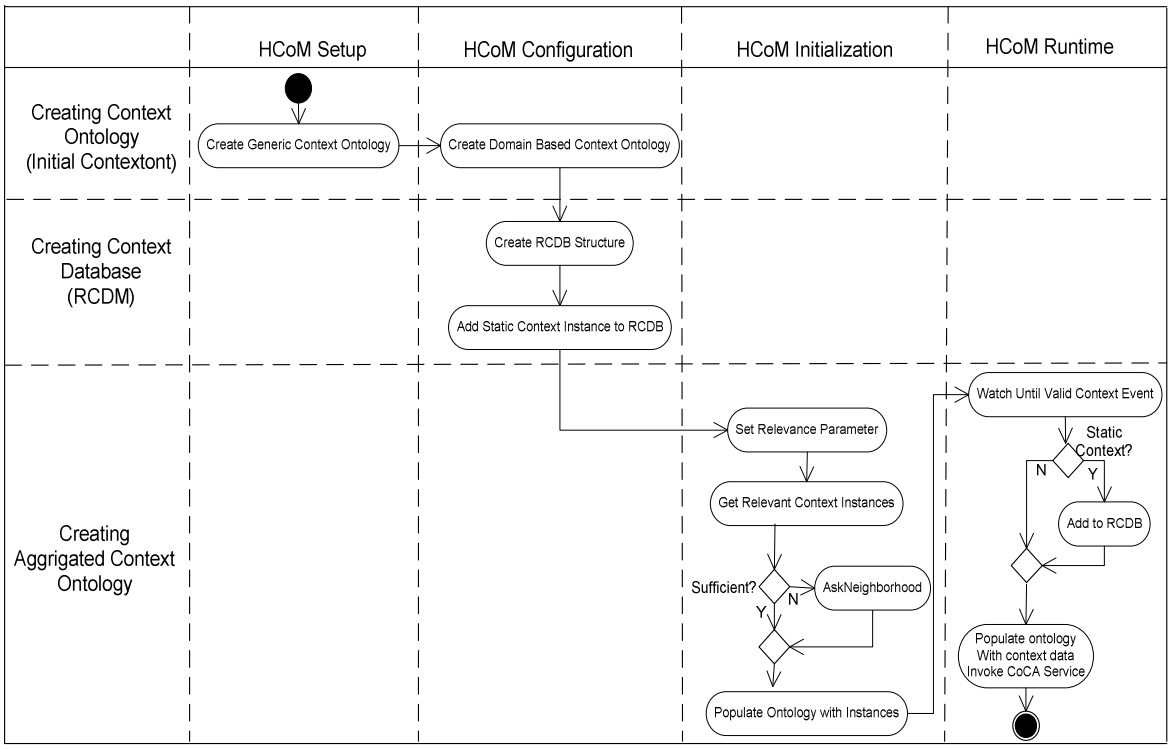


Figure 12: Flux du processus HCoM

La figure 12 montre le résumé du diagramme du système HCoM allant de la configuration initiale à l'exécution finale et la livraison des données. Les partitions verticales montrent les quatre états : état d'installation, état de configuration, état d'initialisation et état de temps d'exécution. Les partitions horizontales montrent les catégories des activités.

Le modèle HCoM est hybride ce qui signifie qu'il hérite des caractéristiques importantes de l'approche ontologique, de l'approche graphique, de l'approche du langage "markup" et de l'approche de modélisation relationnelles. Ainsi, HCoM modèle répond le mieux aux exigences de [Strang, 2004] : distributed composition, partial validation, richness and quality of information, incompleteness and ambiguity, level of formality, and applicability to existing environments.

4. Plateforme CoCA : un service collaboratif réactive au contexte

Un système conscient du contexte (ou géré par le contexte) doit recueillir des informations auprès de l'environnement ou la situation de l'utilisateur (acquisition du contexte), de traduire ces informations dans le format approprié, combiner l'information du

contexte afin de générer un contexte de hauteur nouveau, d'effectuer le raisonnement, de prendre des actions fondées sur l'information de contexte, et de rendre les informations accessibles aux autres applications et au voisinage.

4.1 Acquisition de contexte : Exemple sur le positionnement à l'intérieur

Nous avons développé un système de localisation basé sur le signal WiFi dans l'emplacement intérieur où le titulaire d'un appareil mobile se trouve. Comme dans tout processus de prédiction, on a deux grandes phases d'activités : l'apprentissage et la prédiction.

Durant la *phase d'apprentissage*, une personne titulaire d'un PDA se déplace dans les pièces et espaces du bâtiment afin d'obtenir le maximum de données possible sur la force du signal WiFi. L'ensemble des *n* signaux de fréquences radio des points d'accès atteignables ainsi obtenu est associé au nom de la pièce ou du lieu (tel que le numéro de la salle ou le nom de l'endroit dans le bâtiment). Pour chaque endroit dépisté *k*, nous avons enregistré un vecteur du signal avec des valeurs et une étiquette correspondant à la pièce où est situé l'endroit dépisté.

$$\left(ap_k^1, ap_k^2, ..., ap_k^n, room_k\right)$$

Le modèle est représenté en format Prédictive Modèle Mark-up Langage (PMML).

Au cours de la phase de *prédiction*, le module WiFi de prédiction utilise ce modèle pour prédire les emplacements en temps réel. Pour chacune des salles impliquées, notre module de prédiction calcule la probabilité des valeurs observées dans la pièce, puis sélectionne la pièce ayant la plus grande probabilité. Soit $P(R_j)$ la probabilité des valeurs données qui sont observées depuis la salle R_j, l'expression pour sélectionner la salle R est donnée par :

$$R = R_k : P(R_k) = \underset{j=1}{\overset{m}{Max}} P(R_j)$$

4.2 Une architecture de la plateforme CoCA

En informatique pervasive, une plateforme de service réactive au contexte devrait avoir comme objectif l'acquisition et l'utilisation du contexte et pour fournir des services appropriés sans la supervision de l'utilisateur. Donc, nous proposons une plateforme de service collaborative et réactive au contexte (nommé: la plateforme CoCA). La plateforme CoCA est composée de cinq couches (layer) : *capturing layer*, *pre-processing layer*, *management modeling layer*, *context-aware core service*, et *application layer* qui exécute les actions. La Figure 13 montre l'architecture en couche de la plateforme CoCA.

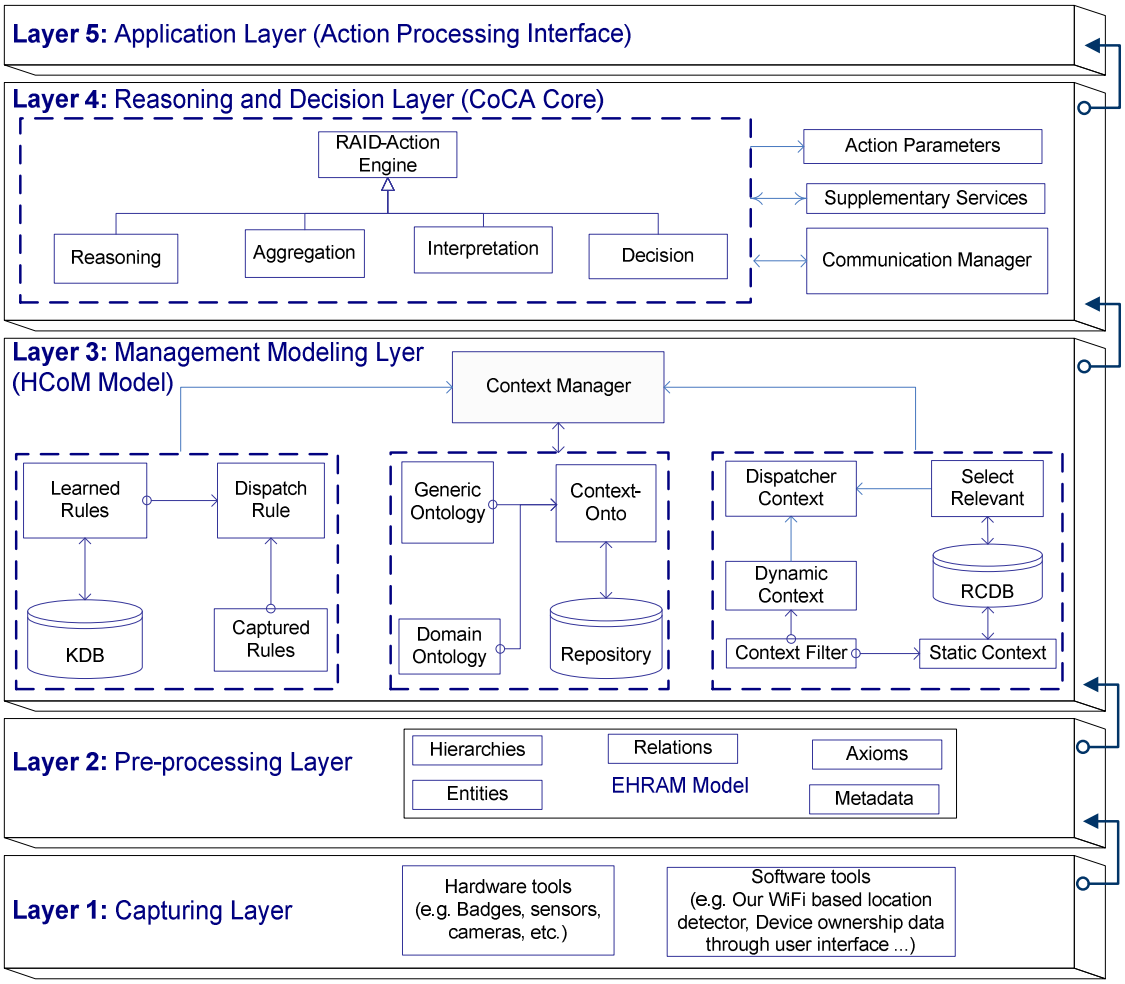


Figure 13 : Architecture de CoCA

Layer 1 : La couche d'acquisition du contexte qui contient les instruments de capture et/ou les logiciels de prédiction.

Layer 2 : La couche de modélisation et de formalisation de la représentation des données de contexte capturées. Elle sépare les données contextuelles pour regrouper les entités, les hiérarchies, les relations, les axiomes et les métadonnées (EHRAM).

Layer 3 : La couche de modélisation de gestion du contexte. Elle traite la façon dont nous organisons les ressources du contexte utiles pour le raisonnement. Une représentation formelle de cette couche peut être observée sur le modèle HCoM.

Layer 4: La couche de CoCA de base. Elle est l'endroit où le raisonnement final de la réactivité au contexte et les décisions sont exécutées. Elle est composée du moteur RAID-Action (résonnement, agrégation, interprétation, décision et action) qui peuple l'ontologie de données de contextuelles et applique ensuite les règles et axiomes de raisonnement et de décision sur les actions à déclencher. Le service supplémentaire dans cette couche est composé d'éléments externe au service de CoCA de base. Ceci inclut des services comme la découverte de connaissances, la confidentialité, la sécurité, etc.

Layer 5 : La couche d'application. Cette couche dépend du domaine d'application où sont déclenchées des actions de façon réactive ou proactive. Il accueille également des

processus de déclenchement de l'action en fonction de l'application spécifique du domaine dans lequel la plateforme est utilisée.

4.3 Collaboration dans CoCA

La technologie de soutien JXTA est un ensemble de protocoles ouverts permettant à tout périphérique connecté au réseau allant des téléphones cellulaires et assistants numériques personnels sans fil à des ordinateurs, des serveurs et des super ordinateurs de communiquer et de collaborer pair-à-pair.

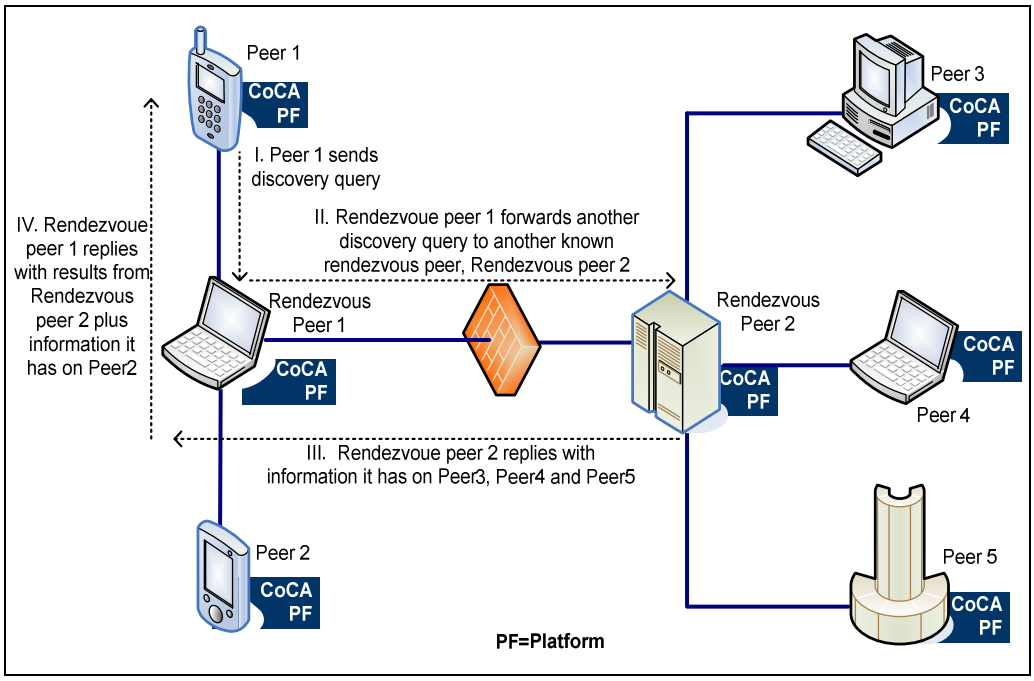


Figure 14: Principes de la collaboration et de la découverte dans CoCA

Le rôle de la gestion de collaboration dans l'espace du voisinage de la plateforme CoCA est de partager des ressources informatiques comme les contextes, les règles et les ontologies. Les principales exigences de base pour le service collaboratif entre le voisinage du CoCA sont la capacité de s'organiser en groupes de pairs, de s'entre découvrir et de découvrir les services et les ressources de chacun. Ce principe est illustré sur la Figure 14.

Un algorithme pour les processus impliqués dans la gestion de la collaboration dans la plateforme CoCA est indiqué sur la Figure 15. L'algorithme indique les étapes suivre pour découvrir les pairs et établir un lien pair-à-pair entre les modules de collaboration dans les deux pairs selon notre architecture basée sur JXTA. Le schéma a cinq partitions verticales indiquant le type et la catégorie des pairs impliqués dans le processus de collaboration.

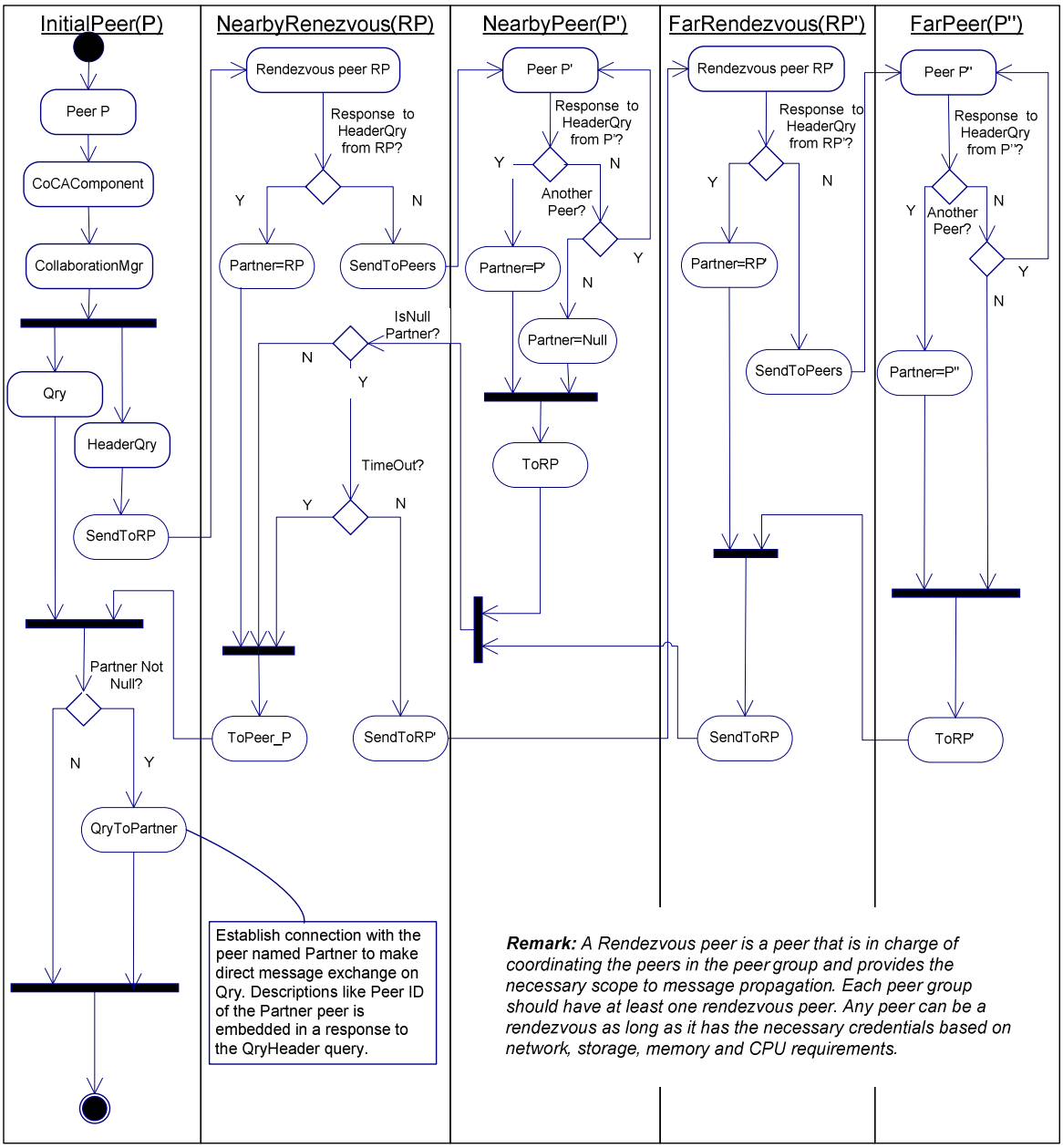


Figure 15: Algorithme de collaboration dans CoCA

4.4 Un scénario du cas d'utilisation

Nous utilisons un scénario de PiCASO (*Pervasive Campus-Aware Smart Onlooker)*. PiCASO est basé sur le scénario d'un campus universitaire où sont impliqués des étudiants chercheurs et des enseignants-chercheurs. Outre le calendrier des réunions régulières prévues entre les étudiants et les professeurs, des réunions et des discussions informelles et spontanées sont importantes pour l'avancement de leur travail. Une discussion peut avoir lieu entre deux ou plusieurs des chercheurs selon la pertinence de leur travail.

Les questions qui peuvent être soulevées dans ce scénario sont : Quand est-ce qu'ils vont avoir une telle réunion ? Comment informer uniquement ceux qui sont disponibles à propos des champs d'intérêt de quelqu'un d'autre pour discuter sur des thèmes spécifique pendant la

pause café ? Quel est le type de transmission de messages approprié pour envoyer des informations à une personne donnée située à un endroit et à un moment particulier? Si le téléphone est utilisé pour recevoir un tel message, quel devrait être son mode d'appel ou d'avertissement (vibration, sonnerie,)? et ainsi de suite.

La mise en œuvre du scénario de PiCASO utilise un nombre de contexte créé par la combinaison du contexte et les relations définies dans un domaine du campus. (variant de 200 à 6000 instances). De telles instances sont stockées dans la base de données relationnelle du contexte, RCDB. Ensuite, seules les données contextuelles pertinentes sont chargées de la RCDB vers l'espace du *reasoner* pendant l'initialisation. Il utilise également des règles (variant de 20 à 200 lignes) créées à partir des politiques de domaine et des besoins de l'usager. En combinant tous ces facteurs avec le schéma de l'ontologie de PiCASO, Le modèle HCoM produit jusqu'à 9000 triples de contexte dans l'ensemble de l'espace de raisonnement.

4.5 Démonstration du raisonnement dans CoCA

Étant donné l'ontologie de PiCASO avec ses données de contexte et certaines règles de auto-messagerie enter les usagers de PiCASO, nous allons essayer de montrer comment l'occurrence d'un simple contexte déclenche l'envoi d'un message (i.e. une action).

Figure 16 est un segment d'ontologie de PiCASO montrant la relation *advisorOf* qui existe entre les trois professeurs (*Eve*, *Professor_11*, *Dave*) et les cinq étudiants (*Carol*, *Bob*, *Student*_23, *Alice*, *Student*_21). Certains de ces nœuds dans le graphique indiquent les relations actuellement existantes. Par exemple, le nœud du professeur Dave montre les relations *OfferClass*, *AdvisorOf*, *engagedIn*, *hasOffice* et *ownerOf* et leurs valeurs. Le terme *isa* (is a) dans le graphe représente la relation *rdfs:subClassOf*, et également le terme *io* (instance of) représente la relation *rdf:type*. Dans l'ontologie, les relations *AdvisorOf* et *studentOf* sont définies comme des inverses mutuels. Cela signifie que chaque fois que la relation *advisorOf* existe, il est également vrai pour *studentOf* dans le sens inverse.

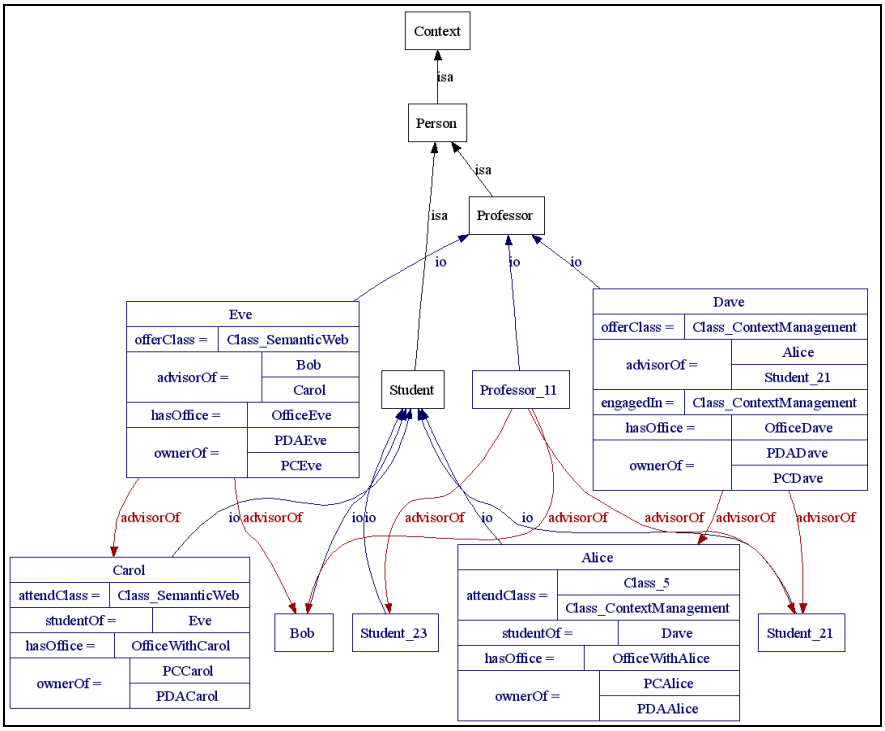


Figure 16 : L'ontologie du contexte du PiCASO montrant la relation advisorOf

```
# Send trigger message
[InformRule1:
   (?S rdf:type pre:Student)
   (?P rdf:type pre:Professor)
   (?S pre:studentOf ?P)
   (?S pre:locatedWith ?P)
->(?S pre:hasMessageTogo
      pre:ProfessorHere)
]
[InformRule3:
   (?S1 rdf:type pre:Student)
   (?S2 rdf:type pre:Student)
   (?P rdf:type pre:Professor)
   (?S1 pre:studentOf ?P)
   (?S2 pre:studentOf ?P)
   (?S1 pre:locatedWith ?S2)
   notEqual(?S1,?S2)
->(?S1 pre:hasMessageTogo
      pre:ColleagueHere)
```

```
#Set mobile ringing tone
[MobilePhoneRule1:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:Library)
  ->(?d pre:setNotificationMode
  pre:SilentMode)
]
[MobilePhoneRule2:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:ClassRoom)
  ->(?d pre:setNotificationMode
  pre:VibratingMode)
]
```

Figure 17 : Quelques règles de PiCASO pour déclencher des messages et des sonneries

La colonne 1 de la figure 17 porte sur les règles qui régissent le message de déclencheurs. Si on regarde la première règle, elle indique que « si l'ensemble des contextes capturés sont agrégés pour nous donner un nouveau contexte indiquant la présence d'un étudiant et de son professeur au même endroit, alors on transforme la valeur de *MessageToGo* de l'étudiant en un message nommé *ProfessorHere* ». Le processus d'agrégation du contexte, par exemple, vérifie l'emplacement du professeur et de l'étudiant et décide s'ils sont situés au même endroit ou non. Le message *ProfessorHere* par lui même

est un objet dans lequel sont définis une valeur, un module d'exécution, etc.. Dans ce cas particulier, le contenu du message est "Envoyer à: Etudiant --> Votre professeur est là!" Nous pouvons avoir plusieurs politiques et règles comme celles-ci qui sont basées sur le domaine d'application. Un autre ensemble de règles que nous utilisons dans cette démonstration (colonne 2 dans la figure) est celui qui est destiné à la gestion de la sonnerie du téléphone portable.

HCoM est alors utilisé pour créer un modèle de raisonnement basé sur la mémoire qui peut être interrogé en utilisant le langage de requête SPARQL pour l'action finale.

La plate-forme CoCA fournit une interface pour l'exécuter et tester les décisions et les actions de ce type. Nous avons exécuté et vérifié que toutes les mesures attendues sont également exécutées de la même manière.

4.6 Mesure de la performance

Dans cette section, nous utilisons les résultats du scénario de PiCASO pour mesurer la performance et l'évolutivité de la plateforme CoCA et du modèle HCoM. Notre expérience vise à démontrer comment l'utilisation du modèle l'HCoM garantit l'évolutivité et l'extensibilité du processus du raisonnement. Le raisonnement basé sur la machine est un processus qui consomme beaucoup de temps et qui a une complexité temporelle exponentielle à l'égard de la taille des données dans l'espace de raisonnement [Zuo, 2006]. Une expérience sur notre prototype est réalisée afin d'évaluer l'évolutivité de notre processus de raisonnement concernant le nombre total de triplets dans l'espace du raisonnement.

Nous avons fait de multiples expériences avec de données de taille variée pour l'évaluation de la performance des deux approches, GCoM et HCoM (le modèle GCoM est le prédécesseur du modèle HCoM qui utilise l'ontologie uniquement pour la gestion de contexte). Le résultat montre que la performance du raisonnement dépend de la taille de raisonnement des triples et de la complexité des règles de raisonnement. L'utilisation du modèle GCoM produit une représentation graphique du temps qui croît sans cesse avec l'augmentation du volume et de la complexité de la ressource de raisonnement. D'autre part, l'utilisation de HCoM donne un graphique qui tend à rester constant avec la croissance du volume et la complexité de la ressource de contexte. La figure 18 montre le résultat de notre expérience en utilisant à la fois les approches GCoM et HCoM. Une expérimentation similaire faite par [Wang, 2004] sur leur contexte ontologique CONON et illustrée par la figure 19, montre l'évolutivité du raisonnement du contexte basée sur l'ontologie qui donne un graphique plus ou moins semblable au notre GCoM.

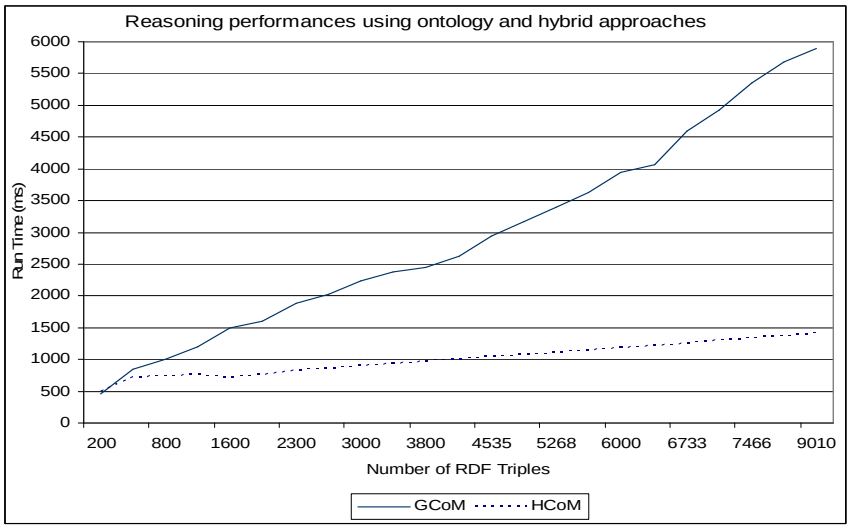


Figure 18: Résultats des mesures de GCoM et HCoM (2x1.83 GHz CPU)

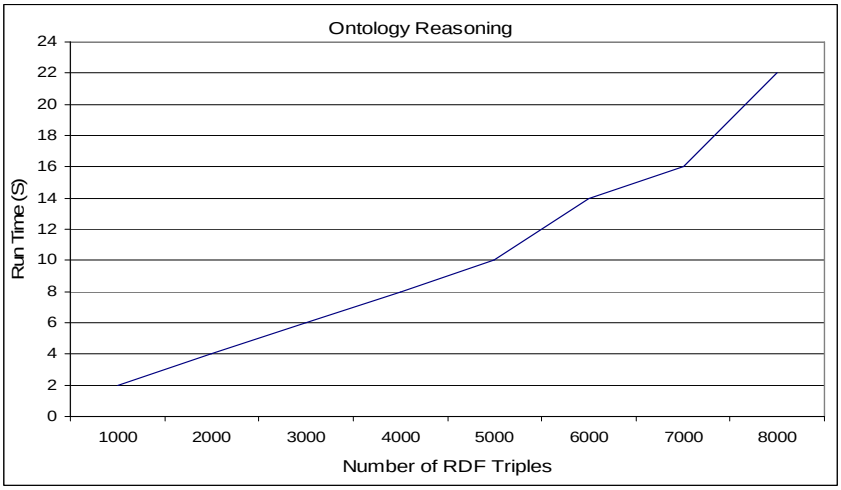


Figure 19: Performance de raisonnement pour CONON (2,4 GHz CPU)

Notre expérimentation montre que l'utilisation de HCoM donne un temps de réponse relativement constant à l'augmentation de la taille des données. En conclusion, le modèle HCoM et son constructeur, le modèle EHRAM, améliorent les performances du service CoCA et en font une plateforme évolutive et extensible.

5. Conclusions et travaux futurs

Dans ce travail, nous avons présenté HCoM, notre nouveau modèle de gestion, d'agrégation et de présentation des données du contexte en utilisant EHRAM, notre modèle de représentation du contexte. Dans HCoM nous utilisons une approche hybride où le schéma de l'ontologie, le contexte et les données de règles sont stockés et traités séparément avant qu'ils soient combinés et présentés pour faire le raisonnement. Seules les données pertinentes sont sélectionnées et chargées dans le schéma de l'ontologie du contexte à partir de RCDB (Relational Context Database). Nous avons également présenté la plateforme CoCA, un intergiciel collaboratif indépendant des données du contexte qui donne un

service de réactivité au contexte basé sur les modèles EHRAM et HCoM dan un environnement pervasif.

La validation des modèles et de la plateforme est effectuée en utilisant un exemple de démonstration sur le scénario d'un campus universitaire, PiCASO. Nous avons également testé CoCA avec des données d'un hôpital sur le scénario de suivi du patient et des services, et avec des données d'adaptation de logiciel d'application sur les propriétés de divers types de dispositifs tels que la taille de l'écran, la taille de la mémoire, la capacité d'affichage, la vitesse du processeur, et la vitesse de connexion. Les résultats de notre expérimentation montrent que les modèles EHRAM / HCoM sont extensibles et évolutifs à la gestion de contexte de taille variable, et que la plateforme CoCA est un intergiciel générique, collaboratif et indépendant de données.

La gestion de l'incertitude, de la sécurité et de la confidentialité, et la tolérance aux pannes sont parmi les travaux futurs envisagés. Le processus de raisonnement de l'être humain est basé sur l'abondance d'incertitudes environnementales. Dans la plupart des cas, l'erreur de capteur (granularité inhérente et/ou erreurs de lecture), les données périmées, et les mauvaises prédictions donneront l'incertitude lors de la capture du contexte. Nous avons besoin de certaines méthodes pour s'occuper de ce problème d'incertitude avant d'utiliser des données de contexte. L'utilisation des métadonnées composante du modèle EHRAM peut être un point de départ pour la réalisation d'un modèle de gestion de contexte plus robuste et probabiliste.

Le système informatique pervasive réactive au contexte doit faire face à des défis de sécurité liés à la confidentialité, à l'intégrité et à la confiance. Nous devons avoir des services génériques qui gèrent ces problèmes de sécurité dans un environnement pervasif hétérogène. L'aspect collaboratif de la plateforme CoCA expose naturellement le système à des pairs défectueux et malveillants. Un domaine de recherche possible consisterait donc à incorporer la tolérance de défaut et les mécanismes d'auto-guérison. Le travail peut commencer par l'identification et le marquage des pairs malveillants afin de les prendre en considération lors prochaine participations, la préparation d'alternatives multiples de sorte qu'une défaillance de communication ne puisse bloquer le fonctionnement du système, etc.

Chapter 1 INTRODUCTION

1.1 Background

Today's growth in the advancement of computing applications accompanies the evolution of distributed middleware. In e-commerce and cooperative business, the Web and its underlying protocols (HTTP, SOAP, FTP …) are becoming the standard execution platform for distributed and component-based applications. The increasing number of computers and users on the Internet has led not only to cooperation structures such as peer-to-peer computing that has great potential for scalability but also stimulated new developments in the area of computing clusters called grid computing. The integration of mobile clients into a distributed environment and the ad-hoc networking of dynamic components are becoming ever more important in all areas of application. [Mattern03] indicated that, given the continuing technical progress in computing and communication, we are heading towards an all-encompassing use of networks and computing power named as *ubiquitous or pervasive computing.*

1.1.1 Pervasive computing

According to Dan Russell, director of the User Sciences and Experience Group at IBM's Almaden Research Center, by 2010 computing will have become so naturalized within the environment that people will not even realize that they are using computers. In the future, *smart* devices all around us will maintain information about their locations, the contexts in which they are being used, and relevant data about their users.

According to Matteren et al, the vision of pervasive computing is grounded in the firm belief amongst the scientific community that Moore's Law, the observation that "*the computer power available on a chip doubles every two years"* will hold true atleast for the foreseeable future. This means that in the next few years, microprocessors will become so small and inexpensive that they can be embedded in almost everything – not only electrical devices, cars, household appliances, toys, and tools, but also in ordinary things such as pencils and clothes. These devices will be interwoven and connected together by wireless networks.

Portable and wireless appliances are already a hot topic in today's computing industry. Personal data assistances (PDAs), smart phones and global positioning systems (GPSs) are only the first precursors of new devices and services that will emerge. This leads towards the full realization of Mark Weiser's vision that states, "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable," [Weiser91].

Pervasive computing systems target at constantly adapting their behavior in order to meet the needs of users within every changing physical, social, computing and communication context. Pervasive devices make ad-hoc connection among them and may be connected to different types of sensors to capture changes in the environment.

1.1.2 Context-Aware computing

As a result of research initiatives at IBM [IBM01] and subsequent research interests and activities, autonomic computing is becoming one of the top challenging IT research areas. According to [Hariri06], *autonomic computing is inspired by the human autonomic nervous system that handles complexity and uncertainties, and aims at realizing computing systems and applications capable of managing themselves with minimum human intervention*. Both works underline that context awareness is one of the key challenging component of the autonomic computing paradigm.

Figure 1.1 shows the flow in the evolution chain from centralized computing to pervasive computing as presented by [Satyanarayanan01] and [Strang04]. This classification places *context awareness* at the heart of pervasive computing problems. The complexity of such problems increases in multiplicative (⊗) fashion rather than additive (⊕), with the addition of new components into the chain.

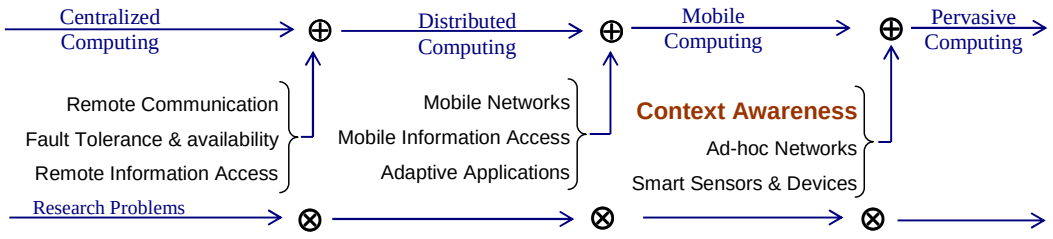


Figure 1-1: Context awareness in the computing evolution chain

The term context-aware computing was formally defined and used for the first time by [Schilit94] to describe applications that *"adapt according to their location of use, the collection of nearby people and objects, as well as the changes to those objects over time"*. [Dey00] define context-aware computing as *"a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*. [Burrell02] gave another definition stating *"context-aware computing is the use of environmental characteristics such as the user's location, time, identity and activity to inform the computing device so that it may provide information to the user that is relevant to the current context."*

From these definitions and our observation about this new computing dimension, context-aware computing for us is an environment in which applications can discover and take advantage of contextual information such as location, time, people, devices, and user activity. Context awareness in pervasive computing environment needs to take account not only of the proactive and collaborative aspect of the services but also user inconveniences due to services that are not adapted to user's regular duties that may create frustrations and resistance to subscribe to the service.

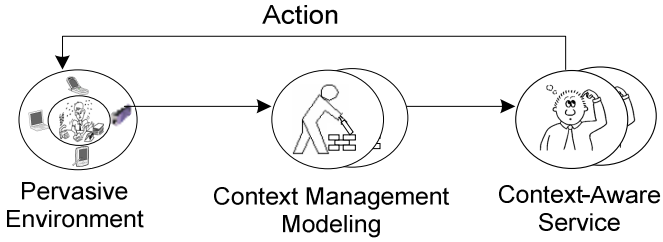


Figure 1-2: Conceptual framework of a pervasive context-aware computing

Our conceptual framework showing major components of a pervasive context-aware system is shown in Figure 1.2. The framework has three basic components: pervasive environment, context management modeling and context-aware service. *Pervasive environment* is characterized by dynamicity, heterogeneity and ubiquity of users, devices and other computing resources, ad-hoc connection among the devices and existence of hardware and software sensors. *Context management modeling* deals with how context data is collected, organized, represented, stored and presented to the reasoning module. *Context-aware* service performs context reasoning and decisions about the actions to be triggered.

1.1.3 **Elements of pervasive context-aware computing**

Among the basic elements of services that context-aware systems provide to users are:

- *Context triggered action:* for example, if the user is in a meeting and there is a phone call for him or her, then the call should be routed to voicemail. Triggers can also be placed on certain events, to make the context-aware system take actions such as notifying an administrator about an increase in temperature in a server room.
- M*ulti-facet command processing:* commands issued by the user can produce different results depending upon the context in which they were issued. A web browse command issued by a user responds according to the context of the user and his environment (for example, resources available in the device: memory, screen size and resolution).
- *Location based proximity:* given information about the location of the user and the situation of the user, objects located nearby can be given high priority. Such prioritization would be particularly useful when using small pervasive devices which might have limited resources. For example, a nearby printer would be proposed when the user wishes to print something.
- *Metadata tagging:* context information can be attached to existing pieces of information to give descriptive information about the subject. This context acts as metadata about both physical and virtual objects in our system. For example, when a user records an audio clip on a handheld device, the system can attach the current context information (date and time, people present, current activity) to the clip for easy retrieval and indexing.
- *Collaborative computing:* voluntary based mission-oriented context-aware and dynamic communities of computing entities that perform tasks on behalf of users in an autonomous manner. Such applications are important in almost every sphere of our lives, such as campus management, health care, telemedicine, pervasive security, military, and crisis management.

A context-aware system must be capable of mimicking human's ability to recognize and exploit implicit information in the environment. Although identifying and deducing a human activity is a challenge, it is critical that context-aware applications should operate by conveying the appropriate information to the right place at the right time by inferring the user's intention. To accomplish this objective the context-aware systems must:

- Gather the information from the environment or the user's situation.
- Translate this information into the appropriate format.

- Combine or interpret context information to generate a higher context. A higher context is context information that is derived as a result of the merger of other context information or it is context information that results from interpretation of a low level context (e.g. conversion of geographic coordinates of a location received from satellite based positioning systems into street names).
- Automatically trigger actions based on the context information and monitoring of the actions.
- Make the information accessible to other applications and the neighborhood. The context management model in pervasive computing environment should handle context in a reusable manner to permit context from one source to be exploited by many distinct applications and devices in the neighborhood space that perform a variety of tasks.

1.2 Motivations

Dynamic adaptation of application to a changing environment leads to an enhancement of the user satisfaction. However, how application programmers can effectively manage and use contextual information typically in pervasive environments is still a challenge. Most of the current context-aware systems are based on ad-hoc models of context, which causes a lack of formality and expressiveness.

Most of the computing devices in pervasive computing environment are handheld or wearable, tiny and resource hungry devices. This calls for collaboration among these devices. The growth of enabling open protocols that allow connected devices on the network to communicate gives rise to the popularity of collaborative computing. Grid computing, for example, is one of the successful efforts to handle controlled large-scale collaborative computing. Collaborative software agents have a potential to become intelligent enough to observe us and learn our habits and preferences to serve us better. The new dimension of collaboration in pervasive computing envisages the use of peer-to-peer and ad-hoc networking of devices. On the other hand, the growth of number of computing devices that interfere with our daily activities in our environment may be frustrating if they are not properly adapted to our situations and if they all require our attention. These devices, after all, are meant to simplify life and improve our working conditions.

To achieve these objectives, we need to embed some sort of intelligence (context awareness) into these devices so that their service is adapted to our situation (context). Below are outlines of three examples of motivating scenarios for the pervasive context-

aware applications: the *smart hospital scenario*, *smart campus scenario*, and the *adaptation of applications scenario*. Detailed descriptions and a demonstration version of the implementation of these scenarios are given in chapter 6.

Smart hospital scenario - patient monitoring and follow up: Consider a smart medical ward in a hospital where patients, nurses and physicians, etc. are involved. Assume that the ward is equipped with context sensor technologies (hardware and software) in its rooms, corridors and garden at the disposal of individuals involved. Patients admitted to the hospital may need intensive follow up which may create staff shortage and may result in inappropriate care to the needy ones due to overloading. CoCA based context-aware monitoring and follow up system helps to minimize the engagement of human assistants to the less important activities. Human interventions may be needed only when alerted by the system. Live multimedia recording and transmission of an event that the system has found important may also be used for monitoring purposes. Adapted and personalized delivery can be made to those who are concerned.

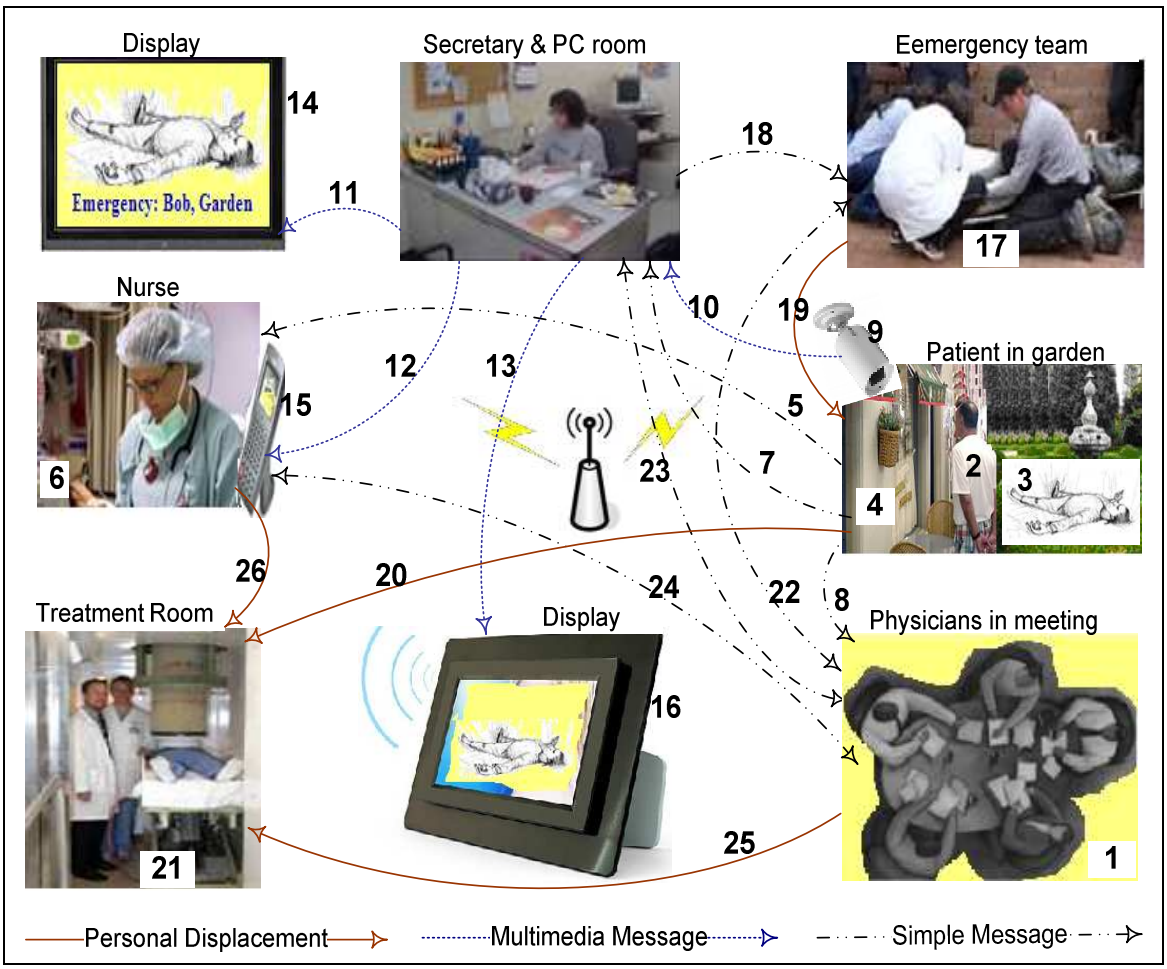


Figure 1-3: Smart Hospital use-case scenario

A particular case of the scenario can be described as follows:

Dr. Pascal is a physician who is assigned to work in the smart medical ward. Ada is a nurse assigned to assist Pascal. Michel is one of the inpatients of Pascal and Ada is in charge of Michel. Instead of assigning a permanent human assistant to take care of Michel whose medical condition is rated improving, he is equipped with wearable devices that record his body conditions like temperature, blood pressure, heart rate, movement pattern, etc. There is also a necessary badge in the form of, say, bracelet for communication and position tracking. Figure 1.3 illustrates a specific use-case scenario of a patient monitoring and follow-up service.

Here is the story about a specific scenario of a patient monitoring and follow-up service in a smart hospital that is illustrated in the figure: *One morning, Dr. Pascal and his colleagues are in a weekly consultation meeting (1). Michel is in a garden to enjoying the morning sun (2). He, however, suddenly feels exhausted and falls down (3). The wearable and the badge he carries on his body immediately deliver all the necessary information to the nearby computing device (4). The system sends (5) an alert message to Ada (6), the nurse, on her smart phone. Another message to central PC room in the office of the medical secretary for administrative and rediffusion purposes. Knowing also of Dr. Pascal's schedule, from his agenda, that he is in the meeting, the system informs (8) him through SMS message on his cell phone that flashes red light when receiving such type of emergency message when he is in a meeting. The camera facing the garden (9) where Michel is located is activated and his picture is sent (10) to the central PC room for adaptation and broadcast (11, 12, 13) to all concerned terminals (14, 15, 16). Emergency workers (17) are also informed (18) about the situation. As a result, Michel is taken (19, 20) to the treatment room (21) by the emergency workers. Dr. Pascal, who has already been aware of his patient's current situation, has made all the necessary consultation (22, 23, 24) and preparations for appropriate medication and is already in the treatment room (25). Ada is also in the treatment room (26) to provide all the necessary aid to the patient.*

Smart campus scenario – Pervasive Campus Aware Smart Onlooker: PiCASO is based on the scenario of a university campus where research students and professors are involved. Besides the scheduled regular meetings among students and professors, informal and spontaneous meetings and discussions are important for the advancement of their work. Discussion can take place among two or more of the researchers depending on the relevance

of their work. Proactive context-aware services can be proposed to answer questions like: When do they make such a meeting? How can only those available are informed about someone else's interest to discuss about a specific subject matter during his tea break? How can a student know that his professor is available for the coming 30 minutes? How can a student know when his professor is in the tea room and is available, in his office or in the corridor passing by the office of the student? What type of messaging method is appropriate to send such information to a particular person located at a particular place at a particular time? If telephone is used to accept such a message, what should its call mode be (vibrating, ringing)? Other similar location, activity and device aware services can also be made available.

Adaptation of applications scenario - adaptation of applications to context: Adaptation of software applications to context is one aspect of context-aware computing in pervasive environment. For example, if a display unit of a device doesn't support images, and if the user selects to view an image on this device, the application should automatically lead the user to the textual description of the image which means the displayImage() module in the application must be locked by the context-aware service.

As indicated in the related work (chapter 2), existing works on context-aware computing are limited either to the development of middleware support for context-aware system development or on the modeling of context in an infrastructure-supported environment. Therefore, to our knowledge, there is no comprehensive context management model and middleware built based on collaboration of peer devices.

Hence, this work is *motivated* by two main observations we have made regarding pervasive context-aware computing environment: the need for a generic context management model and the need for a platform support for pervasive context-aware system development. Our approaches try to solve the problem by proposing a peer-to-peer based collaborative context-aware service platform and a hybrid context-management model that insures scalability, formality and reusability.

1.3 Research problems

Context-aware computing in pervasive environment involves a set of functionalities that need separate attention. We compile and discuss the major problem areas in the following sections.

1.3.1 **Context acquisition and management**

Below are some of the major challenges related to context acquisition and management.

Context Sensing: is a mechanism to obtain the context data from diverse context sources. For example, the indoor location of a user can be obtained from an infrared location sensor system, which detects the presence of a badge to conclude the location of the user wearing the badge.

Context Modeling: Existing context models vary in the expressiveness they support and the types of context they represent. Context modeling deals with an abstraction that acquires context data and then annotates them with semantics that are structured around a set of contextual entities (e.g., *user, location, device*) and relations that hold among them (e.g., *user locatedIn location, device ownedBy User*). It is important to have a standardized context model in order to facilitate context interpretation, context sharing and semantic interoperability.

Context Repository: There are two principal approaches of context storage and use; centralized and shared (distributed). A centralized context repository can provide a persistent storage for distributed context sources and guarantees integrity of context. It relieves context-aware services from overheads caused by querying from distributed sensors. In additional to the traditional problems of centralized data repositories like single point of failure, this approach suffers from dependency on infrastructures. On the other hand, a distributed and shared context repository approach gives autonomy and full control of context resources by the individual partners and allows mobility. When context is represented based on shared context model, context repositories should provide a foundation to merge interrelated information (semantic interoperability) and enables further data interpretation. An optimized approach for storing both data and its semantics under the pervasive computing setup has to be worked out.

1.3.2 **Context semantics and reasoning**

Semantics according to [Nielson95] refers to aspects of meaning. Where meaning, in its brief form, is the content carried by the words or signs exchanged between entities while communicating. For our purpose, context semantics, therefore, is the aspect of meaning of

context for further reasoning and decision. Below are some of the major challenges related to context semantics and reasoning.

Context Query: To explore general means of access to interrelated context spread across distributed context repositories, we need a high-level mechanism for context -aware services to issue queries. For example, a notification service for conference attendees require context query like "F*ind a list of researchers in this hall whose publications are in the same session with mine*". The low-level operations of such a complex context-retrieval task should not be exposed to end users. Context query poses design issues such as context query language, event notification, and query optimization.

Context Aggregation: Atomic context is simple, low-level context, directly provided by a context source and composite context is a high-level context that aggregates multiple atomic or composite contexts. In most cases, software services cannot directly understand and utilize low-level information. Hence, upon detecting that certain context is atomic, a context aggregator will retrieve meta-information from the repository about the specific context providing a related composite context. For example, a location based service wants to know the relative location with different levels of granularity (e.g., room-level, building-level, campus-level) instead of sensor-driven position (the position coordinates retrieved by the sensors). In this case, we need to derive high-level location (e.*g.*, *the building in which the user is located*) from location coordinates and other related contexts.

Context Reasoning/Inferring: The context interpretation layer leverages reasoning/learning techniques to deduce high-level, implicit context needed by intelligent services from related low-level, explicit context. For example, the rule-based reasoning engine can deduce user's current situation based on his location and environmental contexts. The inferred context might suggest that the user might be sleeping currently, since the time is 11 pm and he is staying at a dark and quiet bedroom. Another example of context interpretation could be machine learning based behavior prediction. The intelligent system could learn the pattern of user's actions from historical sequences of context data and then use this learned pattern to predict next event. For example, it could be predicted that once the user finished showering (turn off the electronic water heater) after 10:30 pm, he will check emails using the hand phone, and then go to bed after finishing reading them. Currently, context interpretation tasks are performed through various approaches including

ad-hoc interpretation, rule based reasoning, and machine learning. We need an approach that combines context data and semantics into reasoning and inference service.

Uncertainty: In most cases, sensor error (inherent granularity and/or false readings), out of date data and poor predictions will give rise to some uncertainty about sensed context. We need some means of handling this uncertainty problem before using the context data.

1.3.3 **Context-aware system development support**

Middleware Support: The increased availability of commercial, off-the-shelf, sensing technologies is making it more practical to sense context in a variety of environments. The prevalence of powerful, networked computers makes it possible to use these technologies and distribute the context to multiple applications, in a somewhat pervasive fashion. A major problem has been a lack of uniform support for building and executing these types of applications. Most context-aware applications have been built in an *ad-hoc* manner, heavily influenced by the domain of application and the underlying technology used to acquire the context. This results in a lack of generality, requiring each new application to be built from scratch. To enable application developers and designers to more easily build context-aware applications and to help them concentrate on the application aspect rather than the context acquisition and management details, there needs to be a middleware architectural or platform support that provides the general mechanisms required for context management and reasoning.

1.3.4 **Collaboration and security**

Context Discovery/Delivery: Pervasive computing environment is full of small devices with scarce resource that need a peer-to-peer collaboration. In order for a context-aware service to use a certain kind of context, there is a need for context requestors to find the sources providing it. The aim of context discovery is to locate and access context sources. Issues of context discovery include service description, advertisement and event subscription. Context delivery services perform the job of searching appropriate context and delivering them to the applications. These include registration, query and notification services. Interested peer applications query the registration service to find services of their interests. The registration service upon finding appropriate context source, returns the handler to the requesting clients.

Security and privacy: Due to the inherent need of collaboration in pervasive context-aware systems, they face security challenges in the form of privacy, integrity and trust [Hong04]. Privacy of context information focuses on protecting context resources from unauthorized entities. For example, a user should be able to protect personal information such as his/her health status, or medical history. Integrity of context information focuses on guaranteeing that the provided context information has not been corrupted by a third party. For example, temperature measurements delivered by thermometer must be reliable and not modified by any entity. Trust issue in collaborative context-aware computing has to also deal with a respect for common security policy and common goal.

1.4 Scopes and contributions

In this thesis, we will focus on the context-aware aspect of pervasive computing and special emphasize will be given to context data management and their use in the development of collaborative context-aware applications. Proper modeling, specification and definition of context and its management are essential for efficient reasoning, interpretation, and utilization of context data. Our scope and contributions in this work can be presented in three categories: conceptual context representation model (EHRAM), hybrid context management model (HCoM) and a collaborative context-aware service platform (CoCA).

The context representation model (EHRAM) stands for entity, hierarchy, relations, axioms and metadata. EHRAM is based on hierarchy of descriptors of context entities. It is represented using a layered and directed graph. Hierarchies in EHRAM are important structures to organize and classify context entities and relations. Layered organization also helps to classify and tag context data as generic domain independent or as domain dependent. Entities like *activity, person, device, etc.* in the generic layer are common to all domains of applications. They are high level context entities from which sub context entities can be derived. Relations like *isEngagedIn(Person, Activity), owns(Person, Device) and isLocatedWith(Person, Person)* represent generic relations that can be inherited down in the hierarchy by the sub-entities and instances in the specific domain of application. Consider a medical application where context data comes from medical entities like patients, doctors, nurses, activities and events in the hospital, devices, locations, etc. Entities like *patient, doctor, meeting and phone* are specific to the medical domain of application. Among relations that can be defined in the medical domain are: *hasBodyTemp(Patient, tmp),*

hasDoctor(Patient, Doctor), hasBPlower(Patient, bpL), hasBPupper(Patient, bpH), etc. EHRAM can be easily serialized to standard markup languages for storage, retrieval, transmission and processing. Details about the EHRAM model are given in chapter 3.

The Hybrid Context Management model (HCoM) is a generic context management model based on a hybrid approach. Our rational behind the need for a hybrid context model is to distinguish the works of context data management and context semantic management, process them separately and put the results together for better reasoning and decision support in a context-aware environment. We use an ontology-based approach to manage context semantics and a relational approach to manage context data. HCoM model aims to combine the best from the two worlds. HCoM has a context selector module that provide a means to select and load only part of the large static context data that is accumulated over a period of time depending on who and where the user is, the intended activity to which the user is going to be engaged, devices available for use, institutional policies, etc. It uses matching patterns gained through experience to identify relevant context data. Loading only relevant data for reasoning minimizes the size of the reasoning space and reduces the unnecessary overloading of the reasoner. This improves the overall performance of the context-aware service. It helps to overcome limitations of scalability of most of the reasoning systems to the growing volume of reasoning resources collected and stored overtime. Details about the HCoM model are given in chapter 4.

The collaborative Context-Aware service platform (CoCA) is a neighborhood based middleware in pervasive computing that aims at acquiring and utilizing context information to provide appropriate services. For example, a cell phone is always set to vibrating mode when its holder is in the library if it has the knowledge about his current location. The reasoning engine in the platform accepts a set of context data, domain ontology, rules and their semantics and changes it into concrete knowledge necessary for reasoning and decisions. Applications then perform actions accordingly. Details about CoCA platform are given in chapter 5.

Finally, we implement a demonstration version of the proposed CoCA platform. We then evaluate its performances using data from different scenarios.

1.5 Structure of the thesis

This thesis presents our approach to solving the problems of context modeling, management and utilization under the pervasive computing paradigm.

In chapter 2, we present the state of the art and related works both in the areas of context management modeling and development of pervasive context-aware systems. This chapter also presents the state of the art on some of the baseline technologies deemed necessary for our work. Chapter 3 is about the EHRAM model. EHRAM deals with definition of context and its representation. Context entities, their hierarchy, their relationships, related axioms and metadata are considered as basic building blocks for the definition and modeling of context data.

In chapter 4, we present HCoM model. HCoM deals with the pre-processing and management aspect of context data. We introduce a semantically rich novel approach for context management modeling. It uses the hybrid of ontology and database principles for modeling the management of both context data and context semantics. Chapter 5 is about CoCA service platform. CoCA is a multi-domain context-aware middleware platform that transforms context knowledge into appropriate action depending on the domain of application in which it is used. It uses reasoning tools and implements collaboration protocols for pervasive devices in order to share context related resources. Implementation and evaluation of the proposed solution is given in chapter 6.

Summery of contributions, conclusions and highlights of future works are given in chapter 7. Finally, we present chapters on glossary of acronyms and bibliography that is followed by the annexes. Annexes provide excerpts of resources used for the implementation of PiCASO in the CoCA platform.

Chapter 2 STATE OF THE ART IN CONTEXT-AWARE COMPUTING

2.1 Introduction

Context-aware computing systems have been introduced and discussed ever since the Olivetti Active Badge project [Want92]. It is then followed by other works like [Shilit94] who gave the first formal definition of the term context-aware (section 1.1.2). Context-aware computing has also been presented as a key feature in different projects over the last decade. According to [Mattern03], many works have been done so far that demonstrate the importance of context awareness in pervasive computing. Earlier efforts focused on the development of application specific systems that use only few types of context information mainly identity, time and location. Since then a number of location-aware systems have been designed for city tours and museum guides. Among the traditional examples of such applications is a context-aware tour-guide that provides city visitors with information tailored to their preferences and environment.

Among such early developments are:

- *ParcTab [Schilit93]:* Based on the Active Badge system, the ParcTab is a mobile computing system that provides different services (active map, location of people within a building, schedule management) based on the user's location and time.
- *Cyberguide [Abowd97]:* Provides information to a tourist based on knowledge of position and orientation by using a position-aware handheld tour guide. The tourist can find directions, retrieve information, and leave comments on an interactive map. The system uses the user's location to make suggestions on places of interest to visit. The location information is collected by GPS outdoors, and by an infrared positioning system indoors.
- *Project CoolTown at HP Research Lab [Kindberg01]:* Is a location aware system that ties web resources to physical objects and places, and allows users to interact with these resources using the information devices they carry.
- *The cricket Compass [Priyanta01]:* Reports the position and orientation indoors, for a handheld mobile device, and informs an application running on the device about the position and orientation in a local coordinate system established beforehand. It uses

fixed active beacons and passive ultrasonic sensors. It proposes an alternative to GPS systems for locating entities indoors.

- *MyCampus [Sadeh02]:* Is a semantic web environment for context-aware services. The environment revolves around a growing collection of customizable agents capable of discovering and accessing Intranet and Internet services as they assist their users in carrying out different tasks within a campus. This allows the agents to automatically access and exploits relevant user preferences and context information.

Other early works include: [Rekimoto95], [Broadbent97], [Pascoe97], [Cheverst00], [Kindberg00], EasyLiving [Brumitt00], [Woodruff01], [Espinoza01], [Feiner02], Project Oxygen at MIT Media Labs Dertouzos99] and [Rudolph01], Project Aura at CMU [Garlan02] and Project BlueSpace at IBM Labs [Lai02].

These works indicate that context-aware computing can be a reality if proper context capturing and management techniques are put in place. [Dey00] have proposed location, time, identity, and activity as the primary elements of context. Certain aspects of context such as time and location are easily detected, however others, such as activity are much more difficult to capture.

In this chapter, we try to look at some of the related works in the area of context management modeling and that of context-aware system development support and services. We will also look at the highlights of available tools and protocols for context reasoning and collaboration.

2.2 Related works in context management modeling

Recently, various context management and modeling approaches have been introduced to support standardization of techniques to present context for productive reasoning in different application area. Among the major classifications of context management modeling approaches are *Key-Value-Pair modeling, Graphical modeling, Object oriented modeling, logic based modeling, Markup scheme modeling and Ontology modeling.*

Key-Value-Pair modeling is the simplest category of the models. They are not very efficient for sophisticated and structuring purposes but support only exact matching and no inheritance. Graphical modeling is particularly useful for structuring, but usually not used on instance level. Examples include UML [Eriksson04] and ORM [Halpin07]. Object oriented modeling has a strong encapsulation and reusability feature. Examples include,

Cues (TEA project) [Gellersen00] and Active Object Model (GUIDE project) [Davis99]. Logic based modeling uses logic expressions to define conditions on which a concluding expression or fact may be derived from a set of other expressions or facts. Context is defined as facts, expressions and rules and has a high degree of formality. Examples include McCarthy's Formalizing Context [McCarthy98] and Akman&Surav's Extended Situation Theory [Akman96]. Markup scheme modeling uses standard markup languages or their extensions to represent context data. Details on markup scheme are given in section 2.2.2. Ontology based models use ontology and related tools to represent context data and its semantics. Details discussions about this approach are given in section 2.2.3.

2.2.1 **An overview of context modeling approaches**

Context representation and management modeling is an important aspect of pervasive computing. Because context-aware applications must adapt to changing situations, they need a detailed model of users' activities and entities in the surroundings that lets them share users' perceptions of the real world. These entities may have different meanings associated with them in different environments. In order to have similar meanings of these entities, when used at different times, in different situations, by different applications, their semantics should be formalized. This allows us to store context data for future use and to communicate context universally with other systems. One of the basic steps in the development of context-aware applications is, therefore, to provide formalized representation and standardized access mechanisms to context information.

Earlier approaches to context management modeling were based on *context widgets, networked services and blackboard models*.

Context Widgets [Dey01] are derived from techniques in GUI development GUI is a software component that provides a public interface for a hardware sensor. The main objective of the widget is to separate the application from the context acquisition process. Widgets hide low-level details of sensing and eases application development due to their reusability. Widgets are usually controlled by some kind of a widget manager.

Networked services [Hong01] use the data source-discovery techniques instead of a global widget manager to find networked services.

Blackboard model [Winograd01], in contrast to the process-centric view of the widget and the service-oriented view of networked models, represents a data-centric view. In this asymmetric approach, processes post messages to a shared media, the so-called blackboard, and subscribe to it to be notified when some specified events occur. Advantages of this model are the simplicity of adding new context sources and the easy configuration.

According to [Baldauf07], the tightly coupled widget approach increases efficiency but is not robust to component failure. Baldauf et al also indicated that the networked service based approach is not as efficient as widget architecture due to complex network based components but provides robustness, and the blackboard models need a centralized server to host the blackboard that presents a single point of failure and lacks communication efficiency as two hops per communication are needed.

2.2.2 Context models that use markup scheme approaches

Markup languages are standard encoding systems that consist of a set of symbols inserted in a document to control its structure, formatting, or the relationship among its parts [Britanica07]. Markup languages establish the "vocabulary," "grammar" and “syntax" of the codes applicable to text, image, or other form of data within an electronic document. The most widely used markup languages are SGML, HTML, and XML. SGML served as the foundation for HTML and XML. HTML is used for rendering the document, and XML is used for identifying the content of the document. Mark-up symbols can be interpreted by devices or computing entities (computer, printer, browser, etc) A mark-up document thus contains data to be processed and a mark-up language on how to process it.

Context models in this category are commonly used for profile data representation. Some examples of such models are given in the following sections.

2.2.2.1 CCML

Centaurus Capability Markup Language (CCML) [Kagal02] is divided into *system*, *data*, *add-ons*, *interfaces*, and *info components.* The *system* portion contains the header information, the *id*, timestamp, origin, etc. There are two variables, *update* and *command*. All information regarding the variables and their types are contained in the *data* section. Using the *add-ons* section, one can add a related service to another service; for example, add an Alarm Clock service to a Lamp Control service. The CCML for a client always has one

or more actions in its *data* section that a Service Manager can invoke on it. The *interface* section contains information about the interfaces that the object (Service/Client) implements. This section generally causes the variables in the *data* section to change their values. Other details like description are contained in the *info* section.

2.2.2.2 CSCP

Comprehensive Structured Context Profiles (CSCP), by [Held02] and latter by [Buchholz04], presented profiles that are based on resource description framework (RDF) for representation and manipulation of context data. CSCP does not define any fixed hierarchy. It rather supports the full flexibility of RDF/S to express natural structures of profile information as required for contextual information. Attribute names are interpreted context sensitively according to their position in the profile structure. Hence, unambiguous attribute naming across the whole profile is not required.

2.2.2.3 CC/PP

Composite Capabilities/Preference Profile (CC/PP) [Indulska03] and [CC/PP04] describes structures and vocabularies. A CC/PP profile is a description of device capabilities and user preferences. This is often referred to as device's delivery context and can be used to guide the adaptation of content to be presented to that device. The Resource Description Framework (RDF) is used to create profiles that describe user agent capabilities and preferences. Client capability and preference descriptions use RDF classes to distinguish different elements of a profile, so that a schema-aware RDF processor can handle CC/PP profiles embedded in other XML document types.

2.2.2.4 CDF

Context Description Framework (CDF) [Khriyenko05] is a Framework for the Semantic Web. CDF extends RDF with capabilities to model highly dynamic and context-sensitive information. It also extends RDF statements logically and defines quadruple statements compared to RDF triples. A CDF quadruple is a Statement that inherits from the *rdf:Statement* and has an additional *cdfs:trueInContext* property from the namespace of the CDF-Schema Vocabulary Description Language which is an extension of RDF-Schema. The first three components of a CDF quadruple are similar to a RDF triple of subject, predicate and object. Subjects and objects of CDF statements are instances of *rdfs:Resource*

and CDF predicate is an instance of *cdfs:Property*. The fourth component of a CDF quadruple is the context of the statement.

2.2.3 Ontology based context models

Ontology is commonly used as explicit specification of a shared conceptualization. Context is modeled as concepts and facts using ontology. Some examples of systems that that use this approach are discussed in the following sections.

2.2.3.1 CONON

CONON [Wang04] present an OWL based context Ontology for reasoning and representation of contexts in pervasive environments. CONON (stands for CONtext Ontology) is based on the treatment of high-level implicit contexts that are derived from low-level explicit contexts. It is designed to be used in pervasive computing environments to enable context modeling and logic based context reasoning. It supports interoperability of different devices. CONON defines generic concepts regarding context and provides extensibility for adding domain specific concepts. Logic reasoning is used in order to perform consistency checks and to calculate high-level context knowledge from explicitly given low-level context information. The authors also present the results of a performance study which in order to evaluate the feasibility of logic based context reasoning in pervasive computing environments. This is especially important because in these environments computational resources such as processing power and memory are often limited.

CONON consists of an upper ontology which is extended by several domain specific ontologies for intelligent environments such as “home”, “office” or “vehicle”. The upper ontology holds general concepts which are common to the sub domains and can therefore be flexibly extended. CONON is implemented by using OWL. The authors conclude that context reasoning in pervasive environment is time-consuming but is still feasible for non-time-critical applications. For time-critical applications such as navigation systems or security systems, the data size and rule complexity must be reduced.

Concerning the overall architecture, the authors recommend decoupling the context reasoning from the context usage: a strong server performs the reasoning while small devices such as mobile phones receive the pre-calculated high-level context from the server for direct use. This requires infrastructure based environment.

2.2.3.2 CoBrA-ONT

CoBrA-ONT [Chen04f] is a context management model that enables distributed agents to control the access to their personal information in a context-aware environment. CoBrA-ONT is a collection of OWL ontologies for context-aware systems. CoBrA-ONT is designed to be used as a common vocabulary in order to overcome the obstacle of proprietary context models that hinder the interoperability of different devices. Furthermore, the semantics of OWL are used for context reasoning.

CoBrA-ONT is central part of CoBrA, a "broker-centric agent architecture in smart spaces" where it supports context reasoning and interoperability as mentioned above. The center of this architecture is context broker agent, which is a server that runs on a resource-rich stationary computer. It receives and manages context knowledge for a set of agents and devices in its vicinity, which is the "smart space". Agents and devices can contact the context broker and exchange information by the FIPA Agent Communication Language.

The architecture of CoBrA-ONT is based on the upper ontology and the domain specific ontology that extends the upper ontology. CoBrA-ONT is defined using the Web Ontology Language (OWL) to model the concepts of people, agents, places and presentation events. It also describes the properties and relationships between these concepts. CoBrA-ONT depends on the assumption that there always exists a context-broker server that is known by all the participants.

2.2.3.3 SOUPA

Standard Ontology for Ubiquitous and Pervasive Applications SOUPA [Chen04e] (the same authors of CoBrA-ONT) is designed to model and support pervasive computing applications. The SOUPA ontology is expressed using the Web Ontology Language OWL and includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, events, user profiles, actions, and policies for security and privacy. SOUPA is more comprehensive than CoBrA-ONT because it deals with more areas of pervasive computing. It also addresses CoBrA-ONT's problems regarding ontology reuse. The SOUPA sub-ontologies map many of its concepts using *owl:equivalentClass* to concepts of existing common ontologies.

2.2.3.4 GAS ontology

GAS ontology [Christopoulou04] is ontology designed for the collaboration among ubiquitous computing devices. The basic goal of this ontology is to provide a common language for the communication and collaboration among the heterogeneous devices that constitute these environments. The GAS Ontology also supports the service discovery mechanism that an ubiquitous computing environment requires.

2.2.4 **Summary**

According to [Strang04], ubiquitous computing systems make high demands on context modeling approach in terms of the following requirements:

Distributed composition (dc): Any ubiquitous computing system is a derivative of a distributed computing system which lacks of a central instance being responsible for the creation, deployment and maintenance of data and services, in particular context descriptions. Instead, composition and administration of a context model and its data varies with notably high dynamics in terms of time, network topology and source.

Partial validation (pv): It is highly desirable to be able to partially validate contextual knowledge on structure as well as on instance level against a context model in use even if there is no single place or point in time where the contextual knowledge is available on one node as a result of distributed composition. This is particularly important because of the complexity of contextual interrelationships, which make any modeling intention error-prone.

Richness and quality of information (qua): The quality of a information delivered by a sensor varies over time, as well as the richness of information provided by different kinds of sensors characterizing an entity in an ubiquitous computing environment may differ. Thus a context model appropriate for usage in ubiquitous computing should inherently support quality and richness indication.

Incompleteness and ambiguity (inc): The set of contextual information available at any point in time characterizing relevant entities in ubiquitous computing environments is usually incomplete and/or ambiguous, in particular if this information is gathered from sensor networks. This should be covered by the model, for instance by interpolation of incomplete data on the instance level.

Level of formality (for): It is always a challenge to describe contextual facts and interrelationships in a *precise* and *traceable* manner. For instance, to perform the task "print document on a printer near to me", it is required to have a precise definition of terms used in the task, for instance what "near" means to "me". It is highly desirable, that each participating party in a ubiquitous computing interaction shares the same interpretation of the data exchanged and the meaning "behind" it (so called *shared understanding*).

Applicability to existing environments (app): From the implementation perspective it is important that a context model must be applicable within the existing infrastructure of ubiquitous computing environments, e.g. a service framework such as Web Services.

Table 2-1: Summary of appropriateness of modeling approaches

Requirements	Approaches				
	Markup Scheme	Graphical models	OO models	Logic based	Ontology based
Distributed composition	+	-	++	++	++
Partial validation	++	-	+	-	++
Quality of information	-	+	+	-	+
Incompleteness/ambiguity	-	-	+	-	+
Level of formality	+	+	+	++	++
Applicability	++	+	+	-	+
(Key: ++ Comprehensive + Partial - Limited or none)					

Based on a survey made on systems from each category of context modeling approaches using the above requirements as a reference, Strang et al summarizes the appropriateness of these approaches for pervasive computing. The summary result given in Table 2.1, by Strang et al shows that the most promising assets for context modeling for ubiquitous computing environments can be found in the ontology modeling category. In this work, we will therefore continue to investigate the use of ontology for semantic context management modeling in a further detail.

As a conclusion, context representation and management modeling has been introduced as a key feature in context-aware computing. Different data centric context management approaches have been proposed and used in context representation, storage and management. Most of them lack standardization and are developed as a proprietary model

for specific domain of application. In this work, we propose a comprehensive data independent ontology based semantically rich and hybrid context-management model that insures scalability and reusability of context resources and reasoning axioms and rules.

2.3 Related works in context-aware computing services

2.3.1 Overview of context-aware computing services

Early context-aware systems were relatively simple and were often constructed simply as distributed application components communicating directly with local or remote sensors. [Henricksen05a] argue that today, additional infrastructural components are desirable in order to reduce the complexity of context-aware applications, improve maintainability, and promote reuse. Such components that can be found in many current context-aware systems as shown in Figure 2.1, by Henricksen et al, are discussed below.

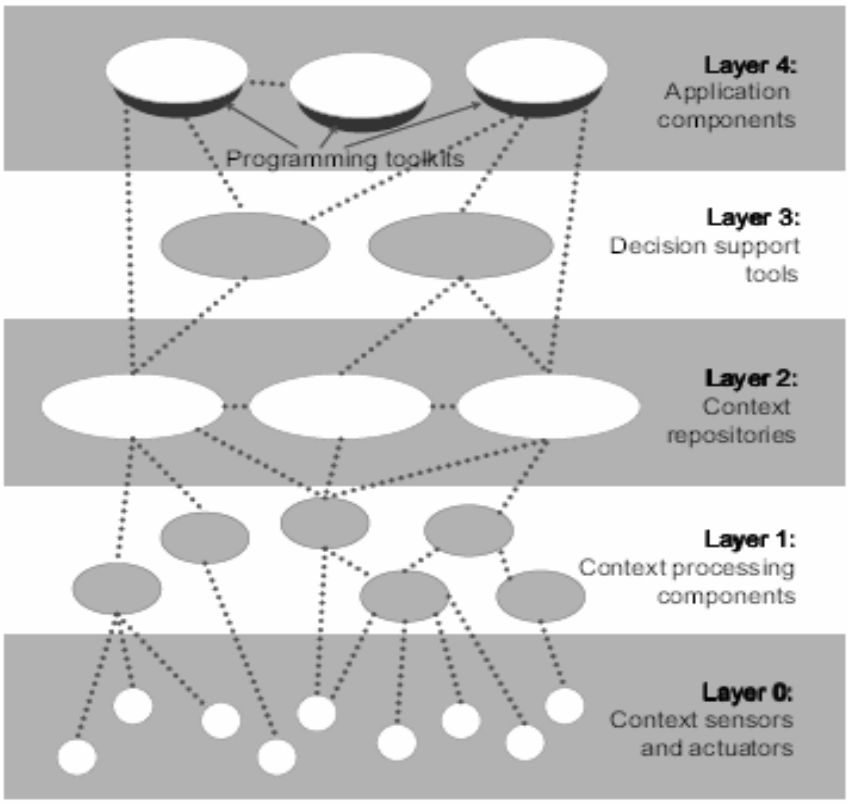


Figure 2-1: Components of Context-Aware Systems

- *Context sensors:* Numerous hardware devices would be equipped with the capability to collect information that will form part of the context of the system. These devices should be relatively inexpensive and readily available and, should hopefully, require a minimum amount of configuration and management. They must also be capable of

transmitting information to some central location, or else communicating with nearby devices.

- *Context Processing:* A model for context management and processing must be formulated to provide a resource for applications thereby enabling them to dynamically respond to changes in context such as network connectivity, user location, sound and lighting conditions, etc.
- *Context repositories:* Context-Aware systems must have a component that provides persistent storage of context information and efficient query facilities.
- *Decision support:* Decisions support tools may involve reasoning and aggregations. This component is responsible for decisions on actions triggers and other related services.
- *Application support*: Application programmers need to be aware of context information, as the addition of this information will fundamentally change how they work. Instead of being driven primarily by explicit user input as they have been in the past, applications will begin to 'act by them selves', proactively, but must do so in a way that attempts to adhere to the law of least surprise for the user. User interaction with a context-aware application should also be simpler and more productive than with a traditional application [Cheverst00].

2.3.2 Agent, blackboard and widget based context-aware systems

Context-aware systems in this category tend to use agents, blackboard or widgets in the middleware to insulate applications from the hassle of context acquisition and processing. In this section, we try to look at some of the recently developed agent, blackboard or widget based context-aware systems or projects we have identified to have close relations with our work.

2.3.2.1 Context Toolkit

The Context Toolkit [Dey01] aims at facilitating the development and deployment of context aware applications. The context information reflects an application's operating environment that can be sensed by the application. The Context Toolkit consists of context widgets and a distributed infrastructure that hosts the widgets. Context widgets are software components that provide applications with access to context information while hiding the details of context sensing. In the same way GUI widgets insulate applications from some presentation concerns, context widgets insulate applications from context-acquisition concerns.

Among the services of the Context Toolkit are:

- Encapsulation of sensors
- Access to context data through a network API
- Abstraction of context data through interpreters
- Sharing of context data through a distributed infrastructure
- Storage of context data, including history
- Basic access control for privacy protection

Context aggregators can be thought of as meta-widgets, taking on all capabilities of widgets, providing the ability to aggregate context information of real-world entities such as users or places. A context interpreter is used to abstract or interpret low-level context information into higher-level information. For example, identity, location, and sound level information could be used to interpret that a meeting is taking place. There is no elaborate implementation of intelligence; instead, a user specifies high-level concepts that are based on raw sensor data and environment information.

2.3.2.2 CMF

The CMF [Korpipaa03] context framework has four major components (Figure 2.2): context manager, resource server, context recognition service, and application. When entities communicate, the context manager functions as a central server while other entities act as clients and use services the server provides. The context manager, any resource servers, and applications run on the mobile device itself, and the services are either distributed or local. At the heart of the mobile terminal is the blackboard-based context manager.

A blackboard-based system has a central server, the blackboard, where all the participating nodes post their information, and the blackboard then processes this information, derives inferences and informs the participants about them. The blackboard node stores context information from any source available to the terminal and serves it to clients in three ways:

- Clients can directly query the manager (as a context database) to gain context data.
- Clients can subscribe to various context changes notification services.

- Clients can use higher-level (composite) contexts transparently, where the context manager contacts the required recognition services.

One interesting distinction from other context aware frameworks is that CMF uses special mechanisms, specifically fuzzy logic, to build high-level concepts from uncertain and fuzzy sensor data. Most other context-aware frameworks assume that the context is intelligently specified in terms of high-level concepts, but as this work points out, this is not a simple and straightforward task. They propose using fuzzy logic to reason about high-level concepts like location.

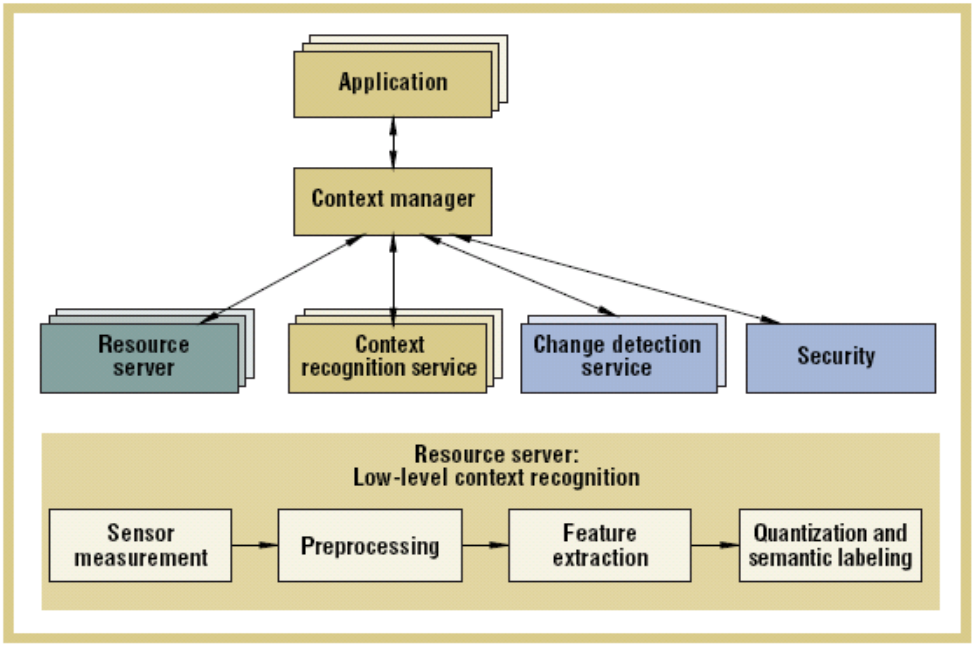


Figure 2-2: CMF Architecture

2.3.2.3 ACAI

ACAI [Kheder05] is an infrastructure that allows context information to be collected, processed, inferred, and disseminated to spontaneous applications. It provides this interaction seamlessly, without revealing the inherent complexity required to manage the heterogeneous sources that provide the context information.

This is achieved through a layered architecture, as shown in Figure 2.3. In the first layer, *the sensing layer*, context is sensed and captured through sources embedded in the environment. In the second layer, *the context service layer*, the context is interpreted and structured by the context ontology module. The context inference module deduces other contextual information that has not been explicitly sensed by the first layer. However, ACAI

provides additional functionalities such as service discovery that provides awareness about services in the environment. An example might be a printing service to accommodate the user's location and printing preferences. It provides context management so that context can be stored, queried and accessed by users and resources with limited capabilities. It also provides a context-sensitive communication protocol for event notification, service and information delivery, and presence projection. These value-added functionalities are passed to mobile users and applications in the third layer. This layer functions primarily as an interface to the lower layer, allowing context information to be negotiated with the context providers, and for context level agreements to be made between different domains. The third layer, *the application layer*, provides the interface between mobile users and applications on the one hand, and the ACAI context service functionalities on the other.

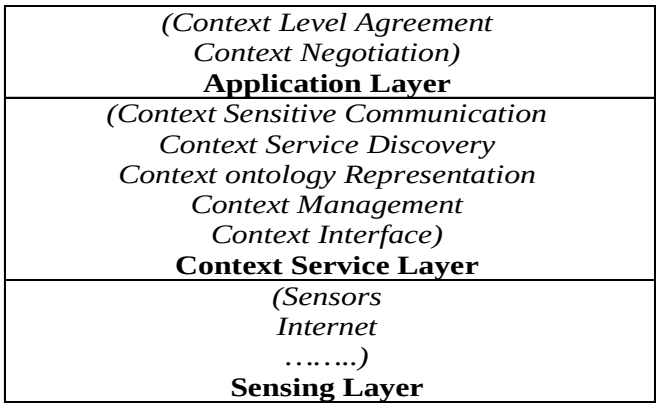


Figure 2-3: The ACAI layered architecture

ACAI is built on agents namely: the Context management agent (CMA), the Coordinator agent (CA), the Ontology agent (OA), the Reasoner agent (RA), the System knowledge base agent (SKBA), and the Context provider agent (CPA).

ACAI does not provide any mechanism for relevant context nor adaptation action description. Adaptation is always carried out at the application level.

2.3.2.4 CAMidO

CAMidO [Behlouli06] is a Context-Aware Middleware Based on Ontology Meta-Model. CAMidO provides ontology meta-model for context description and application adaptation that allows context and relevant context description. All the described data are compiled using the CAMidO compiler for generating adaptation source code and rule files for relevant context detection. Relevant context is associated to an adaptation policy. Then

the container is used to apply this adaptation if a relevant context is detected. The proposed middleware takes into account two types of adaptations, reactive and proactive adaptations. These adaptations are carried out by the component container to which new controllers were added.

CAMidO architecture, Figure 2.4, assumes three basic layers: context sensor layer, middleware layer and application layer.

- *The CollectionManager* is in charge of collecting context information from sensors according to designer description. For each sensor, an agent which is an entity allowing interaction with a sensor type, is activated for collecting data from it. Each agent knows how to interact with the associated sensor by using the described information in the Sensor class. The collected data are transferred by the CollectionManager to the ContextAnalyser and the ContextInterpreter.
- *The ContextAnalyser* is responsible for filtering context information and determining relevant changes. Context filtering consists in detecting context changes by comparing the collected context value with its old value stored in the ContextRepository. If a relevant context occurs, the new context value is saved in the ContextRepository by the ContextAnalyser which notifies the subscribed component about this relevant context changes.

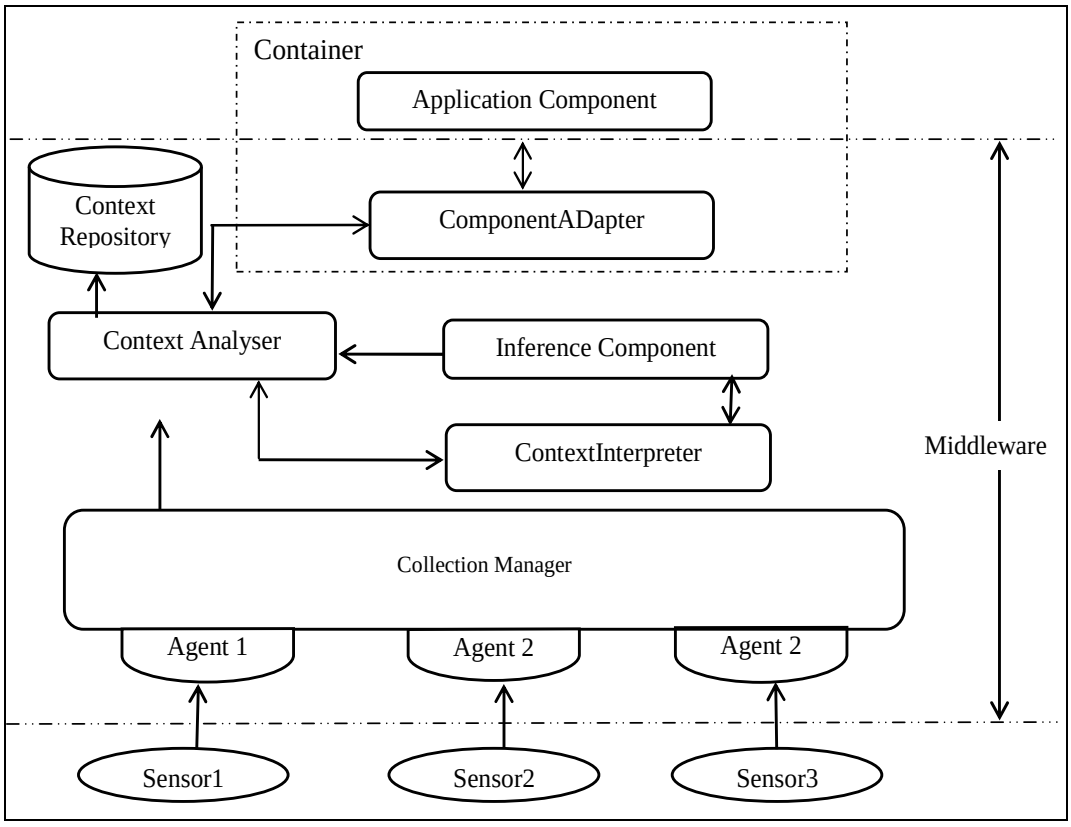


Figure 2-4: CAMidO architecture

Major components in CAMidO are:

- *The ContextInterpreter* has to deduce high-level context information by using the HowDeduce relation provided by CAMidO meta-model.
- *The ContextRepository* stores context information.
- *The ComponentAdapter* has to adapt an application component to context changes according to the adaptation rules defined by the application designer. It subscribes to the ContextAnalyser for relevant context in order to be notified when it occurs for applying adaptation policies. The ComponentAdapter belongs to the component container, it is made of many components working together in order to adapt application's component to context changes.

The described components are used by the CAMidO compiler to generate source code for application adaptation and Rule files, to be used by the ContextAnalyser and the ContextInterpreter, for detection of relevant context changes.

CAMidO is among the recent developments that considers an overall approach to context data management and processing. CAMidO is a good approach for proactive actions and adaptations. However, it is based on client-server paradigm and requires proxy service that requires further treatment to fit the service to the pervasive computing environment.

2.3.3 Broker middleware based context-aware systems

Context-aware systems in this category tend to use broker-based middleware to insulate applications from the hassle of context acquisition and processing. In this section, we try to look at some of the recently developed broker based context-aware systems or projects we have identified to have close relations with our work.

2.3.3.1 RCSM

Reconfigurable Context-Sensitive Middleware (RCSM) for context-aware applications [Yau02] is an object-oriented middleware that facilitates the development and runtime operation of context-sensitive software. Figure 2.5 shows RCSM's integrated components. The context-sensitive interface lists the contexts the applications use, a list of actions the applications provide, and a mapping between them that clearly indicates, based on specific context values, when an action should be completed. RCSM provides what is called the adaptive object container (ADC) for runtime context data acquisition.

The RCSM's Context-Aware Interface Description Language (CA-IDL) compiler generates a custom-made ADC tailored code for triggering application adaptation according to developer description that can be used to specify context requirements, including the types of context/situation that are relevant to the application, the actions to be triggered, and the timing of these actions.

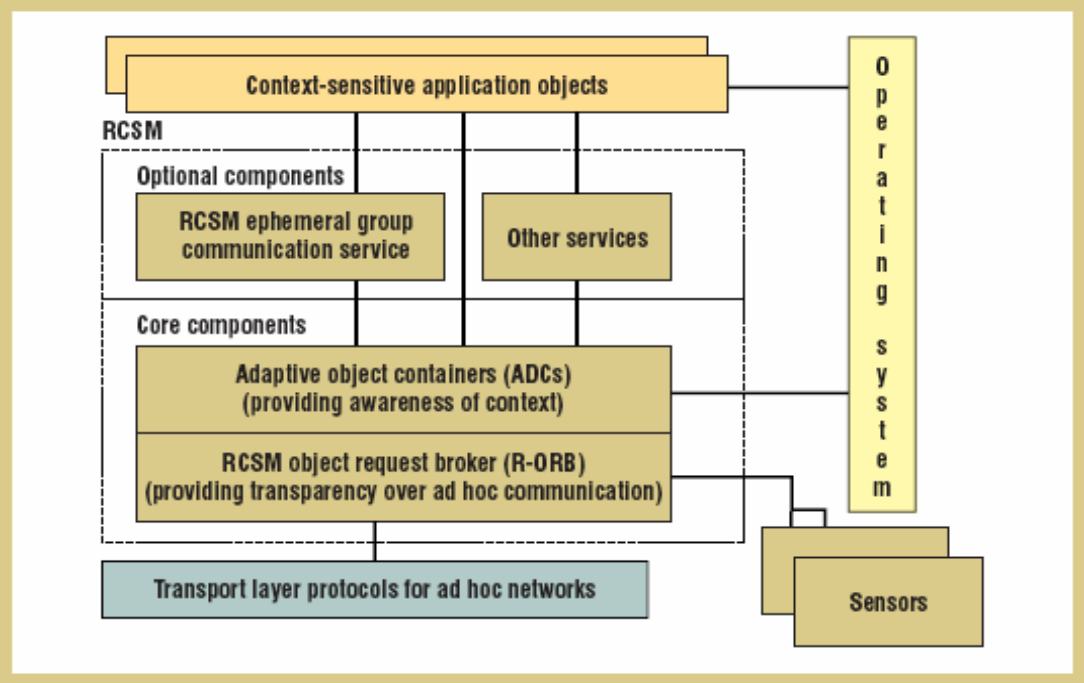


Figure 2-5: RCSM's integrated components.

The IDL interfaces are compiled to produce application skeletons; these interact at run-time with the RCSM Object Request Broker (R-ORB), which manages context acquisition, and the Situation-Awareness processor, which is responsible for managing triggers. The R-ORB provides a context manager that uses a context discovery protocol to manage registrations of local sensors and discover remote sensors. When a context-aware application starts up, the discovery protocol is used to look for local or remote sensors that satisfy the application's context requirements.

The main strength of the approach comes from the use of an IDL to specify context requirements. This makes it possible to incorporate new types of context and context-aware behavior by editing and recompiling IDL interfaces, and partially addresses ease of deployment and configuration. The different situational actions and the mapping between the actions and the context list needs to be stated, as RCSM has no clear method for composing context in a semantic way, nor to reason even if the context is composed. The

context discovery protocol is not flexible enough to support mobility or component failure, and it does not attempt to address scalability and privacy.

RCSM lacks the ability to separate context knowledge from context data, it does not support context inference and composition and it is not scalable to the ever increasing size of context resources.

2.3.3.2 Gaia

The Gaia project [Roma02] defines a meta-operating system that is used to manage ubiquitous computing environment. Gaia is designed to facilitate the construction of applications for smart spaces, such as smart homes and meeting rooms. It consists of a set of core services and a framework for building distributed context-aware applications. The system is built as a distributed object system with three major building blocks: the Gaia Kernel, the Gaia Application Framework, and the Applications. Gaia Kernel services support various forms of context-awareness, and include:

- Context service, which allows applications to find providers for the context information they require
- Event manager service, which monitors the entities entering and leaving a smart space (including people as well as hardware and software components)
- Space repository service, which maintains descriptions of hardware and software components, and
- Context file system, which associates files with relevant context information and dynamically constructs virtual directory hierarchies according to the current context.

The approach in Gaia is based on first-order logic. It is distributed by a client server architecture, which has similarities to the Context Toolkit architecture. Context providers (Widgets) collect various types of contexts and can be queried by context consumers (Applications). A context synthesizer (Aggregator) contains logic rules that form new contexts from existing ones. A context provider lookup service (Discoverer) is used by the consumer for finding the context provider able to produce contexts of an appropriate type. In the blackboard model, such a lookup service is not necessary. Context history is stored in a database, except for context synthesizers. The blackboard model, having central data storage, makes managing a context database more straightforward. Communication between distributed entities is done using CORBA. Components of the system can be distributed and discovered using the CORBA naming service and CORBA trading service.

The system has a smart-space infrastructure-oriented approach as opposed to mobile device-centric. As smart spaces are typically small, constrained environments, Gaia does not address scalability. Similarly, privacy is not addressed by any of the basic Gaia services. Gaia does not support a peer discovery mechanism between the applications and the context providers. Applications can only use the context providers available in the registry and this will introduce scalability issues when the number of application instances increases. Gaia does not support ontology-based context modeling and this limits the ability of applications using Gaia active space to advertise the kinds of context they are interested in.

2.3.3.3 CoBrA

A Context Broker Architecture (CoBrA) [Chen03] discusses architecture for supporting context-aware systems. CoBrA uses Semantic Web languages and tools for managing and sharing context information. The architecture has a core server entity called Context Broker, which has the following responsibilities: provide a centralized model of context, acquire contextual information, reason about contextual information that cannot be directly acquired from the sensors, detect and resolve inconsistent knowledge that is stored in the shared model of context, and protect user privacy. The design of CoBrA is aimed to support context-aware systems in smart spaces, and each smart space is assumed to have a designated central context broker (see Figure 2.6).

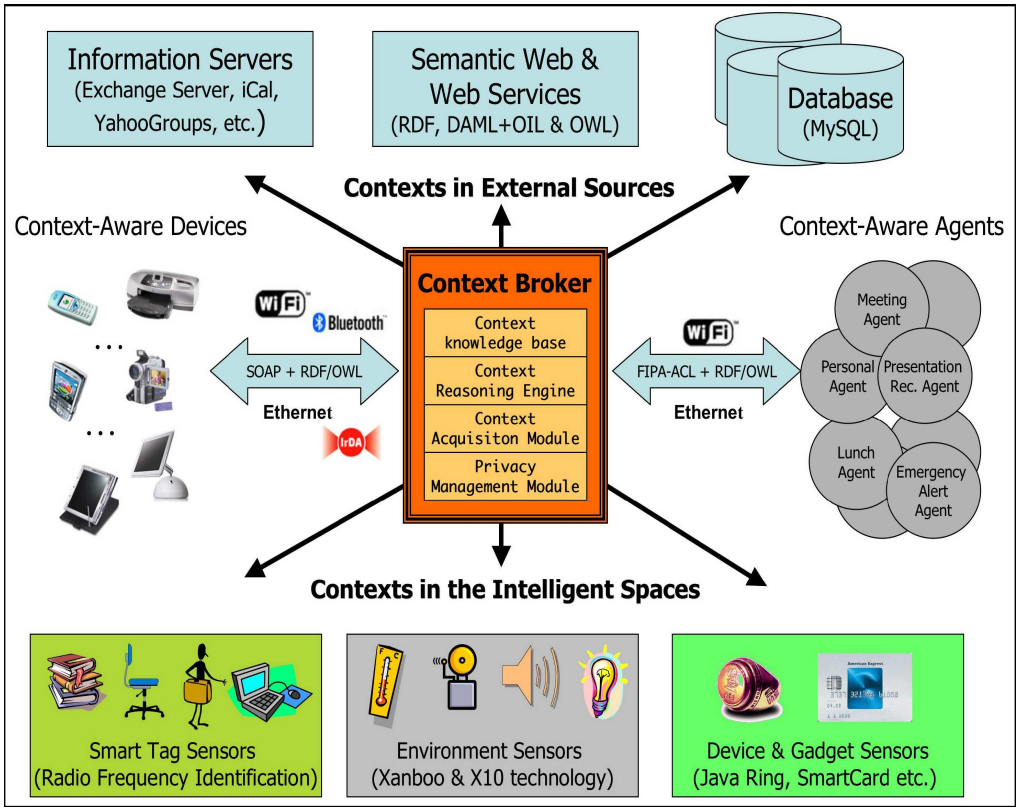


Figure 2-6: Context Broker Architecture

CoBrA is agent-based architecture for supporting context-aware computing in smart spaces that may include physical spaces like living rooms, vehicles, classrooms and meeting rooms that are populated with intelligent systems that provide pervasive computing services to users. Central to CoBrA is the presence of an intelligent context broker that maintains and manages a shared contextual model on the behalf of a community of agents. These agents can be applications hosted by mobile devices that a user carries or wears (e.g. cell phones, PDAs and headphones), services that are provided by devices in a room (e.g. projector service, light controller and room temperature controller) and web services that provide a web presence for people, places and things in the physical world (e.g. services keeping track of people's and objects' whereabouts). The context broker consists of four functional main components: the Context Knowledge Base, the Context Inference Engine, the Context Acquisition Module and the Privacy Management Module. To avoid the bottleneck problem CoBrA offers the possibility of creating broker federations. CoBrA provides a sample intelligent meeting scenario represented in OWL schema.

CoBrA is infrastructure-centric and is not fit for pervasive computing whereas the platform proposed in this work is mobile device-centric, where no additional equipment for a mobile device itself is required for system operation.

2.3.4 **Service oriented middleware based context-aware systems**

Context-aware systems in this category tend to use service oriented middleware layers to insulate applications from the hassle of context acquisition and processing. In this section, we try to look at some of the recently developed service oriented middleware based context-aware systems or projects we have identified to have close relations with our work.

2.3.4.1 CFNs

Context Fusion Networks (CFNs) [Chen04a] provides data fusion services (aggregation and interpretation of sensor data) to context- aware applications. CFNs are based on an operator graph model, in which context processing is specified by application developers in terms of sources, sinks and channels. In this model, sensors are represented by sources, and applications by sinks. Operators, which are responsible for data processing, act as both sources and sinks. They have implemented the CFN model in the form of Solar, a scalable peer-to-peer platform which instantiates the operator graphs at runtime on behalf of context-aware applications. The Solar implementation supports application and sensor mobility by buffering events during periods of disconnection; they also address component failures by providing monitoring and recovery, as well as preservation of component states. However, Solar does not yet address heterogeneity, privacy, or monitoring and control of the system by users.

2.3.4.2 ConFab

Context Fabric (Confab) proposed by [Hong04] is concerned with privacy rather than with context sensing and processing. Confab provides architecture for privacy-sensitive systems, as well as a set of privacy mechanisms that can be used by application developers. The architecture structures context information into infospaces, which store tuples about a given entity. Infospaces are populated by context sources such as sensors, and queried by context-aware applications. They have implemented the infospace model using Web technologies, such that infospaces are identified by URLs and tuples are exchanged in an XML format. Privacy can be supported by adding operators to an infospace to carry out actions when tuples enter or leave the space; for instance, operators can be used to perform access control, notify users of information disclosure, and enforce privacy tags that describe how information can be used after it flows from one infospace to another.

As Confab focuses so heavily on privacy, it does not address traditional distributed systems requirements such as mobility, scalability, component failures and deployment/configuration. However, it does partially address heterogeneity, as it builds on platform and language-independent Web standards. It also provides privacy-related traceability and control via the operator mechanism.

2.3.4.3 SOCAM

The SOCAM architecture [Gu05], shown in Figure 2.7, aims to provide an efficient infrastructure support for building context-aware services in pervasive computing environments. SOCAM is a distributed middleware that converts various physical spaces from which contexts are acquired into a semantic space where contexts can be shared and accessed by context-aware services. It consists of the following components, which act as independent service components:

- Context providers: they abstract useful contexts from heterogeneous sources-External or Internal; and convert them to OWL representations so that contexts can be shared and reused by other service components.
- Context interpreter: it provides logic-reasoning services to process context information. RDFS and forward chaining and backward chaining rule engines.
- Context database: it stores context ontology and past contexts for a sub-domain. There is one logic context database in each domain, i.e. home domain.
- Context aware services: they make use of different level of contexts and adapt the way they behave according to the current context.
- Service-locating service: it provides a mechanism where context providers and the context interpreter can advertise their presence; it also enables users or applications to locate these services.

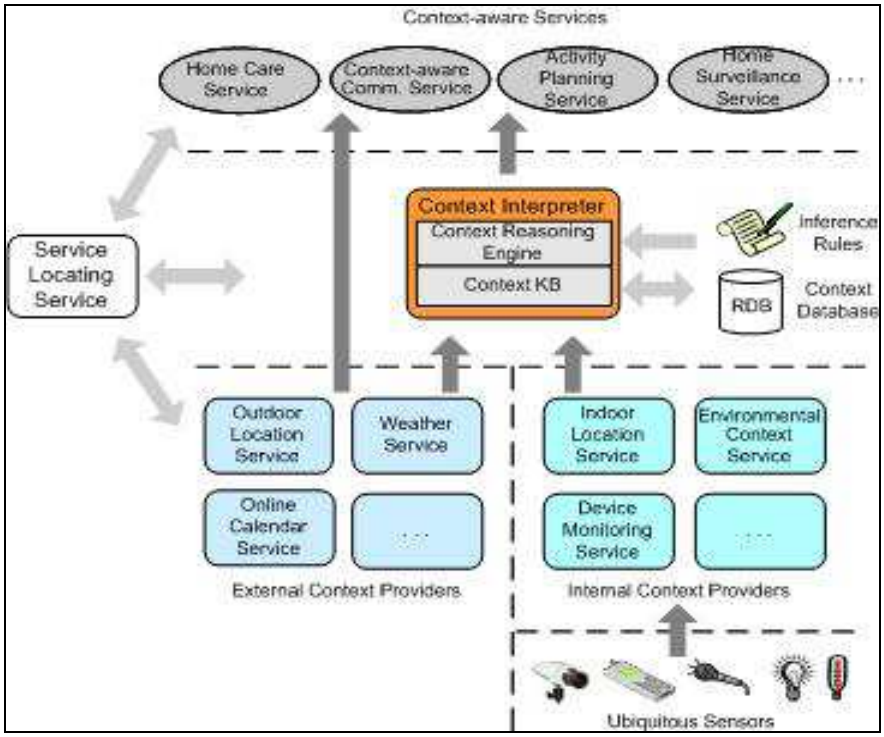


Figure 2-7: SOCAM Architecture

The SOCAM architecture aims to enable rapid prototyping of context-aware services in pervasive computing environment. It provides ontology for context description. This middleware takes into account context acquisition and interpretation. It offers an API for context subscription but there is no means to describe relevant context, so each service has to analyze and alter the acquired context to detect relevant changes. The SOCAM middleware uses the inference engine provided by the ontology for adapting those services to context changes.

2.3.4.4 PACE

PACE project [Henricksen05a] and [Henricksen05b] investigates a variety of issues related to pervasive computing, including the design of context-aware applications and solutions for modeling and managing context information. An early form of their middleware as given in [Henricksen04b] is shown in Figure 2.8. Further tools and components have been added subsequently as additional context-aware application requirements are uncovered.

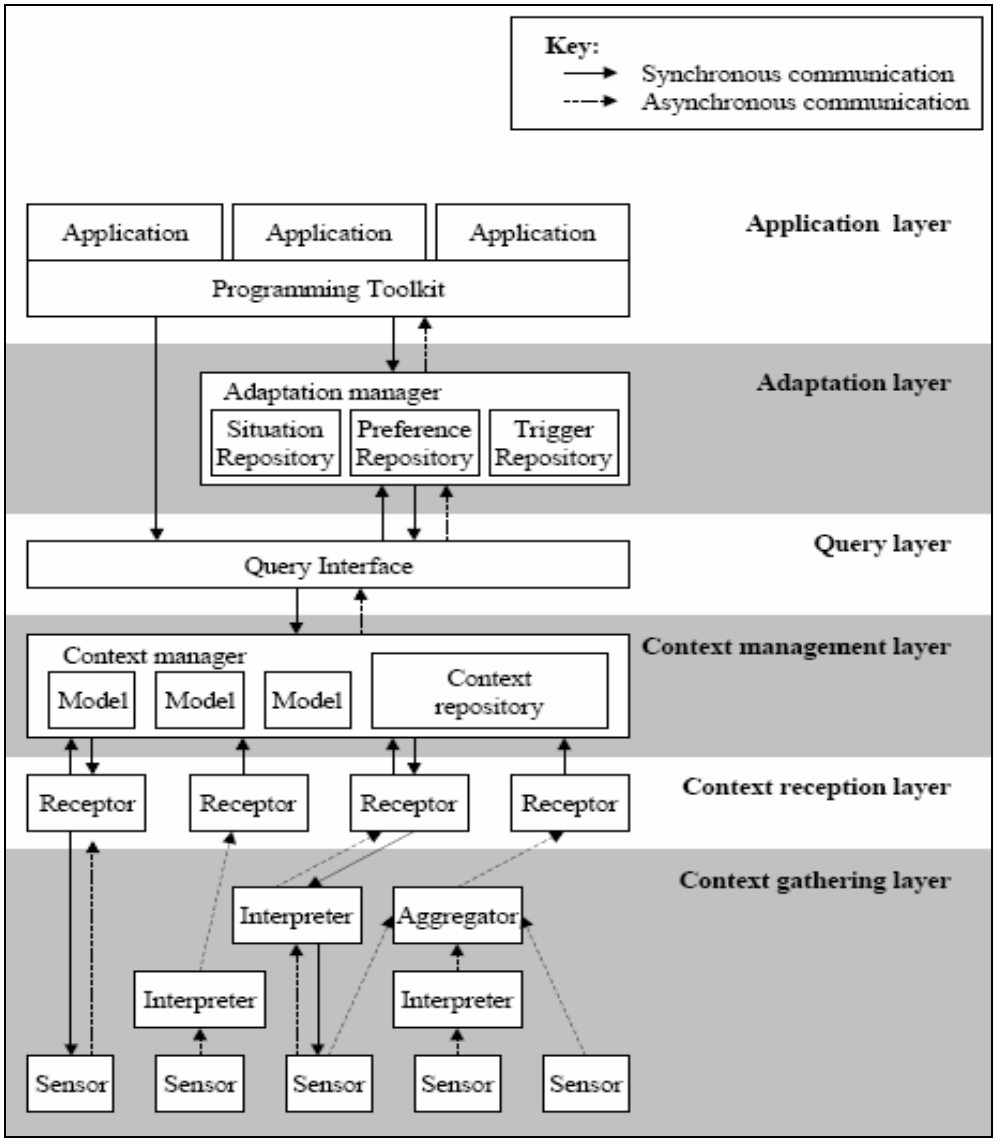


Figure 2-8: Layered context-aware infrastructure

Components and tools in the recent development of context-aware infrastructure in PACE has been developed according to the following design principles:

- The model of context information used in a context-aware system should be explicitly represented within the system. This representation should be separate from the application components and the parts of the system concerned with sensing and actuation, so that the context model can evolve independently, without requiring any components to be re-implemented.
- The context-aware behavior of context-aware applications should be determined, at least in part, by external specifications that can be customized by users and evolved along with the context model (again, without forcing re-implementation of any components).

- The communication between application components, and between the components and middleware services, should not be tightly bound to the application logic, so that a significant reimplementation effort is required when the underlying transport protocols or service interfaces change.

PACE project showed the importance of middleware for building context-aware systems. They also emphasized the importance of privacy, tolerance for failures and decision support. However, issues like scalability, standardization, collaboration in mobility and ubiquity, pervasiveness and the development of domain independent reusable platform has to be closely investigated in order to make context-aware computing a reality.

2.3.4.5 MobiLife

MobiLife project [MobiLife07] [Mrohs06] is to bring advances in mobile applications and services within the reach of users in their everyday life by innovating and deploying new applications and services based on the evolving capabilities of the 3G systems and beyond. The project addresses, with a strong user-centric view, the problematic related to different end-user devices, available communication networks, interaction modes, applications and services. Four major working areas of MbiLife are: User centricity, practical applications and services, architectures and technologies, and evaluation. MobiLife focus areas are based on three communication spheres: self-awareness, group-awareness and world-awareness. This is shown in Figure 2.9.

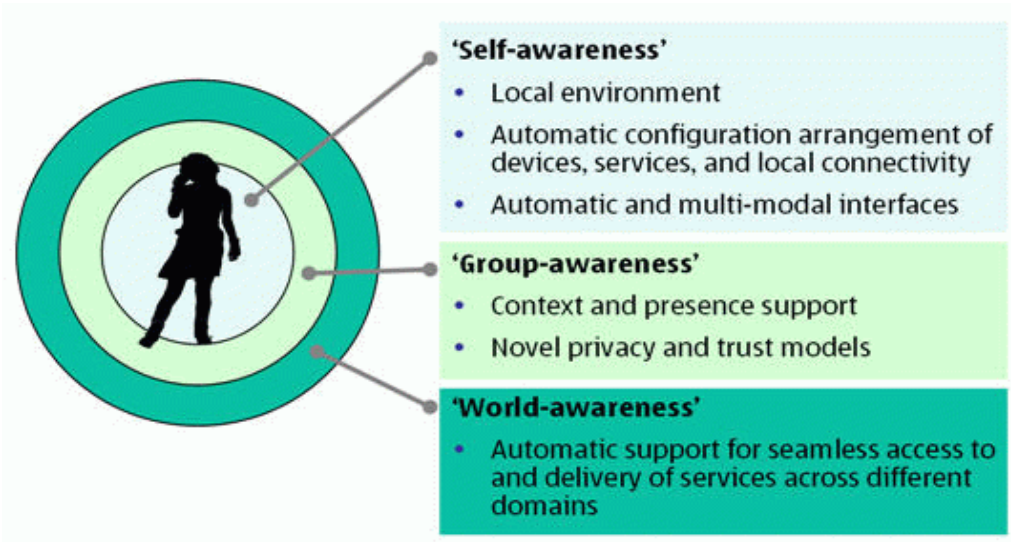


Figure 2-9: MobiLife communication spheres

MobiLife solution is based on methodology of looking at the world from the user perspective. This includes:

- Try to understand the user needs.
- How these needs can be transferred to technological and non-technological requirements.
- How the product development process can be elaborated to iteratively and interactively take those needs and requirements into account.

Technologies for maintaining a “shared cognition” amongst groups of users, such as modeling and reasoning for contextual awareness, technologies for facilitating and maintaining privacy and trust, and technologies for creating and sharing various kinds of content and media related to everyday life belonged to key areas covered in the project. The enablers and technologies were embodied in the application prototypes thus providing the project further opportunities to learn interactively how they could facilitate providing sustained benefit to the end-users.

2.3.4.6 PerSE

PerSE, *a pervasive service environment* project, [Gripay06] and [Pigeot07], initiated in our team represents the vision to develop a user-oriented pervasive computing environment. PerSE allows the user to have access to resources (services, data) hosted on various surrounding devices by simply expressing an intention.

To take part in a PerSE environment, each device has to run a meta-service, the Base, enabling it to share its local resources. The PerSE Base is in charge of communications with other Bases, in order to run distributed services in a smart and optimized way. The PerSE environment consists of many independent Bases, able to discover each other, and to send and receive messages through different communication channels (LAN, Wifi, Bluetooth) available on the devices.

In PerSE, intension of users is expressed using PsaQL [Bihler06], a Pervasive Service Action Query Language developed by the same group. PsaQL looks like SQL at a glance but has enhanced features to handle queries in pervasive environment. The PsaQL language enables the user (or an application) to express intention (called a *partial* action) describing the services the user wants to use and their possible location. The PerSE Base has to then interpret this intention into a connected graph of services meant to be executed (called a complete action).

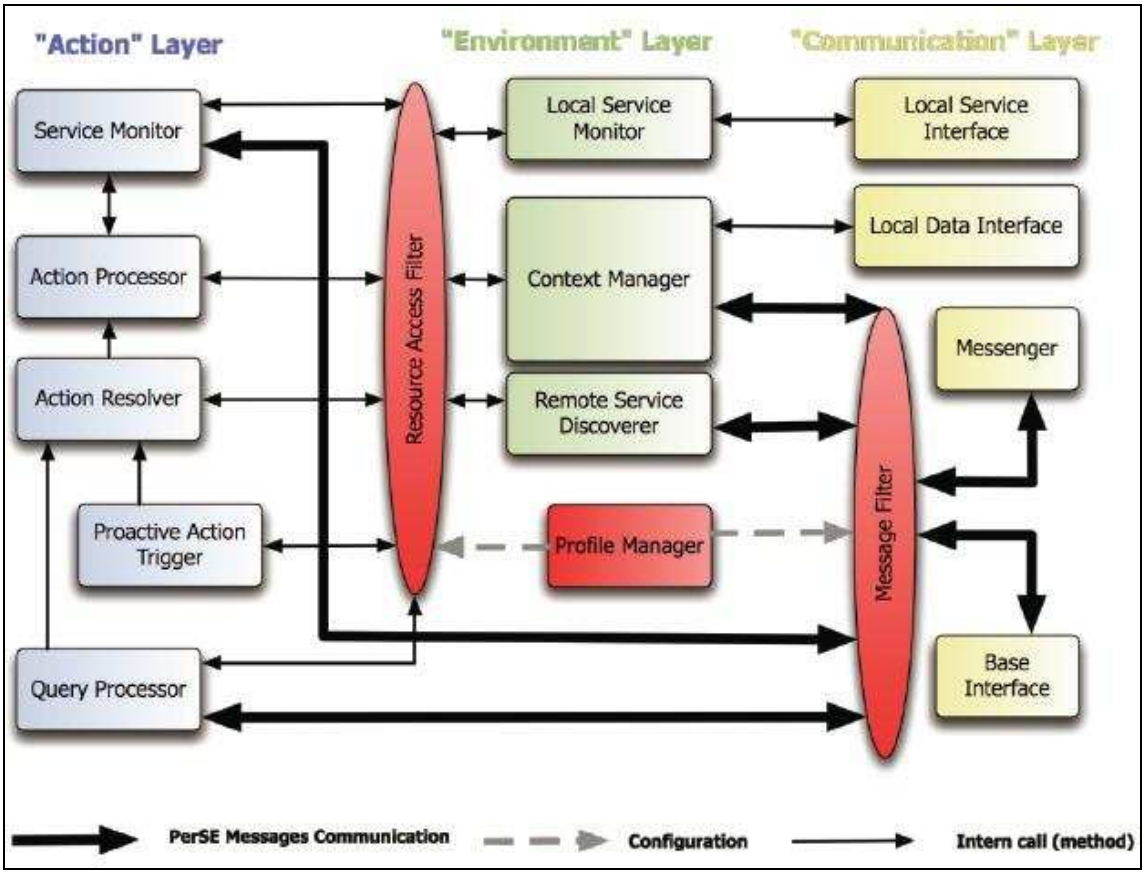


Figure 2-10: PerSE Architecture

The PerSE architecture (Figure 2.10) is composed of three layers, corresponding to the three main functionalities of the Base: Communication, Environment and Action. Between and within these layers, a security infrastructure, composed of three modules is integrated.

The idea behind the development of the PerSE middleware as part of the PerSE project is to promote the idea of creating a standardized pervasive system development platform to which components can be added from time to time. The work in this thesis is running in parallel with the project PerSE. At last, we are aiming to interface with (plug into) the PerSE platform.

2.4 Review of technologies and tools

In this section, we highlight on some baseline technologies that we found are necessary for the implementation of our semantic based collaborative context management and reasoning. The advancement of enabling technologies in the area is among the motivating factors for this work.

2.4.1 RDF and RDFS

Resource Description Framework (RDF) [RDF06] is a W3C-endorsed language for describing information about resources on the WWW. The main entity in RDF is called a resource. RDF uses the URI mechanism from XML to identify uniquely these resources, and the RDF language consists of statements that are made about the resources. RDF is a directed, labeled graph data format for representing information in the Web. Each RDF statement consists of a subject, a predicate and an object. A subject is the resource about which the statement is made. A predicate is an attribute or characteristic of the subject. An object can be either a resource or a literal value like a string or an integer.

RDF is based on the idea of identifying things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. The group of statements "there is a Person identified by `http://www.w3.org/People/EM/contact#me`, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr." could be represented as the RDF graph in Figure 2.11.

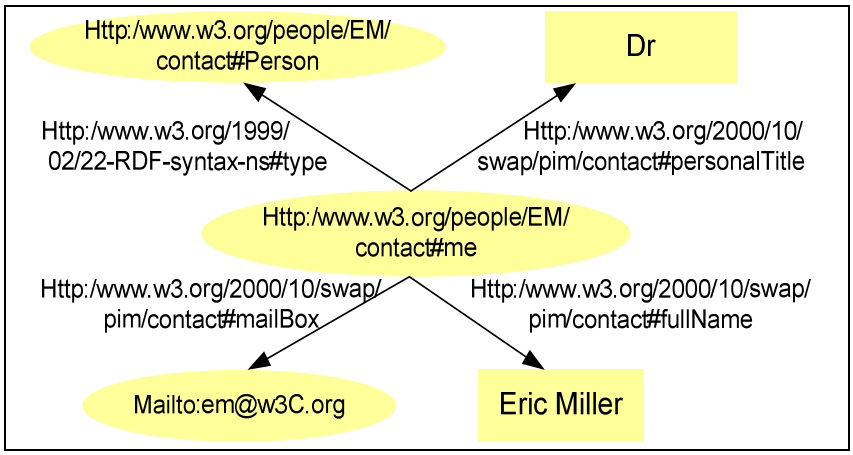


Figure 2-11: RDF graph example

- individuals, e.g., Eric Miller, identified by
`http://www.w3.org/People/EM/contact#me`
- kinds of things, e.g., Person, identified by
`http://www.w3.org/2000/10/swap/pim/contact#Person`
- properties of those things, e.g., mailbox, identified by
`http://www.w3.org/2000/10/swap/pim/contact#mailbox`

- values of those properties, e.g. `mailto:em@w3.org` as the value of the mailbox property (RDF also uses character strings such as "Eric Miller", and values from other data types such as integers and dates, as the values of properties)

RDF uses XML Schema data types to denote the type of these literal values. RDF by itself does not define very strong semantics and therefore, in order to bring in more structure, RDFS (RDF Schema) was developed. RDFS allows authors to create simple hierarchies using the rdfs:Class resource and rdfs:subClassOf property. Resources can be declared to be instances of an rdfs:Class by means of the rdfs:type property. RDFS also allows authors to specify the domain and range of properties. Finally, properties in RDFS can be declared to be sub properties of other properties by means of the rdfs:subPropertyOf property.

2.4.2 **Ontology and OWL**

Ontology formally defined by [Gruber93], for the first time, is “an explicit specification of a conceptualization”. The term is borrowed from philosophy, where ontology is a systematic account of Existence. For artificial intelligence systems, what “exists” is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, ontology is the statement of a logical theory”.

Ontology therefore is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. In addition, it is certainly a different sense of the word than its use in philosophy. Ontology allows a programmer to specify, in an open, meaningful, way the concepts and relationships that collectively characterize some domain. Examples might be the concepts of red and white wine, grape varieties, vintage years, wineries and so forth that characterize the

domain of 'wine', and relationships such as 'wineries produce wines', 'wines have a year of production'. This wine ontology might be developed initially for a particular application, such as a stock-control system at a wine warehouse. As such, it may be considered similar to a well-defined database schema. The advantage to ontology is that it is an explicit, first-class description. So having been developed for one purpose, it can be published and reused for other purposes. For example, a given winery may use the wine ontology to link its production schedule to the stock system at the wine warehouse. Alternatively, a wine recommendation program may use the wine ontology, and a description (ontology) of different dishes to recommend wines for a given menu.

Because of the limited expressive power of RDFS, people have worked on more powerful alternative to extensions to RDFS to represent ontology. One of the most widely known alternatives is the Web Ontology Language, OWL [McGuinness07] [OWL07]. OWL has evolved out of DAML and OIL and it is a W3C recommendation. It is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. It is built on top of RDF/RDFS, which means that OWL ontology also consists of statements that are made about resources. Like RDFS, OWL distinguishes between classes and instances. Among the language constructs that give OWL more expressive power than RDFS are: ontology importing, different types of properties, cardinalities, characteristics of properties, value restrictions, equivalence between classes and individual and set operation.

OWL has vocabularies that make it among the best candidate languages in a semantic Web and for reasoning and inference in knowledge management. Details of these vocabularies are given in the annex section.

2.4.3 **Protégé editor and tools**

Protégé [Protégé07] is an ontology editor developed at Stanford. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontology. It is available as a stand-alone Java application or as a Java Applet. It allows users to create and edit ontology and store them in RDF(S) or other formats. Protégé allows the installations of plug-ins. Plug-ins

are available for importing/exporting OWL files; provide graphical tools for authoring ontology, etc. It has a clean graphical user interface, with separate tabs for displaying ontology classes, properties and instances. Classes and properties are organized in tree structures. The Protégé-OWL editor enables users to:

- Load and save OWL and RDF ontologies.
- Edit and visualize classes and properties.
- Define logical class characteristics as OWL expressions.
- Execute reasoner such as description logic classifiers.
- Edit OWL individuals for Semantic Web markup.

At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building advanced knowledge-based tools and applications. The Protégé-OWL API is an open-source Java library for the Web Ontology Language (OWL) and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces. The API is designed to be used in the development of components that are executed inside of the Protégé-OWL editor's user interface.

The output obtained by executing this program as stand-alone is "Class URI: http://hello.com#World". The Protégé-OWL APIs are built from collection of Java interfaces from the model package. These interfaces provide access to the OWL model and its elements like classes, properties, and individuals. Application developers should not access the implementation of these interfaces directly, but only operate on the interfaces. Using these interfaces, you do not have to worry about the internal details of how Protégé stores ontology.

The most important model interface is OWLModel, which provides access to the top-level container of the resources in the ontology. You can use OWLModel to create, query, and delete resources of various types and then use the objects returned by the OWLModel to

do specific operations. For example, the following snippet creates a new OWLNamedClass (which corresponds to owl:Class in OWL), and then gets its URI:

```
OWLModel owlModel = ProtegeOWL.createJenaOWLModel();
OWLNamedClass worldClass=owlModel.createOWLNamedClass("World");
System.out.println("Class URI: " + worldClass.getURI());
```

Note that the class ProtegeOWL provides a couple of convenient static methods to create OWLModels, also from existing OWL files. For example, you can load an existing ontology from the web using:

```
String uri = "http://www.owl-ontologies.com/travel.owl";
OWLModel owlModel=ProtegeOWL.createJenaOWLModelFromURI(uri);
```

2.4.4 Jena reasoning framework

Jena [Jena07] is a Java API based framework for building Semantic Web applications that allows users to read, write, and manipulate RDF(S) and OWL models. The “com.hp.hpl.jena...impl” packages contains most of the common Jena implementation classes.

Jena can read and write files in any of the standard RDF storage formats. In addition, Jena can store and read RDF data in a relational database (MySQL, PostgreSQL, and Oracle are supported). Jena offers statement-centric (based on the subject-predicate-object structure) support for manipulating RDF and OWL data (including typed literals, RDFS and OWL specific constructs, and standard vocabularies), and comes with a built-in RDF query language, SPARQL. Jena provides a programmatic environment for RDF, RDFS, OWL and SPARQL and includes a rule-based inference engine. Jena has object classes to represent graphs, resources, properties and literals. The interfaces representing resources, properties and literals are called Resource, Property and Literal respectively. In Jena, a graph is called a model and is represented by the Model interface. The code to create this graph, or model, is demonstrated below. The code begins with some constant definitions and then creates an empty Model, using the ModelFactory method called createDefaultModel() to create a memory-based model. Jena contains other implementations of the Model interface, e.g one which uses a relational database: these types of Model are also available from ModelFactory. The John Smith resource is then created and a property added to it. The property is provided by a "constant" class VCARD that holds objects representing all the definitions in the VCARD schema.

```
// some definitions
static String personURI = "http://somewhere/JohnSmith";
static String fullName = "John Smith";
// create an empty Model
Model model = ModelFactory.createDefaultModel();
// create the resource
Resource johnSmith = model.createResource(personURI);
// add the property
 johnSmith.addProperty(VCARD.FN, fullName);
```

Let us add some more detail to the *vcard*, exploring some more features of RDF and Jena. In the first example, the property value was a literal. RDF properties can also take other resources as their value. Extending this example, we can add new property, *vcard:N*, to represent the structure of John Smith's name. The *vcard:N* property takes a resource as its value. We can also represent the compound name that has no URI known as an *blank Node*. The Jena code to construct this extended example is given below.

```
// some definitions
String personURI    = "http://somewhere/JohnSmith";
String givenName    = "John";
String familyName   = "Smith";
String fullName     = givenName + " " + familyName;
// create an empty Model
Model model = ModelFactory.createDefaultModel();
// create the resource
//   and add the properties cascading style
Resource johnSmith
  = model.createResource(personURI)
         .addProperty(VCARD.FN, fullName)
         .addProperty(VCARD.N,
               model.createResource()
                  .addProperty(VCARD.Given, givenName)
                  .addProperty(VCARD.Family, familyName));
```

Each arc in an RDF Model is called a statement. Each statement asserts a fact about a resource. A statement has three parts: the subject is the resource from which the arc leaves, the predicate is the property that labels the arc and the object is the resource or literal pointed to by the arc. A statement is sometimes called a triple, because of its three parts.

An RDF Model is represented as a *set* of statements. Each call of `addProperty` in the above example added another statement to the Model. (Because a Model is set of

statements, adding a duplicate of a statement has no effect.) The Jena model interface defines a `listStatements()` method which returns an `StmtIterator`, a subtype of Java's `Iterator` over all the statements in a Model. `StmtIterator` has a method `nextStatement()` which returns the next statement from the iterator (the same one that `next()` would deliver, already cast to `Statement`). The `Statement` interface provides accessor methods to the subject, predicate and object of a statement.

Now we will use the interface to extend the above example to list all the statements created and print them out as follows:

```
// list the statements in the Model
StmtIterator iter = model.listStatements();
// print out predicate, subject and object
while (iter.hasNext()) {
    Statement stmt = iter.nextStatement();
    Resource subject = stmt.getSubject();
    Property predicate = stmt.getPredicate();
    RDFNode object = stmt.getObject();
    System.out.print(subject.toString());
    System.out.print(" " + predicate.toString() + " ");
    if (object instanceof Resource) {
       System.out.print(object.toString());
    } else {
        // object is a literal
        System.out.print(" \"" + object.toString() + "\"");
    }
    System.out.println(" .");
}
```

Since the object of a statement can be either a resource or a literal, the `getObject()` method returns an object typed as `RDFNode`, which is a common superclass of both `Resource` and `Literal`. The underlying object is of the appropriate type, so the code uses `instanceof` to determine which and processes it accordingly.

Jena also provides operations for manipulating Models as a whole. The common set operations like union (Model1.union(Model2)), intersection (Model1.intersection(Model2)) and difference (Model1.difference(Model2)); can be used to manipulate data models.

An extension to RDF based Jena model for reasoning in the semantic applications is the Jena ontology model. The Jena Ontology API is language-neutral. The Jena OntModel extends the Jena RDF Model by adding support for the kinds of objects expected to be in ontology: classes (in a class hierarchy), properties (in a property hierarchy) and individuals. The properties defined in the ontology language map to accessor methods. For example, an OntClass has a method to list its super-classes, which corresponds to the values of the subClassOf property. When the OntClass listSuperClasses() method is called, the information is retrieved from the underlying RDF statements. Similarly adding a subclass to an `OntClass` asserts an additional RDF statement into the model.

An ontology model is an extension of the Jena RDF model that provides extra capabilities for handling ontology data sources. Ontology models are created through the Jena ModelFactory. The simplest way to create an ontology model is as follows:

```
OntModel m = ModelFactory.createOntologyModel();
```

To create a model with a given specification, invoke the `ModelFactory` as follows:

```
OntModel m = ModelFactory.createOntologyModel(
OntModelSpec.OWL_MEM );
```

To create a custom model specification, we can create a new one from scratch and call the various methods to set the appropriate values. More often, we want only a variation on an existing recipe. In this case, we copy an existing specification and then update the copy as necessary:

```
OntModelSpec s = new OntModelSpec( OntModelSpec.OWL_MEM );
s.setDocumentManager( myDocMgr );
OntModel m = ModelFactory.createOntologyModel( s );
```

MySQL database engine [MySQL07] with its ODBC-JDBC [ODBC07] and database connection to Jena framework is a key entry point to data transactions in Jena. Transaction and query on the data axis of the context is based on the SQL [Groff02].

2.4.5 **The SPARQL query language**

Data retrieval on the semantic axis of context triples while reasoning is done using the SPARQL [SPARQL07] query language. SPARQL query language is based on matching graph patterns that contain triple patterns, conjunctions, disjunctions, and optional patterns. Triple patterns are like RDF triples, but with the option of a query variables in place of RDF terms in the subject, predicate or object positions. Combining triple patterns gives a basic

graph pattern, where an exact match to a graph is needed. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

For example, the following SPARQL query returns all mobile devices having a VGA display and a processor speed more than 400Mhz. It also displays the name of the owner if that information is available. The result from the query will look like the one in Table 2.2.

```
//SPARQL query formation
PREFIX ns: <http://www.myexample.com/coca.owl#>
SELECT ?mdv ?ps ?person
WHERE {
      ?mdevice rdfs:subClassOf ns:Device.
      ?mdevice ns:deviceType ns:Mobile.
      ?mdv rdf:type ?mdevice.
      ?mdv ns:displayType ns:VGA.
      ?mdv ns:processorSpeed ?ps.
      OPTIONAL{?mdv ns:ownedBy ?person.}
      FILTER(?ps>400)
}
```

Table 2-2: Sample output from a SPARQL query

mdv	ps	person
PDA001	**480**	
TabletPC009	**700**	Bob
SmartPhone023	**420**	

2.4.6 JXTA collaboration protocols

The potential use of peer-to-peer networking has been demonstrated by the popularity of applications like Napster [Napster06], Gnutella [Gnutella06], FreeNet [FreeNet06]. As a complement to this, Jxta [JXTA07] has been introduced by Sun Microsystems. JXTA technology is a set of open, generalized peer-to-peer protocols that allows any connected device (cell phone to PDA, PC to server) on the network to communicate and collaborate. Project-JXTA is an open source effort that had involved the developer community from the start. It provides a peer-to-peer infrastructure over which other peer- to-peer applications

can be built. JXTA protocols standardize the manner in which peers self-organize into peer groups, discover each other, advertise network services, communicate with each other, and monitor each other. Figure 2.12 shows an overlay of such virtual peer-to-peer collaboration network.

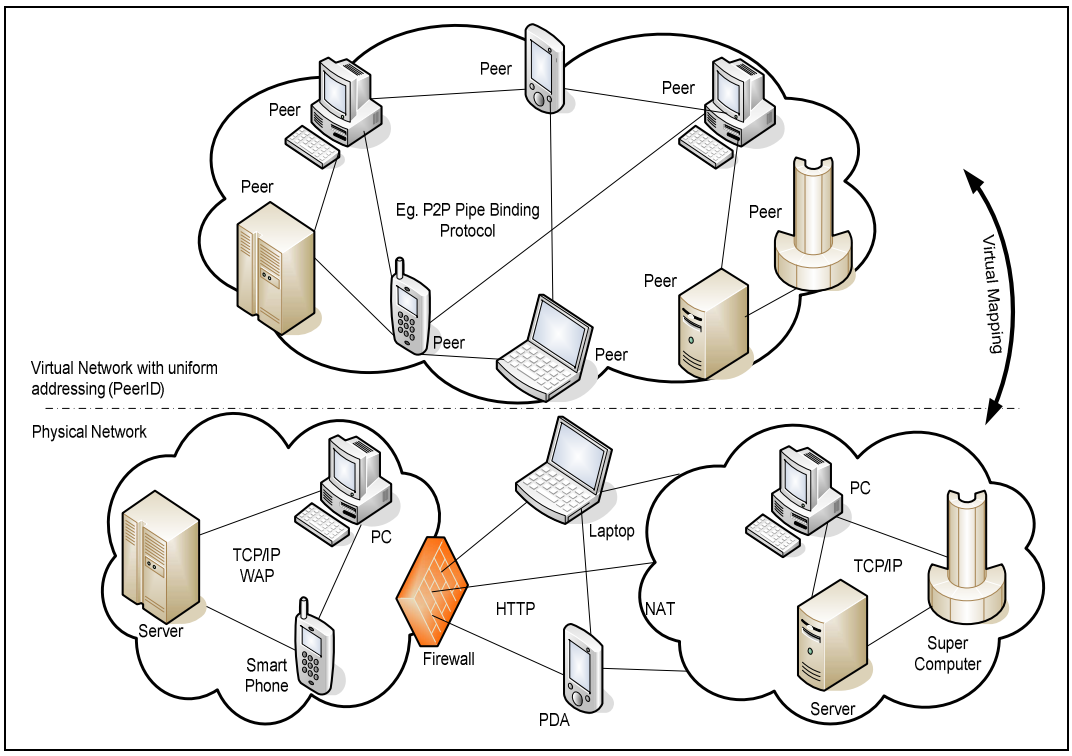


Figure 2-12: Overlay of the virtual peer-to-peer collaboration network

A JXTA peer is any networked device (sensor, phone, PDA, PC, server, supercomputer, etc.) that implements the core JXTA protocols. Each peer is identified by a unique ID. Peers are autonomous and operate independently and asynchronously of all other peers. Some peers may have dependencies upon other peers. This might be due to special requirements such as the need for gateways, proxies or routers. Peers may publish network services and resources (CPU, storage, databases, documents, etc.) for use by other peers. A peer may cache advertisements for JXTA resources, but doing so is optional. Peers may have persistent storage.

Peers are typically configured to discover spontaneously each other on the network to form transient or persistent relationships with other peers. Peers that provide the same set of services tend to be interchangeable. As a result, peers typically need to interact with only a small number of other peers (network neighbors or buddy peers). Peers should not make assumptions about the reliability of other peers. Peers may join or leave the network at any

time. A peer should always anticipate that connectivity might be lost to any peer that it is currently communicating with.

Peers may advertise multiple network interfaces. Each published interface is advertised as a peer endpoint. A peer endpoint is a URI that uniquely identify a peer network interface (for example, a URI might specify a TCP port and associated IP address). Peer endpoints are used by peers to establish direct point-to-point connections between two peers.

Peers self-organize into Peer Groups. A peer group is a collection of peers that have a common set of interests. Each peer group is uniquely identified by a PeerGroup Id. The JXTA protocols do not dictate when, where, or why peer groups are created. The JXTA protocols only describe how peers may publish, discover, join, and monitor peer groups. Communicating peers are not required to have direct point-to-point network connection between themselves. A peer may need to use one or more intermediary peers to route a message to another peer that is separated due to physical network connections or network configurations (e.g., NATs, firewalls, or proxies).

The JXTA protocols are a set of six protocols that have been specifically designed for ad hoc, pervasive, and multi-hop peer-to-peer network computing. Using the JXTA protocols, peers can cooperate to form self-organized and self-configured peer groups independent of their positions in the network (edges, firewalls, network address translators, public vs. private address spaces), and without the need of a centralized management infrastructure.

The JXTA protocols are designed to have very low overhead, made few assumptions about the underlying network transport and impose few requirements on the peer environment, and yet are able to be used to deploy a wide variety of peer-to-peer applications and services in a highly unreliable and changing network environment.

Peers use the JXTA protocols to advertise their resources and to discover network resources (services, pipes, etc.) available from other peers. Peers form and join peer groups to create special relationships. Peers cooperate to route messages allowing for full peer connectivity. The JXTA protocols allow peers to communicate without the need to understand or manage the potentially complex and dynamic network topologies, which are increasingly common.

The JXTA protocols allow peers to route dynamically messages across multiple network hops to any destination in the network (potentially traversing firewalls). Each message carries with it either a complete or a partially ordered list of gateway peers through which the message might be routed. Intermediate peers in the route may assist the routing by using routes they know of to shorten or optimize the route a message is set to follow.

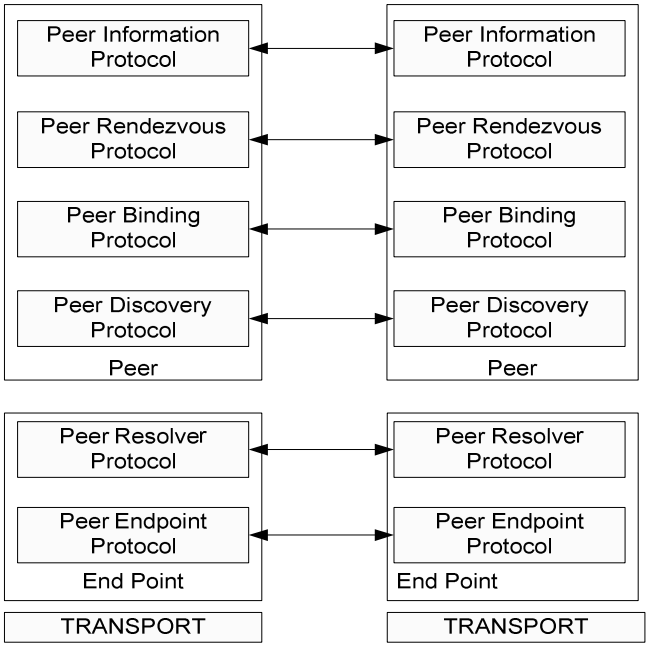


Figure 2-13: JXTA protocols

The JXTA protocols are composed of six protocols (Figure 2.13) that work together to allow the discovery, organization, monitoring and communication between peers:

- Peer Resolver Protocol (PRP) is the mechanism by which a peer can send a query to one or more peers, and receive a response (or multiple responses) to the query. The PRP implements a query/response protocol. The response message is matched to the query via a unique id included in the message body. Queries can be directed to the whole group or to specific peers within the group.
- Peer Discovery Protocol (PDP) is the mechanism by which a peer can advertise its own resources, and discover the resources from other peers (peer groups, services, pipes and additional peers). Every peer resource is described and published using an advertisement. Advertisements are programming language-neutral metadata structures that describe network resources. Advertisements are represented as XML documents.
- Peer Information Protocol (PIP) is the mechanism by which a peer may obtain status information about other peers. This can include state, uptime, traffic load, capabilities, and other information.

- Pipe Binding Protocol (PBP) is the mechanism by which a peer can establish a virtual communication channel or pipe between one or more peers. The PBP is used by a peer to bind two or more ends of the connection (pipe endpoints). Pipes provide the foundation communication mechanism between peers.
- Endpoint Routing Protocol (ERP) is the mechanism by which a peer can discover a route (sequence of hops) used to send a message to another peer. If a peer "A" wants to send a message to peer "C", and there is no known direct route between "A" and "C", then peer "A" needs to find intermediary peer(s) who will route the message to "C". ERP is used to determine the route information. If the network topology changes and makes a previously used route unavailable, peers can use ERP to find an alternate route.
- Rendezvous Protocol (RVP) is the mechanism by which peers can subscribe or be a subscriber to a propagation service. Within a peer group, peers can be either rendezvous peers or peers that are listening to rendezvous peers. The Rendezvous Protocol allows a peer to send messages to all the listening instances of the service. The RVP is used by the Peer Resolver Protocol and by the Pipe Binding Protocol in order to propagate messages.

All of these protocols are implemented using a common messaging layer. This messaging layer is what binds the JXTA protocols to various network transports. Each of the JXTA protocols is independent of the others. A peer is not required to implement all of the JXTA protocols to be a JXTA peer. A peer only implements the protocols that it needs to use. For example:

- A device may have all the necessary advertisements it uses pre-stored in memory, and therefore not need to implement the Peer Discovery Protocol.
- A peer may use a pre-configured set of router peers to route all its messages. Because the peer just sends messages to the known routers to be forwarded, it does not need to fully implement the Endpoint Routing Protocol.
- A peer may not need to obtain or wish to provide status information to other peers; hence, the peer might not implement the Peer Information Protocol.

Each peer must implement two protocols in order to be addressable as a peer: the Peer Resolver Protocol and the Endpoint Routing Protocol. These two protocols and the advertisements, services and definitions they depend upon are known as the JXTA Core Specification. The JXTA Core Specification establishes the base infrastructure used by other services and applications.

The remaining JXTA protocols, services and advertisements are optional. JXTA implementations are not required to provide these services, but are strongly recommended to do so. Implementing these services provides greater interoperability with other implementations and broader functionality. These common JXTA services are known as the JXTA Standard Services.

A peer may decide to cache advertisements discovered via the Peer Discovery Protocol for later usage. It is important to point out that caching is not required by the JXTA architecture, but caching can be an important optimization. By caching advertisements, a peer avoids the need to perform a new discovery each time it accesses a network resource. In transient environment, performing the discovery is the only viable solution. In static environments, caching is more efficient.

A unique characteristic of peer-to-peer networks, like JXTA, is their ability to replicate spontaneously information toward end-users. Popular advertisements are likely to be replicated more often, making them easier to find as more copies become available. Peers do not have to return to the same peer to obtain the advertisements they seek. Instead of querying the original source of an advertisement, peers may query neighboring peers that have already cached the information. Each peer may potentially become an advertisement provider to any other peer.

The JXTA protocols have been designed to allow JXTA to be easily implemented on uni-directional links and asymmetric transports. In particular, many forms of wireless networking do not provide equal capability for devices to send and receive. JXTA permits any uni-directional link to be used when necessary, improving overall performance and network connectivity in the system. The intent is for the JXTA protocols to be as pervasive as possible, and easy to implement on any transport. Implementations on reliable and bi-directional transports such as TCP/IP or HTTP should lead to efficient bi-directional communications.

The JXTA uni-directional and asymmetric transport plays well in multi-hop network environments where the message latency may be difficult to predict. Furthermore, peers in a peer-to-peer network tend to have nondeterministic behaviors. They may join or leave the network on a very frequent basis.

Another important advantage of using JXTA is the development of its Micro Edition, JXME [JXME07]], that provides a JXTA compatible platform on resource constrained devices using the Connected Limited Device Configuration (CLDC) or the Mobile Information Device Profile 2.0 (MIDP), or Connected Device Configuration (CDC). The range of devices includes the smart phones to PDAs. Using JXTA Java Micro Edition platform, any CLDC/MIDP/CDC device can participate in the JXTA network with any other JXTA device.

2.5 Summary

Pervasive context-aware systems must address many of the requirements of traditional distributed systems, such as heterogeneity, mobility, scalability, and tolerance for component failures and disconnections. In addition, according to [Henricksen04b], it must protect users' personal information, such as location and preferences, in accordance with their privacy preferences, and ensure that automatic actions taken by context-aware applications on behalf of users can be adequately understood and controlled by users. The large number of distributed components that are present in context-aware systems introduces a requirement for deploying, configuring and managing networks of sensors, actuators, context processing components, context repositories, and so on. Summary of these requirements are:

Support for *heterogeneity*: Hardware components ranging from resource-poor sensors, actuators and mobile client devices to high-performance servers must be supported.

Support for *mobility*: All components can be mobile, and the communication protocols that underpin the system must therefore support appropriately flexible forms of routing. Context information may need to migrate with context-aware components. Flexible component discovery mechanisms are required.

Scalability: Context processing components and communication protocols must perform adequately in systems ranging from few to many sensors, actuators and application components. Similarly, they must scale to large volume of context data.

Support for *privacy*: Flows of context information between the distributed components of a context-aware system must be controlled according to users' privacy needs and expectations.

Traceability and control: The state of the system components and information flows between components should be open to inspection - and, where relevant, manipulation - in order to provide adequate understanding and control of the system to users.

Tolerance for component failures: Sensors and other components are likely to fail in the ordinary operation of a context-aware system. Disconnections may also occur. The system must continue operation, without requiring excessive resources to detect and handle failures.

Ease of *deployment* and configuration: The distributed hardware and software components of a context-aware system must be easily deployed and configured to meet user and environmental requirements, potentially by non-experts.

Decision support*:* A decision is a choice between alternatives based on estimates of the values of those alternatives. Context-aware systems must be able to make decisions or help to make decisions.

Henricksen et al presented the summary of the capabilities of the common solutions selected from each category we have discussed. The result of the survey, as is shown in Table 2.3, indicates that comprehensive solutions do not exist yet. They have emphasized that none of the solutions provide decision support.

Table 2-3: Comparison on middleware support for context-aware systems

Requirement	ContextToolkit	CFN	ConFab	Gaia	RCSM
Heterogeneity	√	X	√	√	√
Mobility	√	√√	X	√	X
Scalability	X	√√	X	X	X
Privacy	X	X	√√	X	X
Traceability	X	X	√	X	X
Tolerance	X	√√	X	√	X
Deployment	X	√	X	√	√
Decision	X	X	X	X	X

(Key: √√ = Comprehensive √=Partial X=None)

In a recent similar work, [Gu05] have proposed a service based framework and middleware solution that uses context reasoning schemes as a basic source of context-aware services. [Sharmin06] have developed MARKS that have partially addressed the problem of mobility and lack of semantics for context reasoning. They have emphasized the

importance of ad-hoc communication and knowledge usability in the development of context-aware systems in pervasive computing environment.

Despite all these efforts, there is still a lot to do to ensure standard, scalability, quality of service and robustness with respect to the inherent pervasive component failure that needs self healing, varying device capacity, and frequent change of context in a pervasive environment. The uses of semantic ontology as a source of domain knowledge and ad-hoc sharing of this knowledge among pervasive devices with varies capacity are among the important issues to be addressed in a pervasive context-aware computing. The computationally intensive characteristics of context reasoning process and the lack of semantically rich context model have been a bottleneck for the development of such applications. Moreover, supporting technologies for pervasive computing like ontology tools, peer-to-peer protocols, and reasoning tools are becoming more popular and more available.

In this thesis, we propose ontology based semantically rich and hybrid context model, and a collaborative context-aware service platform that insures scalability and reusability. We use collaboration of pervasive peers where context data and all its semantic supporting elements like context ontology and rules are distributed and shared among the collaborating peers. The merits of our platform in relation to the above requirements will be discussed and comparison with related works will be presented under the summary section of chapter 5.

Chapter 3 CONTEXT MODELING: THE EHRAM MODEL

3.1 Introduction

Humans have always used their understanding of circumstances or context to navigate the world around them, to organize information, and to adapt to conditions. For example, when we are having a conversation in a market place, we talk louder so that the other person can hear. But when we are in a meeting room, we whisper so as not to disturb other people. This phenomenon is called context-awareness. Context-awareness has also been a major part of computing. For example, by using the current time, computers can give us reminders of calendar events. By using our login identity, computers can personalize the appearance of our user interface. Similarly, computers can tag files with time giving us many ways of organizing information.

In this chapter, we present our new approach to modeling context representation. We use EHRAM, a conceptual context meta model that uses context entities, their hierarchies, relationships, axioms and metadata as basic building block for context representation.

3.2 What is Context?

The most widely referenced definition of context is given by [Dey00]. Dey states that context is *"Any information that can be used to characterize the situation of an entity. An entity is a user, a place, or a physical or computational object that is considered relevant to the interaction between a user and an application, including the user and application themselves."* [Winograd01], on the other hand, indicates that definitions like that of Dey are intended to be adequately general to cover the works that have been done on context-based interactions. He however argues that in using open-ended phrases such as "any information" and "characterize", it becomes so broad that context covers everything without boundary. According to Winograd, *"something is context because of the way it is used in interpretation, not due to its inherent properties. The voltage on the power lines is a context if there is some action by the user and/or computer whose interpretation is dependant on it, but otherwise is just part of the environment."*

From Dey's generalized definition and Winogard's specification on definition of context, we consider context as *an operational term whose definition depends on the intention for which it is collected and the interpretation of the operations involved on an entity at a particular time and space rather than the inherent characteristics of the entities and the operations themselves.* Something that is context for one person might not make sense for the other. Even something that is context under certain condition for a particular person might not be important to the same person under another condition. With this background, context can be seen as *a response to the how, where, when, what, who and which type questions on entity descriptors and their interaction with one another that affect actions taken by or actions accepted by the entities.* Context can also be described in terms of *a statement that we make about the characteristics of a the entities, their relationships and properties of the relationships*.

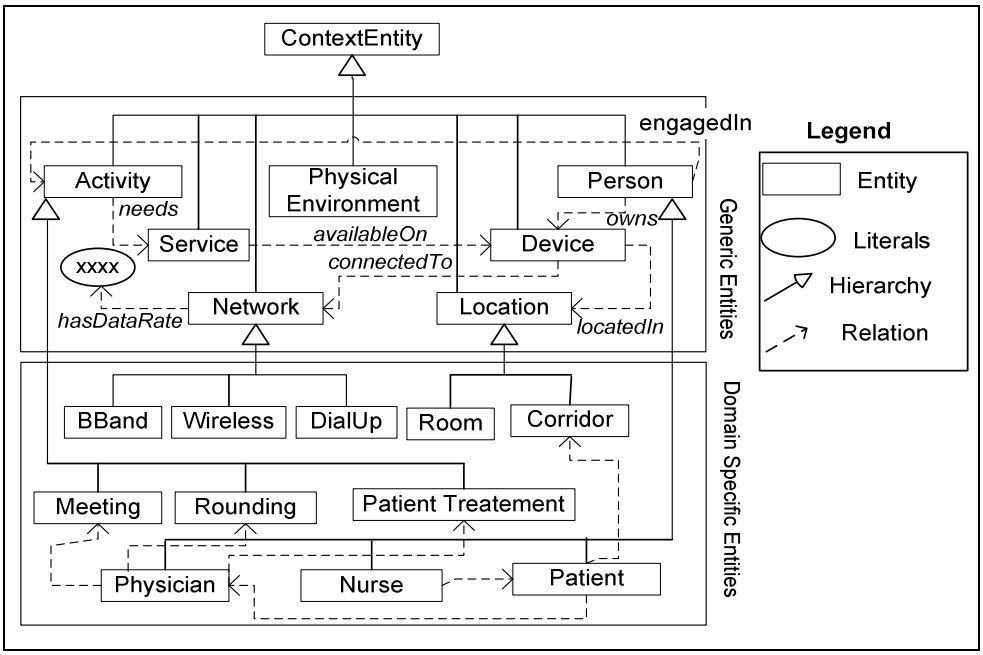


Figure 3-1: Context Entities, hierarchies and relationships

Context data is collected for each participating entity using hardware or software tools. Classes of generic context entities with some example of sub entities are given in Figure 3.1. The figure also shows relations between entities including the *subEntityOf* relation. At the root of the hierarchy is a global entity named *ContextEntity.* We can define object type relations like *locatedIn* or *connectedTo* between entities. We can also define attribute type relations like *hasDataRate* relation that take literal as its value. Entities and relations are sources of context data.

Figure 3.1 also shows the major classes of context entity descriptors we use to represent a pervasive computing environment. This includes: personal context, device context, physical environment context, network context, activity context, service context, and location context. Personal entity provides contexts like person's identity, address, service preference, device ownership, activity, location, etc. Device entity provides contexts like hardware properties, software properties, display properties, device capabilities, etc. Network entity contexts are expressed in terms of properties like delay and error characteristics, data rate, transport protocols, etc. Physical environment entity provides contexts like illumination, noise level, humidity, temperature, etc. Activity entity contexts tell if an activity is scheduled or not, if it needs special location or not, type of the activity, starting time, etc. Location entity provides contexts about its containment and situation with respect to other entities. Service entity provides context about where the service is located, type of the service (data service, audio service, video service, application service …), service availability, etc.

Some examples of how we define and represent context is shown in Table 3.1. The column on the left side shows a generic level definition of relationship between entities. This is equivalent to defining the domain and range of a relation. The column on the right side shows the use of the same relationship in a specific domain of application, in this case medical application.

Table 3-1: Sample generic and domain based definition of relationships

Generic level definition	Sample domain level equivalent
Person isEngagedIn Activity	Physician isEngageIn Patient-treatment
Location isLocatedIn Location	Library isLocatedIn Campus
Person isLocatedIn Location	Student isLocatedIn Library
Network hasDataRate xxx	ConnectionX hasDataRate low

A close look at such statements shows the need for a higher-level statement about these statements that can be expressed in terms of meta-statement or axioms. Information like time of occurrence, precision, source of data, etc. can be part of such meta-information. For example, we can state "*student isLocatedIn library*" at a given *time t*. We can also state that the precision of the statement *"a network connection has low speed"* is 85%. Time *t* that is associated with "*student isLocatedIn library,*" and precision of 85% that is associated with

"connection hasSpeed Low" refer not to the individual components of the statements but to the entire statements.

Regarding the axioms, if we define a relation *locatedIn* is transitive then *locatedIn* obeys the transitivity axiom: for example, "*library isLocatedIn campus"* and "*student isLocatedIn library"* means that "*student isLocatedIn campus"*, Similarly *if we define owned-by* is an *inverse of owns* then: "*device owned-by person"* means "*person owns device"*.

3.3 The EHRAM context representation model

The behavior of context-aware applications depends not only on their internal state and user interactions but also on the context sensed during their execution. Some early models of context representation already exist, however basic issues related to context data modeling are still not fully addressed as existing context models vary in the types of context information they can represent. While some models take the user's current situation, others model the physical environment. The challenge is to put in place a more generic approach to context modeling in order to capture and represent various features of context information including a variety of types of context information, dependencies among context information and quality of context information.

Context representation needs a model that supports easy transaction of piece wise tiny but voluminous, and dynamic context data. Equally important is the capacity to aggregate and interpret the context data with respect to its semantics in order to make it ready for reasoning and decision.

3.3.1 **EHRAM model presentation**

We now present our novel context representation model known as EHRAM context representation model. We define EHRAM as a layered context representation meta-model that uses set of entities (E), set of hierarchies (H), set of relations (R), set of axiomatic relations (A) and set of metadata (M) to represent context data and its semantics. The name EHRAM is composed from the initials of the components of EHRAM described below:

- **E** is set of entities for which context is to be captured.
- **H** is set of binary relations that form an inversely directed acyclic graph (inverted DAG) on entities. *Nodes* of the graph represent entities and *arcs* of the graph represent hierarchical relations. The root entity at the top of the hierarchy graph is a global entity known as *ContextEntity*.

- **R** stands for the union of the sets of binary relations R_e and R_a. $\mathbf{R_e}$ is set of binary relations having both its domain and range from the set E. $\mathbf{R_a}$ is set of binary relations defined from set of entities E to set of literals representing entity attributes. Domain of the relation R_a is the set of E while its range is set of literal values.
- **A** is set of axiomatic relations. Axiomatic relation is a relation about relations. For all r_i an element of set of relations R, we have zero or more a_j an element of set of axioms A that r_i obeys. For example, if we define a relation r_1 as a transitive relation then r_1 obeys the transitivity property (axiom): (e_1, r_1, e_2) and (e_2, r_1, e_3) => (e_1, r_1, e_3). Similarly, if we define a relation r_2 as a symmetric relation then r_2 obeys the symmetric property (axiom): (e_1, r_2, e_2) => (e_2, r_2, e_1).
- **M** is set of metadata information about a defined relation instance. For example, if we have a statement that says, *"Bob reported that Alice is located in the garden this morning"*. The qualifier phrases *"bob reported"* and *"this morning"* are metadata of the statement made about Alice. It answers the question *who* and *when* about the base statement. Set of Metadata M together with set of Axioms A enhance EHRAM to be a context meta-model for handling semantics of Context data.

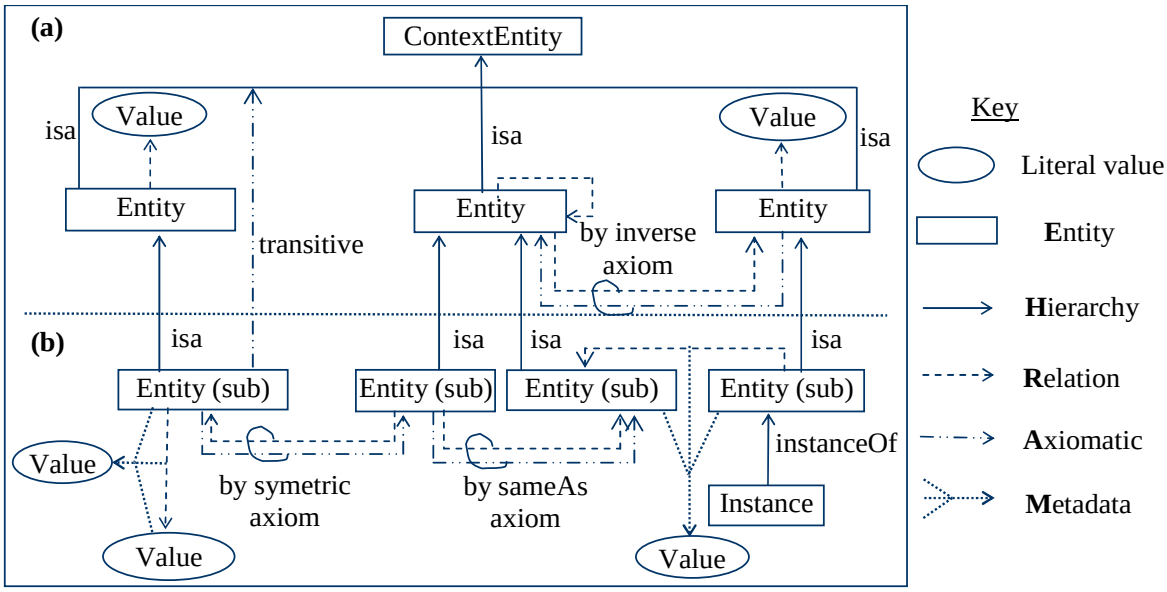


Figure 3-2: Layered representation of EHRAM components

Hierarchy is an important structure to organize and classify context entities and relations into two layers. Layered organization helps us to classify and tag context data as generic domain independent or as domain dependent. Figure 3.2 is a graphical representation of EHRAM structure that shows hierarchies, entities, entity relations, attribute relations, axiomatic relations, metadata and the layers: Layer (a) is the generic layer and layer (b) is the domain layer.

3.3.2 **EHRAM Model by example**

Consider an application in a medical domain where context data come from medical entities like patients, doctors, activities and events in the hospital, devices, locations, etc. Representation of components of the EHRAM model using few example data from the application in a medical domain is given in Figure 3.3.

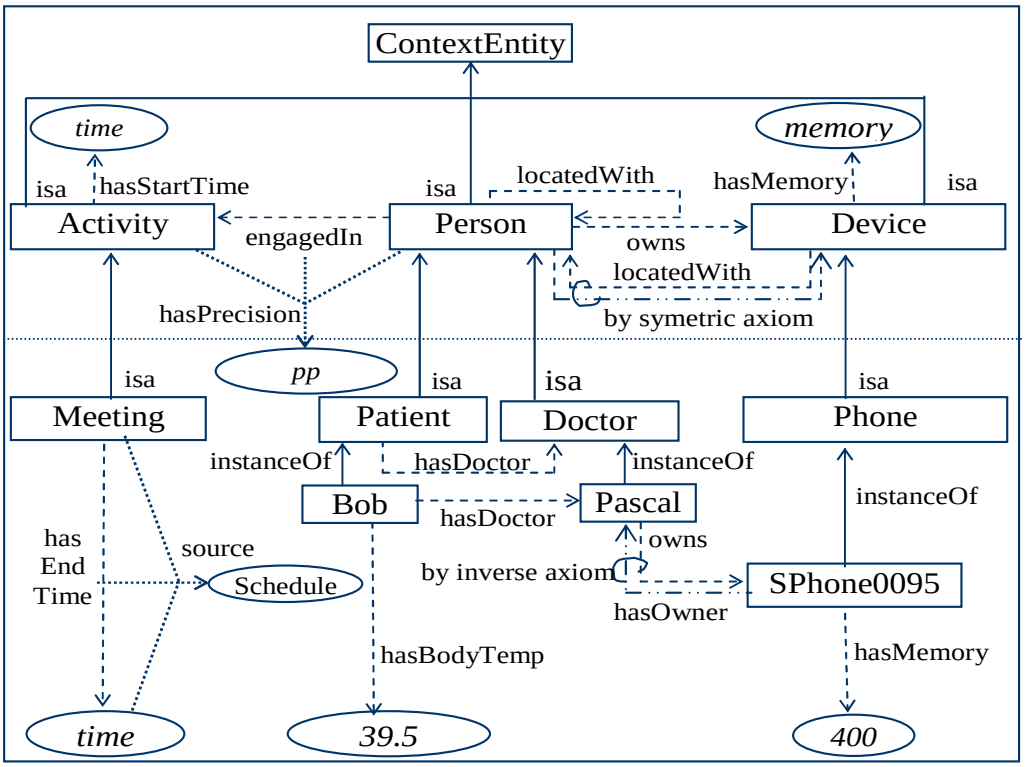


Figure 3-3: Example of context data using EHRAM components

A close look at the figure helps us understand how the five elements of EHRAM (entity, hierarchy, relation, axiom, metadata) are represented in the graph.

Entity and Hierarchy: *Activity*, *Person* and *Device* are examples of generic entity classes and they are presented in the generic layer. They are high-level context entities from which specific entities can be derived and they are common to all domains of applications. All entities in this category have hierarchical relation named *isa* with the root entity known as ContextEntity. *Meeting*, *Patient*, *Doctor* and *Phone* are examples of domain entity classes in a medical application, and are presented in the domain layer. *Bob*, *Pascal* and *SPhone0095* are examples of entity instances in the medical application. Examples of hierarchical relations from the diagram are *(Device, isa, ContextEntity), (Person, isa, ContextEntity), (Activity, isa, ContextEntity), (Doctor, isa, Person) , (Pascal, instanceOf, Doctor)*, etc.

Relation: Relations like *(Activity, hasStartTime, time), (Person, isEngagedIn, Activity), (Person, locatedWith, Person), (Person, locatedWith, Device), (Device, hasMemory, memory)* defined in the generic layer. Such generic relations can be inherited down in the hierarchy by the sub-entities and instances in the specific domain of application. Similarly relations like *(Meeting, hasEndTime, time) and (Patient, hasDoctor, Doctor)* from the medical application are presented in the domain layer. They restrict the domain and the range of the relations that are inheritable down in the hierarchy by entity instances. Finally, relations like *(Bob, hasBodyTemp, 39.5), (Pascal, Owns, SPhone0095), (Bob, hasDoctor, Pascal)* and *(SPhonr0095, hasMemory, 400)* defined on entity instances represent our basic context definition formalism.

Relations like *hasEndTime, hasMemory, hasStartTime, and hasBodyTemp* are elements of what we have defined as attribute relation (R_a) because the range of these relations is the set of literal values. Relations like *hasDoctor, locatedWith* and *owns* are elements of what we have defined as entity relations (R_e) because they are defined from entity to entity i.e., both their domain and range are drawn from set of entities.

Some relations in the diagram are defined to have associated axioms and some have metadata. Examples of relations with associated axioms are *(Person, locatedWith, Device)* and *(Pascal, owns, SPhone0095).* In the diagram, *locatedWith* is defined to be symmetric and therefore it obeys the symmetric-axiom property, that means the relation *(Device, locatedWith, Person)* automatically holds true. Similarly, because *owns* is defined to be an inverse of *hasOwner*, it obeys the inverse-axiom property, that means the relation *(SPhone0095, hasOwner, Pascal)* automatically holds true. The relation *(Person, engagedIn, Activity)* has a metadata that tells about its precision represented by *hasPrecision*.

3.4 More on layers, axioms and metadata

Generic layer in the EHRAM model consists of classes representing basic entities. Such classes have *generalization* relation with the base classes called *ContextEntity* that represents EHRAM root entity. All *association* relations and attributes defined on these entities apply to all sub entities down in the hierarchy. They are defined independent of the domain of application. For example, if we define a relation *hasAddress* that applies to an entity class Person, i.e. *hasAddress*(Person, Address), then this relation applies to all sub entities and instances of Person. Domain layer represents entities that define specific domain

of application. In the hierarchy graph, domain layer consists of all entities that do not have a direct generalization relation with the root entity. In addition to their defined relations, they inherit relations from the parent entities.

An axiom is any sentence, proposition or rule that is taken for granted as valid, and serves as a necessary starting point and formal logical expression for deducing and inferring logically consistent statements. Axiomatic relations can be defined both at the generic layer and at the domain layer of the EHRAM model. Description of some of the generic level axiomatic relations, *sameAs*, *inverse*, *symmetric* and *transitive* can be given as follows:

$\forall r \in R\ \text{symmetric}(r) \Leftrightarrow (\forall e_1, e_2 \in E,\ r(e_1, e_2) \Rightarrow r(e_2, e_1))$
$\forall r \in R\ \text{transitive}(r) \Leftrightarrow (\forall e_1, e_2, e_3 \in E,\ r(e_1, e_2) \wedge r(e_2, e_3) \Rightarrow r(e_1, e_3))$
$\forall r_1, r_2 \in R\ \text{inverse}(r_1, r_2) \Leftrightarrow (\forall e_1, e_2 \in E,\ r_1(e_1, e_2) \Rightarrow r_2(e_2, e_1))$
$\forall r_1, r_2 \in R\ \text{sameAs}(r_1, r_2) \Leftrightarrow (\forall e_1, e_2 \in E,\ r_1(e_1, e_2) \Rightarrow r_2(e_1, e_2))$

Similarly, application domain based axiomatic relations are used to state axioms and rules that are used to deduce further knowledge for reasoning. A statement that says "*under normal condition, a patient is always treated by the same doctor"* can be considered as an axiom (assumption) in a medical domain. Given this assumption, we can create another domain based deduction rule as follows:

$\forall d\ \text{instanceOf Doctor}, p\ \text{instanceOf Patient}: \text{hasDoctor}(p,d) \wedge \text{engagedInActivity}(p, \text{takeTreatement}) \Rightarrow \text{engagedInActivity}(d, \text{giveTreatement})$

We can associate some degree of accuracy to such axioms or rules as metadata. *Metadata* is data about data. Metadata in context modeling is important to associate quality, precision, source, time stamps and other information to the context data. Such information is important to prepare the context data for reasoning and decisions. In the EHRAM model, metadata information is a relation that describes another relation instance.

For example, if we are given context information that says *"patient is located in the garden"*, we can then make other statements about this statement to answer questions like: *Who* reported this information? *Which* service is used to report this information? *When* did it happen? *How* accurate is the information? *Why* is the subject in this situation? *What* will happen next? Etc.

3.5 From the EHRAM conceptual model to UML

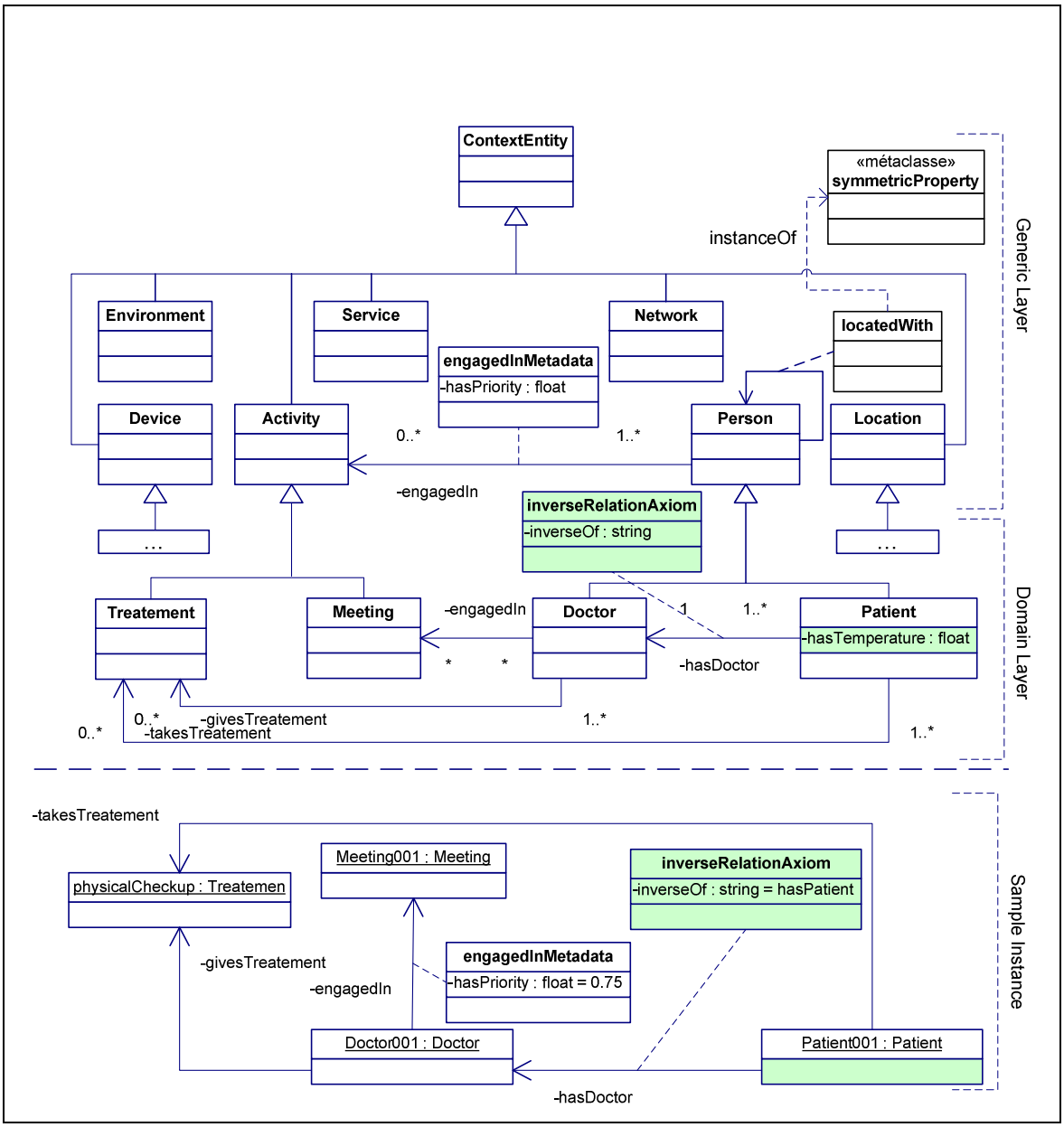


Figure 3-4: UML representation of the EHRAM model

Now, we try to see how we can develop a formal model to represent EHRAM structure as a generalized model based on its components described above. We use the Unified Modeling Language (UML) to formalize EHRAM as a conceptual context representation model. UML [Jacobson99] is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system. We will also show limitations of UML as a tool to represent EHRAM for modeling context data and suggest the necessary adjustments for improvement.

Incremental mapping of the concepts in EHRAM to the UML is given as follows. A UML class diagram based on this mapping is given in Figure 3.4.

- *Entity* in EHRAM can be represented as a UML *class*.
- The concept of *hierarchical* relation in EHRAM can be represented as *generalization* relationship in the UML.
- *Entity relations* in EHRAM can be represented as *association relationships* in UML, and attribute relations in EHRAM can be represented using *attributes* in the UML class.
- *Axiomatic relations* in EHRAM can be represented as *association classes* in the UML. The concept of metaclass can also be used to represent axiomatic properties like *symmetric property*, *inverse property*, etc... As indicated in the diagram, an entity relation *locatedWith* is marked as an instance of the symmetric property class.
- *Metadata* in EHRAM can be represented using *association classes* in the UML.

Among the limitations of using UML as modeling tool to represent the EHRAM modeling components are: attribute relationship, context metadata and axiomatic relations cannot be adequately represented in the UML. Representing axiomatic relations in the EHRAM model as association classes in the UML has the limitation of repeating the same set of axiomatic relations for every occurrence of instance of the entity. For example, the inverse axiomatic relation, *hasPatient* that is defined on *hasDoctor* using association class should be independent of every instance of the association but refer to the association it self. Similarly, representing attribute relations in EHRAM as attributes in the UML class also lacks a means to add metadata about the attribute. For example, the attribute *hasTemperature* itself may need a metadata like accuracy, time, etc.

Some of the limitations of UML to represent the EHRAM meta-model can be overcome by using extended UML features like *meta-classes* and *association-classes* (for example, see the definition of the property-class *locatedWith* and the meta-property-class *symmetric* defined in the generic layer of Figure 3.3). However, we still have a limited support for the representation of the semantic aspect of context data. As a conclusion, UML modeling tools can be used to formalize representation of the EHRAM model only partly and therefore we continue investigating other formal methods.

3.6 EHRAM and relational models

Context and context metadata can be represented using notations and concepts in a binary relation. Binary relation R is any subset of a Cartesian product X⊗Y where X and Y are arbitrary sets. The sets X and Y are called the domain and range, respectively, of the relation. The statement (x, y) ∈ R is read "x is-R-related-to y", and is denoted by xRy or R(x,y). The order of the elements in each pair is important: if a ≠ b, then R(a, b) and R(b, a) can be true or false, independently of each other.

Based on this definition of *relations*, given the sets of context entities E and set of values V drawn from set of context entities and set of literal values, a relation R is the subset of the Cartesian product of the sets of E and V:

$$R \subseteq \{(e_i, v_j): e_i \in E,\ v_j \in V\}$$

We are interested in all meaningful set of statements,

$$(e_i, v_j) \in R \text{ that can also be represented as:}$$
$$\{R(e_i, v_j): (e_i, v_j) \in R\} \text{ or in a more linear form}$$
$$\{(e_i, R, v_j): (e_i, v_j) \in R\}$$

This triple can be used to define a context (C) as follows:

$$C \equiv \{(e_{i,k},\ r_k,\ v_{j,k}) : e_{i,k} \in E_k,\ r_k \in R,\ v_{i,k} \in V_k\}$$

This can be extended to define context and context metadata CM using basic context C, meta-relation RM and meta-value VM as follows:

$$CM \equiv \{(c_i,\ rm_k,\ vm_j) : c_i \in C,\ rm_k \in Rm,\ vm_j \in Vm\} \text{ Or}$$
$$CM \equiv \{((\ e_i,\ r_k,\ v_j),\ rm_l,\ vm_p): e_i \in E, r_k \in R, v_j \in V, rm_l \in RM, vm_p \in VM\}$$

Basic context data:

```
(Schedule, isA, Service),(TimeTable, instanceOf, Schedule),
(Meeting, isA, Activity), (Meeting005, instanceOf, Meeting),
(Meeting005, hasStartTime, #200611081400)
(Doctor, isA, Person),(Pascal, instanceOf, Doctor)
(Pascal, isEngagedIn, Meeting005)
```

Context with metadata:

```
((Pascal, isEngagedIn, Meeting005), hasSource, Agenda)
((Pascal, isEngagedIn, Meeting005), hasPrecision, xx%)
```

N-ary relation (relation of degree N) as a base of relational database model inherits its properties from binary relations. A detailed discussion on mapping of the EHRAM model to the entity relationship (ER) model of the relational database is presented in chapter 4.

As a conclusion, relational models can be used to represent the entity, hierarchy, relations and metadata components of EHRAM model. They can also be extended some how to represent axioms in the EHRAM model using definitions like *(locatedWith, is, symmetric), (locatedWith, is, transitive), (owns, inverseOf, hasOwner)*, etc. This however is not sufficient to represent fully the semantic aspect of the context data represented in EHRAM. Therefore, we continue to investigate other formalism.

3.7 EHRAM and the RDF data model

RDF models have been in use to represent semantic metadata in different application domains. The work by [Bouzeghoub04] uses RDF semantic description model to allow the reuse and assembling of learning objects that represent pedagogical materials available on the web. The core element is the representation of semantic metadata that allows description of domain model, user model and learning object model. In this section, we will investigate RDF and its extensions to build a generic context meta-model.

We start with setting equivalence between terminologies in EHRAM and that of the RDF model. The primary characteristic of a context data is that it possesses an actor or a *subject* (an entity). The context value defined on the *subject* is expressed in terms of multiple properties. In our subsequent discussion, we use the terms *predicate* and *object* to represent the situation of the subject with respect to a specific property. This convention goes with the RDF-triple representation formalism, *<subject, predicate, object>*, which in turn maps to all types of relations in the EHRAM model (R and H).

Additional context metadata can also be included as part of the context data. In addition to the subject, predicate and object triples, context modeling requires context metadata to extend the context model towards historic, probabilistic, or confidence-carrying model. Such attributes are meaningful only when thought of as referring to a particular instance of the triple, not to each individual element.

To describe this situation we use RDF and its extension called RDF reification [W3C07]. Reification is used to represent facts that must then be manipulated in some way; for example, to compare logical assertions from different witnesses to determine their credibility. The message "*John is six feet tall*" is an assertion of truth that commits the sender to the fact, whereas the reified statement, "*Mary reports that John is six feet tall*" defers this commitment to Mary.

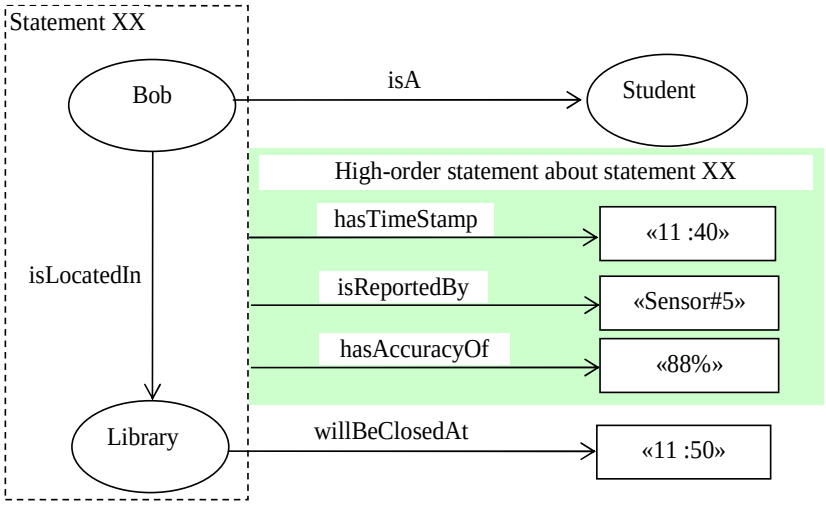


Figure 3-5: Context metadata represented using reification

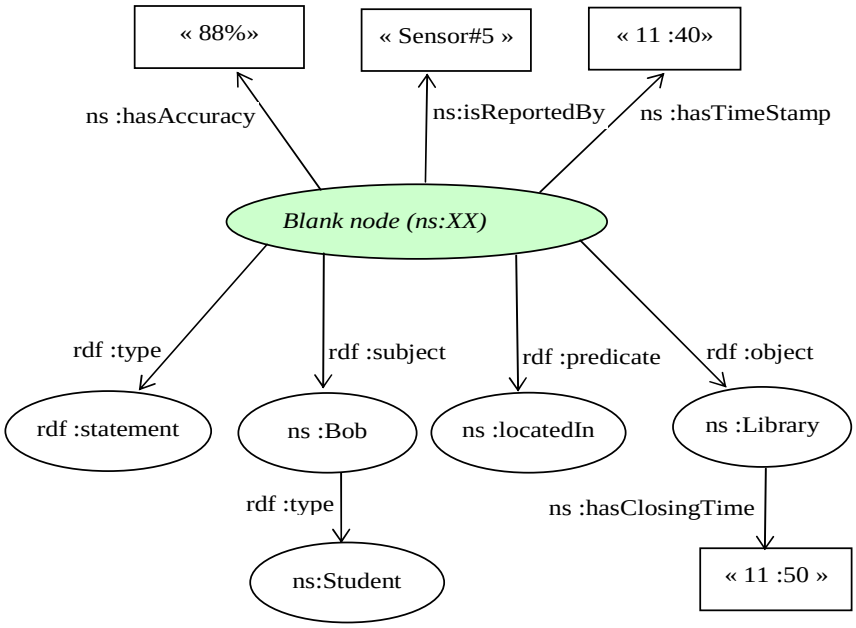


Figure 3-6: RDF data model for the reified context data

In the same way, a reified RDF data contains each original statement as a resource and the other additional statements made about it. The four properties used to model the original statement as the RDF resources are, *subject, predicate, object and type.* A new resource with these four properties represents the original statement and can be used as the subject or object of other statements with additional statements made about it.

Figure 3.5 shows demonstration of context metadata representation using statement reification. The figure show an example triple statement: *"Bob is located in the Library"*. This statement can be reified by additional meta-information like *"is reported by sensor #5", "has accuracy of 88%", "has occurred at 11:40 today",* etc. Figure 3.6 shows an

equivalent graphical representation of the RDF data model for the reified context data. Figure 3.7 shows an abridged RDF data model for this context metadata example. The RDF reification principle, therefore, can be used to represent additional context attributes to the basic context triples. RDF is one of the major building blocks in a formalism to represent ontology and ontology has features for defining and representing axioms. Therefore, axioms can be represented using RDF/OWL formalisms. Details on mapping EHRAM to ontology will be given in chapter 4.

```
//Axioms, context metadata and rdf reification
<owl:Class rdf:ID="Context"/>
<owl:Class rdf:ID="Person">
    <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>
<owl:Class rdf:ID="Location">
    <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>
<owl:Class rdf:ID="Student">
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Library">
    <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">   //Axiom
    <rdfs:domain rdf:resource="#Context" />
    <rdfs:range rdf:resource="#Location" />
    <rdf:type rdf:resource="&owl;TransitiveProperty" />
</owl:ObjectProperty>
<coca:Bob rdf:type coca:Student/> //original statement
<coca:DocINSA rdf:type coca:Library/>
<coca:Bob coca:locatedIn coca:DocINSA/> //original statement
  <coca:XXrdf:type resource=rdf:Statement/> //reification on XX
  <coca:XX rdf:subject rdf:resource=coca:Bob/>
  <coca:XX rdf:predicate rdf:resource=coca:locatedIn/>
  <coca:XX rdf:object rdf:resource=coca:DocINSA/>
<coca:XX coca:reportedBy coca:Sensor#5/> // Metadata using XX
<coca:XX coca:hasTimeStamp>
    <timeStamp rdf:datatype="&xsd;string"> 2007051140 </timeStamp>
</coca:XX>// using reified XX
<coca:XX coca:hasAccuracy coca:high”/>
```

Figure 3-7: RDF/OWL data model for context, axiom and metadata representation

3.8 Summary

The future of our computing environment will be highly affected by a range of large and small computing devices that will present in our vicinity. Unless properly controlled, the interference of such devices and services with our personal roles and institutional policies might sometimes have a negative impact. Such a control mechanism should be automated in such a way that users’ involvement is minimal. This can be achieved by building an intelligent (context-aware) component that acts depending on the situation or the context of

the person, the devices, the environment or other related elements. Hence, a formal definition and representation method for context data is important.

In this chapter, we have presented our context representation model named as EHRAM model. The name EHRAM is coined from its building blocks; entities, hierarchy of entities, relations defined on entities, axioms and context metadata. A generic EHRAM graph is used to represent an abstract conceptual representation of the EHRAM model. We have also shown different ways of context representation in the EHRAM model: using UML, binary relations, RDF and its reifications. All representation methods have minor limitations when representing metadata and axiomatic relations. A more comprehensive approach that deals with serialization and mapping of the EHRAM model into standard data management structures that supports the storage and processing of both the context data and the context semantics will be presented in chapter 4.

The advantage of using EHRAM can be summarized as follows:

- Components of EHRAM model (entity, hierarchy, relations, axioms and metadata) are derived directly from the definition of the context. This makes it simple and natural way of abstraction and conceptualization of context data and its semantics in the form of axioms and metadata.
- We have shown that different data representation formalism like UML, binary relation with its extended meta-form, and RDF model can be used to convert the EHRAM conceptual model into a concrete data representation formalism with some limitations.
- EHRAM is scalable to context data of any type and complexity.

Chapter 4 CONTEXT MANAGEMENT: THE HCOM MODEL

4.1 Introduction

The EHRAM conceptual context representation model needs serialization to store, use, transmit and reuse context data. We cannot fully manage context data using standard database management principles because in relational database, for example, the meaning of the data is in the "head" of the user. Management of context data requires not only processing the data itself but also processing the meaning of the data. This requires an approach that not only deals with the content of the data (context data) but also semantics of the data (context knowledge).

So what is context knowledge and context data? How do they differ? [Ackoff89] has presented knowledge and data as part of classification of the content of human mind. According to Ackoff:

> "*data is raw fact. Information is data that has been given meaning, useful or not, by way of relational connection in the data. Knowledge is the appropriate collection of information, such that its intent is to be useful. Understanding is the process by which one can take knowledge and synthesize new knowledge from the previously held knowledge.*"

Context-aware services, aims to possess *understanding* in the sense that they are able to synthesize and infer new *knowledge* from previously stored *information* and *knowledge*.

Extending the above definitions, *context data* is, therefore, a raw fact that simply exists and does not have a meaning of it self. *Context knowledge* is the collection of useful context information derived from the interpretation of context data and related concepts. The derived context knowledge has useful meaning, but it does not provide, by itself, inference of further knowledge. Context knowledge, therefore, is more about context semantics that will be dealt with in the subsequent sections of the chapter. The *understanding* aspect that

deals with the reasoning, inference and decision support in the context-aware service will be presented in chapter 5.

From our observation, ontology approach is becoming a common method to deal with data semantics. Ontology however is not well suited to support efficient and optimized query processing and transaction of voluminous data. In this chapter, we try to investigate how our EHRAM conceptual context representation model can be mapped to the relational data model and to the ontology data model. We show the pros and cons of both approaches. We finally propose a novel hybrid approach for modeling context management called HCoM model. HCoM uses both ontology and relational models to process context data and semantics separately and then combine the results together for inference and reasoning.

4.2 Mapping from EHRAM to ER model

Relational database is a stable model that is used in a wide range of database management applications. In the EHRAM model, context is represented by combination of entities, hierarchy, relations, axioms and metadata. In relational database, entity relationship (ER) model is used to represent entities, attributes and relationships. Can we therefore have a lossless mapping from EHRAM to relational database model?.

A step-by-step mapping algorithm from EHRAM model components to relational schema is given as follows:

Step 1. Collect all context entities in the EHRAM model and create a relational table with the following attributes. This represents entities and their hierarchy.

- An attribute that stores name of context entities (*cEntity*).
- An attribute that stores name of the entity one step above in the hierarchy (*isa*).
- An attribute that stores whether an entity is in the generic or domain layer (*layer*).

table tblEntity (CEntity, IsA, Layer), Primary Key (CEntity)

Step 2. Collect all non hierarchical relations in the EHRAM model (relations other than isa and isInstanceOf) and create a relational table with the following attributes.

- An attribute that stores name of relations (*Relation*)

- An attribute that stores persistence of the relation when applied to an entity. There are two options: static and dynamic. Values of relations with static persistence are stored in the persistent context repository while values of those with dynamic persistence are stored only temporarily for immediate use (*Persistence*)

  tblRelation (Relation, Persistence), Primary Key (Relation)

Step 3. Collect all relation instances defined on entities in the EHRAM model and create a relational table with the following attributes.

- An attribute that stores name of relations (relation).
- An attribute that stores name of entity on which the relation can be applied (*CEntity*).
- An attribute that stores source of value as a context entity or as a "Litral" (*ValueFrom*).

  tblRInstance (Relation, CEntity, ValueFrom), Primary Key (CEntity,Relation), Foreign Key (CEntity) references tblEntity (CEntity), Foreign Key (Relation) references tblRelation (Relation)

Step 4. Collect all context instances in the EHRAM model and create the following relational table. This represents entity instances.

- An attribute that stores name of instances (*EInstance*)
- An attribute that stores types of the instances (*CEntity*)

  tblEInstance (EInstance, CEntity) Primary Key (EInstance), Foreign Key (CEntity) references tblEntity (CEntity)

Step 5. Collect all relations defined on instances in the EHRAM model and create a relational table with the following attributes. Include also all metadata information defined on each context instance. Metadata, in our case, includes timestamp when the context instance has occurred, source of the context data and precision of the context data. This represents context instances.

- An attribute that stores name of entity instance (*EInstance*)
- An attribute that stores name of relation (*Relation*)

- An attribute that stores value of the relation after applied to the instance (*Value*)
- An attribute that stores context timestamp (*Timestamp*)
- An attribute that stores context source (*Source*)
- An attribute that stores context precision (*Precision*)

tblCInstance (EInstance, Relation, Value, TimeStamp, ContextSource, ContextPrecision), Primary Key (EInstance,Relation,Value,TimeStamp, ContextSource), Foreign Key (EInstance) references tblEInstance (EInstance), Foreign Key (Relation) references tblRelation (Relation)

Step 6. Collect all axioms in the EHRAM model and create a relational table with the following attributes.

- An attribute that stores relation (relation)
- An attribute that stores axiom (axiom)
- An attribute that stores value of axiom on this relation (aValue)

tblAxiom (Relation, Axiom, AValue), Primary Key Relation,Axiom,AValue), Foreign Key (Relation) references tblRelation (Relation)

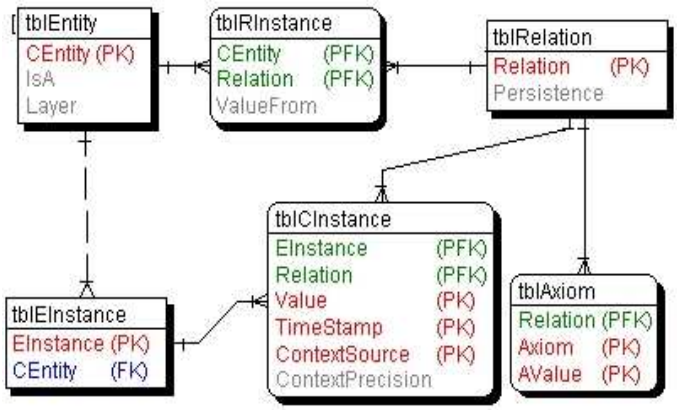


Figure 4-1: Generalized schema of the RCDB

A relational context database (RCDB) schema for the EHRAM that follows from the above systematic mapping is given in Figure 4.1. For representation of extended form of components of axioms in EHRAM, we may consider the object relational models in which set of axioms and metadata can be treated as object properties. However, we are more interested to use the lightweight relational model for its well-established data representation, storage, querying and presentation tools.

EInstance	Relation	Value	TimeStamp	ContextSource	ContextPrecision
Bob	takeTreatement	physicalExam	200707121035	medical chart	85%
Pascal	engagedIn	weeklyMeeting001	200707120900	Schedule	95%
Pascal	giveTreatement	physicalExam	200707121040	schedule	80%
Pascal	owns	SPhone01	*	profile	100%

CEntity	IsA	Layer
Activity	ContextEntity	Generic
Device	ContextEntity	Generic
Doctor	Person	Domain
Meeting	Activity	Domain
Patient	Person	Domain
Person	ContextEntity	Generic
Sphone	Device	Domain
Treatement	Activity	Domain

Relation	Axiom	AValue
owns	inverseOf	hasOwner

EInstance	CEntity
Bob	Patient
Pascal	Doctor
PhysicalExamination	Treatement
SPhone001	Sphone
WeeklyMeeting001	Meeting

CEntity	Relation	ValueFrom
Doctor	engagedIn	Activity
Doctor	giveTreatement	Treatement
Patient	takeTreatement	Treatement
Person	Owns	Device

Relation	Persistance
engagedIn	Dynamic
giveTreatement	Dynamic
owns	Static
takeTreatement	Dynamic

Figure 4-2: Sample demonstration context data for from medical application

Data from a domain of application can be entered and processed in the relational context data schema. Figure 4.2 shows a sample context data for demonstration. This data is extracted using the above set of algorithm from our previous medical domain example given under the EHRAM model representation. On this data set, we can use standard SQL query to retrieve context information and pass the result for reasoning and decision. For example, *to check if there is any dynamic context data with more than 80% accuracy that has occurred within the last 20 minutes (given current time is 2007 07 12 10:45).* The corresponding query can be:

```
SELECT tblCInstance.EInstance, tblCInstance.Relation,
tblCInstance.Value
 FROM tblRelation INNER JOIN ((tblEntity INNER JOIN
   tblEInstance INNER JOIN tblCInstance ON
   tblEInstance.EInstance = tblCInstance.EInstance) ON
   tblEntity.CEntity = tblEInstance.CEntity) INNER JOIN
   tblRInstance ON tblEntity.CEntity =
   tblRInstance.CEntity) ON (tblRelation.Relation =
   tblRInstance.Relation) AND (tblRelation.Relation =
   tblCInstance.Relation)
 WHERE ((tblCInstance.TimeStamp>"200707121025") AND
   (tblCInstance.ContextPrecision>"80%")AND
   (tblRelation.Persistance="Dynamic"));
```

The result of this query on our particular data set is, "Bob *takeTreatement* PhysicalExam". Collection of such type of aggregated context data can be used in the reasoning and decision process.

As a conclusion, we can represent the basic EHRAM elements, entity, relations, hierarchy but only part of metadata axioms.

4.3 The need for semantic context management model

Considering the situation of staff members' (Ben, Dan and Rita) tea break scenario in the following table, a simple query (*select Subject from table.context where predicate= "isLocatedIn" and Object= "Room-305"*) selects "Ben" as an output. But in reality, if the information in the table is given to a human assistant who knows, by common sense, that the terms "Office" and "Room" are synonymous in the domain of interest, s/he will respond "Ben" and "Rita" to the query. Moreover, a human assistant can also deduce that Ben and Rita are now together. However, incorporating such semantic interpretation of data using standard database schema is not a straightforward task.

Table 4-1: Demonstration on the need for context semantics

Subject	Predicate	Object	Time
Ben	isLocatedIn	Room-305	2006022310
Dan	isLocatedIn	Room-301	2006022310
Rita	isLocatedIn	Office-305	2006022310
…	…	…	…

This simple example demonstrates the need for a context model that describes concepts, concept hierarchies and their relationships. We chose ontology and the OWL language for this purpose. For the concepts, *Office* and *Room,* in Table 4.1, for example, we can use the *owl:sameAs* property that defines them as the same concepts. Similarly, the property concepts, *together* and *coLocatedWith,* can also be defined as the same concepts using OWL as follows:

```
//similarity between classes and properties
<rdf:Description  rdf:about= "#Office">
   <owl:sameAs rdf: resource = "#Room">
</rdf:Description>
<rdf:Description  rdf:about= "#together">
   <owl:sameAs rdf: resource = "#coLocatedWith">
</rdf:Description>
```

We can also define the concept that coL*ocatedWith* is *symmetric*, which means if *"X coLocatedWith Y"* then we can state that *"Y coLocatedWith X"*. We can enhance the potential of semantic context reasoning using additional user defined rules like, for example:

"if $user_1$ is located in a $room_N$ and $user_2$ is also located in roomN then conclude that they are coLocatedWith each other or according to the above similarity definition they are together".

This rule can be represented as follows:

```
//User defined Rule
   [ruleR: ?user1 nsp:locatedIn ?roomN)
   (?user2 nsp:locatedIn ?roomN)
        ->?user1 nsp:coLocatedWith ?user2)]
```

4.4 EHRAM and semantic ontology for context management

The expressive power, hierarchical organization, formality, standard, support for efficient reasoning, support for programming abstraction and interoperability are among the attractive features of ontology in context modeling. A hierarchy of ontology classes can be used to organize context entities, concept hierarchies, relationships, axioms and context instances.

Ontology can be used to represent semantics, concept relationships and axioms in the context data. Context ontology is formed by the merger of the generic context ontology that describes domain independent contexts and the domain specific context ontology. *Context* instances may come from context databases or sensors. *Axioms* represent derivation rules that are used by context-aware systems to derive decisions and conclusions about the actions that follow. *OWL* Web Ontology Language [OWL07] is becoming a natural language in ontology representation due to several reasons. It is designed for use by applications that need to process the content of information instead of just presenting information to humans.

OWL is a W3C recommendation that employs web standards such as RDF and XML schema for information representation and exchange. OWL provides the necessary semantic interoperability tools to allow communications between context-aware systems. It also provides language tools for high degree of inference making by providing additional built-in vocabulary along with a formal semantics to define classes, properties, relations and axioms. See annex for detail.

4.4.1 **Representing EHRAM context ontology using directed graph**

Using a generic ontology modeling approach, context management consists of the following two basic components, the base ontology and the domain ontology. The base ontology part is defined based on our context descriptors (from generic layer of the EHRAM context representation model) while the domain ontology part is dependent on domain specific sub-descriptors (from domain layer of EHRAM).

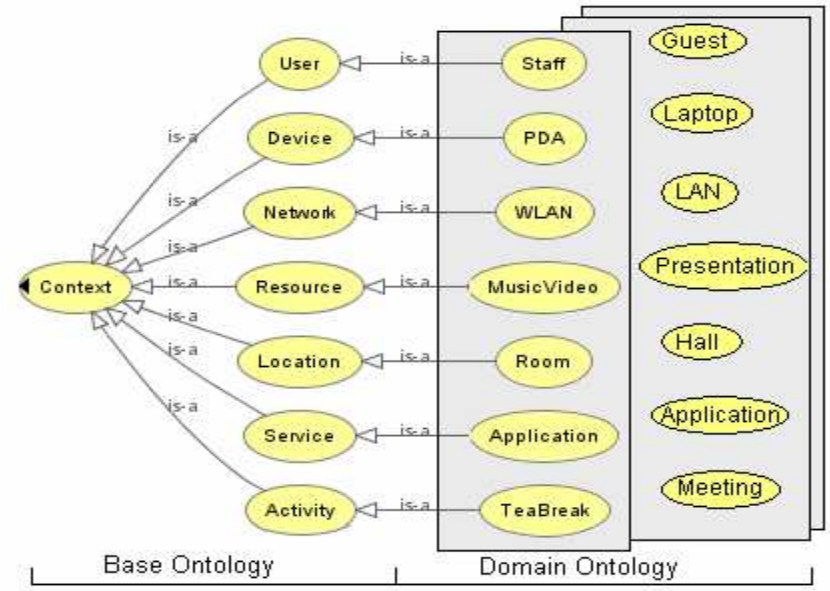


Figure 4-3: Simplified ontology graph showing base and domain ontology classes

Figure 4.3 is an example of ontology graph representation showing base and domain ontology classes. We use a directed graph G = (V, E) that consists a finite set V = {v_1,… v_n} of nodes (or vertices) that represent entities in the EHRAM model and a finite set E = {e_1, … e_n} of edges (or arcs). Each e ∈ E is an ordered triple (v_j, r, v_k) of $v_j \in V$, $r \in R$, and $v_k \in V$ where v_j and v_k are called the tail and the head of the edge e respectively. R represents hierarchical relations in EHRAM. We have a restriction that a node v_0 = "ContextEntity" should exist only ones as a tail of an edge in E and it should never exist as a head.

The EHRAM context ontology G_{co} is then a transpose $G^T=(V,E^T)$ of the graph G where G^T is the same graph G with all the arrows reversed:

$$G_{co} = G^T = (V, E^T)$$

All tails of the edges in E^T whose head is the term "*ContextEntity*" are considered sets of basic context descriptors (B) and are represented as base classes in the generic layer of the

ontology. Similarly all tails of the edges in E^T whose head is not the term "*ContextEntity*" are considered as sets of domain context descriptors (D).

$$\forall e \in E^T, \quad e = (v, r, "ContextEntity") \Leftrightarrow v \in B$$

$$\forall e \in E^T, \quad e = (v, r, v_c) \wedge (v_c \neq "ContextEntity") \Leftrightarrow v \in D$$

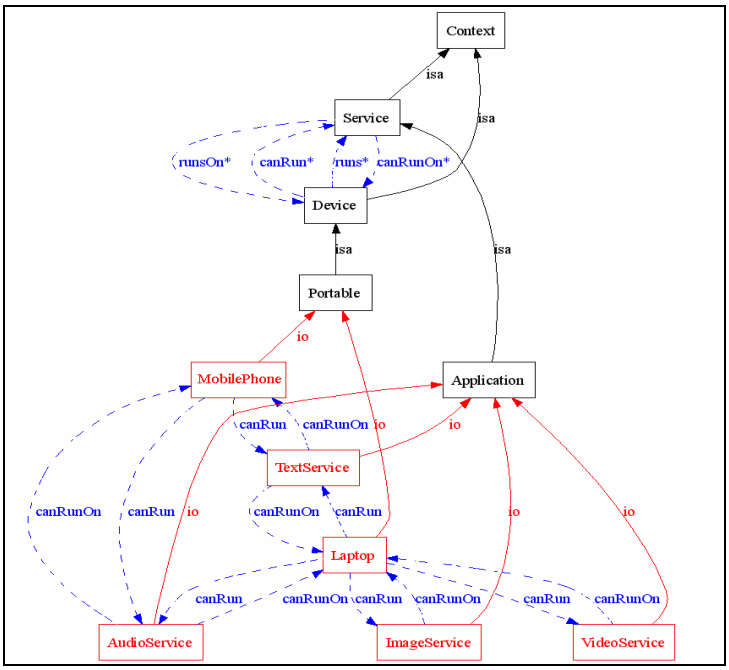


Figure 4-4: Ontology graph for sample base and domain specific classes and instances

Figure 4.4 is ontology graph with other types of relations from an application domain. In addition to the inherited properties from the base class, we can also define domain specific properties for the subclasses. Subclass hierarchy can also be extended further down to accommodate the structure of context data in the specific domain.

4.4.2 EHRAM model and the OWL language

In addition to the standardization of the structure of the context representation, ontology provides semantic descriptions and relationships of entities. Using ontology, we can also perform deeper knowledge analysis by defining domain specific rules.

For example, to define the similarity axiom between the concepts *ownerOf* and *owns*, we can use *owl:sameAs* property. Similarly, the symmetric axiom on the concept *coLocatedWith* can be defined using *owl:symmetricProperty* and the inverse relationship property between *ownerOf* and *ownedBy* can be defined using *owl:inverseOf* property.

```
<?xml version="1.0"?>
  <rdf:RDF ……
  <owl:Class rdf:ID="#ContextEntity"/>
  <owl:Class rdf:ID="Activity"> <rdfs:subClassOf
      rdf:resource="#ContextEntity"/> </owl:Class>
  <owl:Class rdf:ID="Person"> <rdfs:subClassOf
      rdf:resource="#ContextEntity"/> </owl:Class>
  <owl:Class rdf:ID="Device"> <rdfs:subClassOf
      rdf:resource="#ContextEntity"/> </owl:Class>
  <owl:Class rdf:ID="Patient"> <rdfs:subClassOf rdf:resource="#Person"/> </owl:Class>
  <owl:Class rdf:ID="Doctor"> <rdfs:subClassOf rdf:resource="#Person"/> </owl:Class>
  <owl:Class rdf:ID="Phone"> <rdfs:subClassOf rdf:resource="#Device"/> </owl:Class>
  <owl:Class rdf:ID="Meeting"> <rdfs:subClassOf rdf:resource="#Activity"/> </owl:Class>
  <owl:ObjectProperty rdf:ID="isColocatedWith">
    <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Person"/> <owl:Class rdf:about="#Device"/>
        </owl:unionOf> </owl:Class> </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Device"/> <owl:Class rdf:about="#Person"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <owl:inverseOf rdf:resource="#isColocatedWith"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="isOwnedBy">
    <rdfs:domain rdf:resource="#Device"/> <rdfs:range rdf:resource="#Person"/>
    <owl:inverseOf>   <owl:ObjectProperty rdf:ID="Owns"/>    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="isEngagedIn"> … … …
  <owl:DatatypeProperty rdf:ID="hasBodyTemp"> … … …
  <owl:FunctionalProperty rdf:ID="hasStartTime"> … … …
  <owl:FunctionalProperty rdf:ID="hasMemory">… … …
  <Phone rdf:ID="Smartphone0095">
    <hasMemory rdf:datatype="http://www.w3.org/2001/XMLSchema#long">512</hasMemory>
  </Phone>
</rdf:RDF>
```

Figure 4-5: An excerpt of the EHRAM based ontology for the medical application

Table 4-2: Mapping between EHRAM model and ontology

EHRAM model		Ontology
(E)ntity		owl:class
(H)ierarchy Relations		rdfs:subClassOf, rdfs:superClassOf, rdf:type
(R)elations	Entity	owl:objectProperty
	Attribute	owl:dataTypeProperty
(A)xioms		properties (owl:TransitiveProperty, owl:inverseOf, …) restrictions(hasValue, hasMinCardinality, someOf, ...)
(M)etadata		rdf reification

Table 4.2 shows concept matching that can be used to create mapping between the EHRAM model and the owl ontology representation. Part of the ontology mapping of the EHRAM model for context data from the medical application using the concept-mapping schema in the table is given in Figure 4.5.

4.4.3 The GCoM model

For the interpretation, representation and management of context data, we propose an ontology based Generic Context Management (GCoM) model. GCoM is based on the above mapping relationship between the EHRAM model and ontology. GCoM consists of three basic inputs: context ontology, context data and context related rules. The output from the GCoM is aggregated context ontology ready for reasoning and decision by any context-aware service application. The components of the layered GCoM architecture are shown in Figure 4.6.

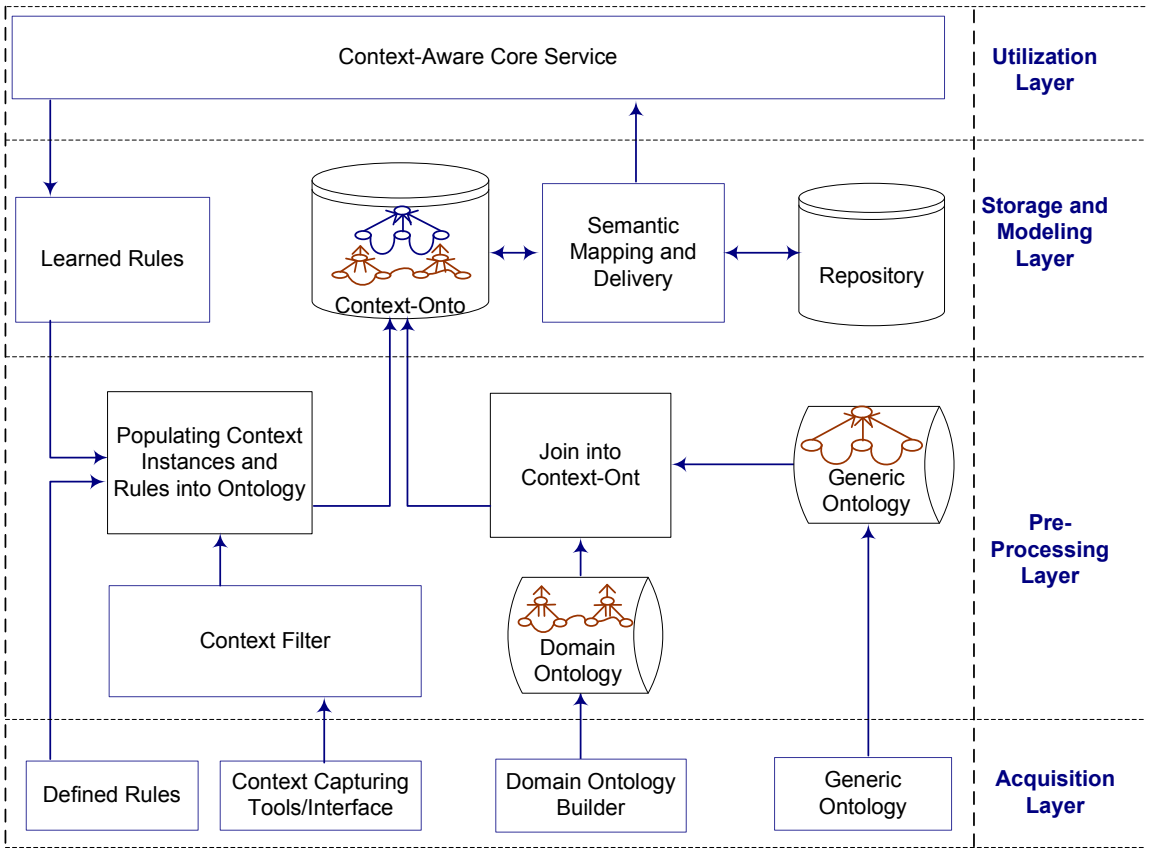


Figure 4-6: Architecture of the GCoM model

Ontology represents semantics, concepts and relationships in the context data. It is formed by the merger of ontology that describes domain independent generic contexts and domain specific contexts. *Context* data represents instances of context that may exist in the

form of profiled data stored on a disk file or in the form of context instances obtained from the sensors. *Rules* represent derivation axioms that are used by context-aware systems to reason out and derive decisions about the actions that follow. These rules have two sources; rules that are explicitly given by the users through the user interface and rules that are implicitly *learnt* by the system itself. *Mapping and populating* service is responsible to put the three data sources together in a semantically coherent manner. *Context-onto* is a repository of aggregated context information. We also have *Semantic mapping and delivery* module that is responsible for mapping and aggregation to deliver aggregated context data that is ready for use by context-aware systems.

```
Ontology representation
<rdf:RDF …….
  <owl:Class rdf:ID="Student">
   <rdfs:subClassOf> <owl:Class rdf:ID="User"/> </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Library">
   <rdfs:subClassOf> <owl:Class rdf:about="#Location"/>
</rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="ownedBy">
   <rdfs:range rdf:resource="#User"/>
   <rdfs:domain rdf:resource="#Device"/>
   <rdf:type
rdf:resource="http://www.w3.org/.../owl#FunctionalProperty"/>
    <owl:inverseOf> <owl:ObjectProperty rdf:ID="ownerOf"/>
</owl:inverseOf>
  </owl:ObjectProperty>
  <Student rdf:ID="Bob">
   <ownerOf>
     <PDA rdf:ID="PDA001">
       <hasScreenSize

     rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Medium
       </hasScreenSize>
     </PDA>
   </ownerOf>
   <ownerOf rdf:resource="#Cellphone001"/>
  </Student>
…....
</rdf:RDF>
```

Figure 4-7: Part of the context ontology for the ringing tone scenario

Demonstration on the GCoM model components can be given using a cell phone ringing tone management service example based on a scenario of a university regulation. To comply with the regulation, students must have their cell phones set to non-disturbing modes during different activities: attending lectures, consultation with their professors, in libraries, etc.

Students therefore need to have their phones automatically switched to silent mode or vibrating mode while in the library, attending lectures, or discussing with their professors and switch back to ringing mode when they are engaged in none of these activities. They also like to use a decent ringing tone when in side the campus and a musical ringing tone when outside the campus. Figure 4.7 is OWL representation of part of the ontology for the phone ringing management scenario.

```
Context representation
Profiled context defined in the Ontology
   <nameSP:Bob sys:type nameSP:Student/>
   <nameSP:CellPhone001 sys:type nameSP:Phone/>
   <nameSP:PDA001 sys:type nameSP:PDA/>
   <nameSP:Bob nameSP:owns nameSP:PDA001/>
   <nameSP:Bob nameSP:owns nameSP:CellPhone001/>
   <nameSP:ClassRoom001 sys:type nameSP:ClassRoom/>
   <nameSP:Semantic-Theory sys:type nameSP:Class/>

Case 1: Bob, has just entered in ClassRoom001 to attend a lecture
   <nameSP:PDA001 nameSP:locatedIn nameSP:ClassRoom001/>
   <nameSP:XX rdf:type resource=rdf:statement/>
         <nameSP:XX rdf:subject resource=nameSP:Bob/>
         <nameSP:XX rdf:predicate resource=nameSP:isLocatedIn/>
         <nameSP:XX rdf:object resource=nameSP:Library/>
   <nameSP:XX nameSP:hasTime “200703251030”/>
   <nameSP:Bob nameSP:hasSchedule nameSP:Semantic-Theory/>
   <nameSP:Semantic-Theory nameSP:scheduledIn nameSP:ClassRoom001/>
   <nameSP:Semantic-Theory nameSP:startTime “200703251000”/>
   <nameSP:Semantic-Theory nameSP:endTime “200703251100”/>

Case 2: Bob is just getting out of the campus
   <nameSP:PDA001 nameSP:locatedIn nameSP:OutSideCampus/>

Case 3: Bob has just entered in the library reading room
     <nameSP:PDA001 nameSP:locatedIn nameSP:DocINSA/>
```

Figure 4-8: Context representation for the ringing tone scenario

Persistent data about static contexts (e.g. ownership relationship of persons to devices like telephone) can be stored in any standard database format that can be selectively populated as context instances into the ontology structure at runtime. Sensed context is to be communicated to GCoM using RDF/XML triple representation format and is then converted to the indicated representation (Figure 4.8) to make the data ready for reasoning. Domain specific rules for students’ explicit wishes in the scenario and context data expressed using Jena generic rule are given in Figure 4.9.

Rules representation

```
Rules derived or imported from ontology (implicit rules defined in
the ontology)

[OntoRule1: (?a gcom:locatedIn ?b) (?b gcom:locatedIn ?c) -> (?a
gcom:locatedIn ?c)] //transitive

[OntoRule2: (?a gcom:ownerOf ?b) -> (?b gcom:ownedBy ?a)]
//inverse

…..

Defined Rules
[locatedRule:(?device gcom:locatedIn ?location)
         (?device gcom:ownedBy ?person)
         -> (?person gcom:locatedIn ?location)
]
[libraryRule:(?student gcom:locatedIn gcom:Library)
        (?student gcom:owns ?phone)
         -> (?phone “switchMode” “silent”)

]
[classRule:(?student gcom:hasSchedule ?class)
        (?class gcom:isScheduledIn ?classRoom)
        (?class gcom:startTime ?t1)
        (?class gcom:endTime ?t2)
        ((?Student gcom:locatedIn  ?classRoom) gcom:hasTime ?t)

        (?t sys:greaterThan ?t1)(?t sys:lessThan ?t2)
        (?student gcom:owns ?phone)
        → (?phone “switchMode” “Vibrating”)
]
[meetingRule:(?student gcom:hasSchedule ?meeting)
        (?meeting gcom:scheduledIn ?meetingRoom)
        (?meeting gcom:startTime ?t1)
        (?meeting gcom:endTime ?t2)
        ((?student gcom:locatedIn ?meetingRoom) gcom:hasTime ?t)
        (?t sys:greaterThan ?t1) (?t gcom:lessThan ?t2)
        (?student gcom:owns ?phone)
        →(?phone “switchMode” “Silent”)
]
[campusRule:(?student gcom:locatedIn gcom:InCampus)
        (not classRule) (not meetingRule)( not libraryRule)
        //because InCampus subsumes ClassRooms, MeetingRooms and
        Library
        (?student gcom:ownerOf ?phone)
        →(?phone “switchMode” “DecentRingingTone”)
        ]
[xcampusRule:(?Student gcom:locatedIn OutSideCampus)
        (?student gcom:owns ?phone)
        → (?phone “switchMode” “MusicRingingTone”)
]
```

Figure 4-9: Rule representation for the ringing tone scenario

4.5 Limitations of relational and ontology approaches

In order to extend our EHRAM context representation model to a more generalized context management model, efficient storage and retrieval, ease of serialization and semantic support are the basic necessary features. In the EHRAM model, we have parts that represent the semantics of the data (context knowledge) and parts that represent the context instance (context data). For proper reasoning and decision in a context-aware computing environment, both context semantics and context data should be treated equally in the process of developing a context management model. In the relational approach, we have mapped our EHRAM model to six relational tables representing entity definition, entity instance, relation definition, relation instance, context instance and axiom. In the ontology approach, we have created a direct mapping of concepts in the EHRAM model to owl ontology representation. Both relational and ontology based approaches, when used for context management modeling, have their own pros and cons.

Ontology representation tools provide a widely accepted and formal representation of context semantics in order to interpret and reason about context information. Ontology tools, however, are only good at statically representing the knowledge in a domain. They are not designed for capturing and processing constantly changing information in dynamic environments in a scalable manner. Moreover, existing ontology languages and serialization formats are text-based (xml/rdf/owl) and therefore are not designed for efficient query optimization, processing and retrieval of large context data.

Table 4-3: Comparison of relational and ontology approaches on appropriateness

Necessary Features	Relational Approach	Ontology Approach
Semantic support	×	√
Ease of transaction (large data)	√	×
Query optimization	√	×
Reasoning support	×	√
Formality	√	√
Scalability	√	×

(Keys: √ appropriate × less appropriate)

Relational models, on the other hand, provide standard interfaces and query optimization tools for managing large and distributed context database or receive and send notifications on context changes. Relational models, however, are not designed for semantic

interpretation of data. Table 4.3 summarizes the appropriateness of both approaches in relation to the necessary features identified.

From the summary table, we can clearly see that both approaches have strong and weak sides with respect to the necessary features for context management modeling. In subsequent sections, we will see how we can combine the best of the two worlds into a hybrid context management model.

4.6 HCoM: Hybrid context management model

4.6.1 **HCoM model overview**

Our rational behind the need for the hybrid context model, is to distinguish the works of context data management and context knowledge management, process them separately and put the results together for better reasoning and decision support in a pervasive context-aware computing. We use ontology approach to manage context semantics and relational approach to manage context data. We name this combination a Hybrid Context Management (HCoM) model. HCoM model aims to combine the bests from the two worlds. It is an upgrade on our earlier ontology based generic context management model, GCoM. GCoM is based on the organization of context related data in the form of context, rules and ontology. Each has two separate sources; context data are either from the user interface or from sensors, rules are from the user interface or from the data-mining module. Ontology also has two sources; generic ontology that serves multiple domain of application and domain dependant ontology.

The selector/pruning module in HCoM provides a means to select and load only part of the large static context data that is accumulated over a period of time depending on who and where the user is, the intended activity to which the user is going to be engaged, devices available for use, institutional policies etc. It uses matching patterns gained through experience to identify relevant group of context data. Through analysis of history profiles, the filtering module assigns a numerical score to each class of entity in relation to a particular request instance. This score is used to determine whether the context entity class is relevant or not.

As indicated in SCOPES, *Semantic Coordinator over Parallel Exploration Space,* [Ouksel03], search space pruning facilitates an incremental construction of context

knowledge necessary to translate a query posed for a local database into equivalent remote database query for semantic reconciliation. In HCoM, we use this principle of pruning to limit the amount of context data in the reasoning space. The HCoM selection algorithm will be discussed in subsequent sections and the implementation and evaluation of both HCoM and GCoM will be given in chapter 6.

Loading only relevant data for reasoning minimizes the size of the reasoning space and reduces the unnecessary overloading of the reasoner to improve the overall performance of the context-aware service. It helps to overcome limitations of lack of scalability of most of the reasoning systems to the constantly increasing volume of reasoning resources in the pervasive environment.

4.6.2 HCoM architecture

Figure 4.10 shows components and functionalities that represent a layered architecture of the HCoM model. The core HCoM functionality is in the three middle layers in the architecture.

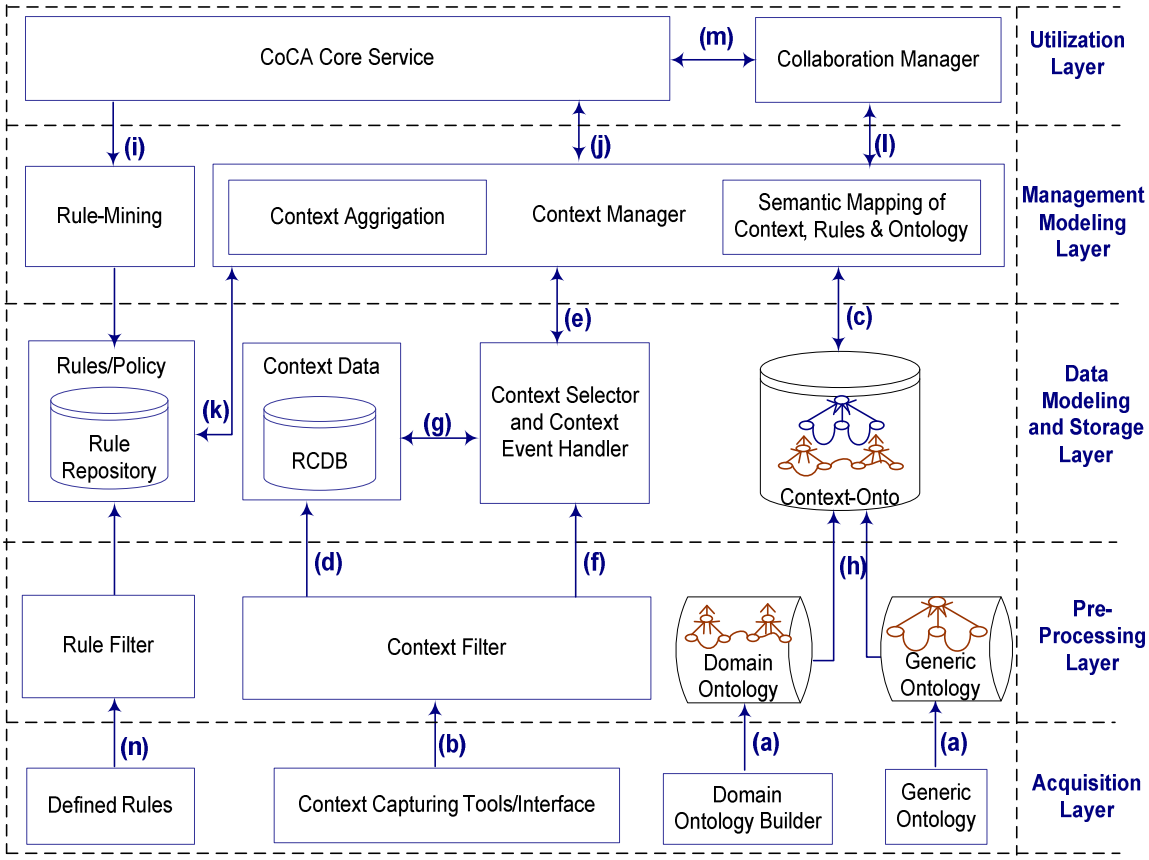


Figure 4-10: Architecture for Layered HCoM model

The arrows in the diagram represent flow of different set of data from or to each component. (a) ontology data, (b) context data, (c) ontology data to context manager (d) static context data, (e) relevant context data from repository and new context data from context filter to context manager, or preference data for selection from context manager, (f) filtered new context data, (g) query and retrieval of relevant static context data during initialization, (h) generic and domain ontology, (i) learned rules from context-aware modules, (j) context data model to CoCA core or preference data from CoCA core, (k) rules to the context manager, (l) learned rules (m) transaction for collaboration with CoCA core, (n) rule and policy data from rule capture interface.

4.6.3 **HCoM components**

HCoM consists of; RCDB, context-onto, context selector, context filter, collaboration manager, context manager and event notice handler, interfaces and data repositories. Description of these components is given as follows:

Context Filter

Context Filter receives a new context instance data that may be captured from hardware or software sensors and then validates and creates context log from which copy of static context instances are selected and added to the RCDB for future use. Context log is recreated every time we initialize the system and it populates new context data, both static and dynamic, into context-onto. In parallel, it also sends a context event notice to the CoCA service.

Figure 4.11 shows a pseudo-code of a context filter algorithm. This algorithm is activated any time the system is running so as to filter and decide if the context is useful depending on its reliability factor (lines 9-17). It also checks if the context is of static category (lines 19-22) that may need to be stored for future use or simply used for one time use. This can be decided based on the predicate used to describe the context. For example, context data defined using predicates like *ownedBy* are static and needs to be stored in the RCDB for future reference while those defined by predicates like *locatedIn* are just for one time use and may not need to be stored. Both category of context are sent to Context-Onto (line 23) for aggregation with existing reasoning resources. This algorithm also sends (line 24) a trigger message to the underlying core context-aware service (in this case CoCA) for further reasoning and proactive action.

```
1.  While (System_Is_Running)
2.  {
3.    newContext=null
4.    ContextBlock=new ContextClass()//context and all related data
5.    repeate until newContext!=null
6.    {
7.      newContext=ContextBlock.getNewContext()
8.    }
9.    reliabilityFactor=0
10.   if (ContextBlock.hasReliableSource()
11.   {
12.     reliabilityFactor=1
13.   }else
14.   {
15.     reliabilityFactor=ContextBlock.estimateSourceReliability()
16.   }
17.   if (reliabilityFactor> ContextBlock.reliabilityThereshold())
18.   {
19.     if (ContextBlock.hasStaticCategory())
20.     {
21.       ContextBlock.addContextToRCDB()
22.     }
23.     ContextBlock.addContextToContextOnto()
24.     ContextBlock.sendNoticeToCoCA()
25.   }else
26.   {
27.     ContextBlock.inValidContextError()
28.   }
29. }
```

Figure 4-11: Context Filter Algorithm

Context Selector

Context selector uses historic and current user information, devices available, institutional policies etc. to select and load only part of the context data from the repository into the reasoning space. Loading only relevant data minimizes the reasoning search space and reduces the unnecessary overloading of the reasoner to improve the overall efficiency of the reasoning process. It helps to overcome limitations of lack of scalability of most reasoning systems.

Rules and Policy

Rules in CoCA come from three different sources; rules defined by user, rules derived from organizational policies and rules derived from history data of past decisions using rule-mining module. *Rule-mining* module uses datamining tools [Hand01] to extracts useful rules from history of decisions and actions. It enhances self-governing and proactivity features of the system. For example, datamining tools can be used to learn the contexts under which a g student switches the ringing modes of his/her telephone. This means that the rules specified

above are to be derived by the system itself. Such rules are dynamic in the sense that their quality improves with time.

This is analogous to a "human assistant" assigned to help the student to change the phone-ringing mode. If, for example, a human assistant is given orders like "now switch to X mode!", "now to Z!", "now again to X!", "now to Y!" etc. while moving around with the student, the assistant can do some of these switching by himself after one week and with more accuracy after two or three weeks. Similarly, the rule-mining module is responsible to learn and propose users' wishes using the historical data. Such autonomous decision and action support helps the user not to be worried about the pervasive world routines.

Context-Onto

Context-Onto is created from the generic and domain layers of the EHRAM conceptual model and serves as an ontology repository. It consists of three basic components: ontology schema, static context data and dynamic context data. Ontology schema in HCoM represents concept hierarchies, concept relations, axioms and metadata information that represent knowledge in both generic and application domain. Static context is a context data that is relatively permanent and used as a base of the reasoning process. Dynamic context is the context that changes frequently and it is used as a source for both reasoning and action-triggering in a context-aware service.

RCDB

RCDB is created from the static context data of the EHRAM conceptual model using the mapping steps described in the relational context modeling section. This data is updated with new static context data that is captured during the run time. RCDB can be stored using any standard database management system. In our case, we use MySQL as a backend to store our context database and its schema.

Context Manager and Context Event

Context manager aggregates and sends the necessary reasoning resources from the HCoM model to the RAID-Action engine in CoCA in a push fashion each time a new context is acquired. This is accomplished based on the trigger information in the *context event* notice that is created by the context filter module when a new and valid context is captured.

Collaboration Manager

Based on decisions from the context selector, if relevant data in the concerned device is not sufficient, collaboration manager initiates neighborhood based context exchange between nearby peer devices. This module uses the principle of virtual network overlay that uses the [JXTA07] peer-to-peer protocol. Detailed specification about this module is given under section 5.6.

Interfaces

Stand for interfaces in HCoM: interface to capturing tools and interface to CoCA services. They are means through which resources are communicated to and from the model.

4.6.4 HCoM and the selection of appropriate context entities

A *reasoning space* is a *search space* from which the right set of information is extracted to perform reasoning and inferences. A *search space* on the other hand is a set of all possible solutions to a problem [Luger05]. Uninformed search algorithms use the intuitive method of searching through the search space, whereas informed search algorithms use heuristic functions to apply knowledge about the structure of the search space to try to reduce the amount of time spent on searching.

Many standard searching algorithms exist to look for the right solution in a search space [Burke05]. Among these are: list search algorithms, tree search algorithms and graph search algorithms. Examples of list search algorithms include: linear search, binary search and hash table search. Examples of tree search algorithms include breadth-first search, depth-first search, iterative-deepening search, depth-limited search, bidirectional search and uniform-cost search. Graph search algorithms can be seen as extensions to tree search algorithms and they exist in the form of graph traversal algorithms such as Dijkstra's algorithm, Kruskal's algorithm, the nearest neighbour algorithm, and Prim's algorithm.

Informed search uses a heuristic that is specific to the problem as a guide. To account for this, many applied fields of artificial intelligence like game playing (chess game tree, for example) use search algorithms like search tree pruning, mini-max algorithm and alpha-beta pruning.

As indicated in [Silva94] and SCOPES [Ouksel03], search space pruning facilitates an incremental construction of context knowledge for semantic reasoning. In HCoM, we use this principle of pruning to limit the amount of context data in the reasoning space so as to improve the performance of the context-aware service in CoCA.

The entire set of context data in the HCoM model is virtually organized into the EHRAM graph. The EHRAM graph consists of the hierarchical tree of context entities and their corresponding relations, axioms and metadata. We therefore use pruning techniques on the hierarchical tree of context entities in the EHRAM graph to minimize the size of the CoCA reasoning space.

4.6.4.1 Heuristics selection by example

The heuristics for pruning in HCoM come from three basic sources: explicit information given by the user during system initialization, entities sensed from the environment during the initialization and history data. History data are used to draw new sets of relevant entities based on entities from the other two sources.

The learning process of the prediction module is done offline by capable devices (devices that have sufficient processing and storage capacities and may exist in the form of PCs, Laptops, etc.) using prediction techniques like decision-tree or other datamining tools. The input in to the prediction module is the history data collected over time on entities participating in the process. The output from the module is a prediction model that, from an existing real-time set of entities and actions, predicts the next action (with probability/accuracy value specified). Such prediction model, in a PMML format for example, is made available for loading from the capable devices. The selector module uses the prediction model to assign a numerical score to each entity and class of entities. This score is used to determine whether an entity or an entity class is relevant or not. The probability threshold for selection can be set based on users' priority (selection time or accuracy of selection or both).

Let us use Mr. Bob's situation below to demonstrate this principle (example taken from the PiCASO scenario chapter 6). If an instance of an HCoM based service is initialized by a user *Bob* on his hand held *PDA* at *nine o'clock* on *Monday morning* while walking in a *corridor* of a building where *Classrooms* and a *Library* are located. Assume that, in a context data repository we have the following two categories of information: (1)

Information about *Classrooms* and *courses* that are scheduled in them, *professors* lecturing the courses, *students* attending the courses, etc. (2) Information about different rooms in the *Library* and other library related *services*, *devices*, *service personnel*, etc.

We will now demonstrate how the selector/pruning module works to decide if information about the *Classrooms*, the *Library*, both or none is to be loaded into the reasoner space of Mr. Bob's world. This means, given the situation in the example, what would *Bob* do next? Is he going to the *Library*? Is he going to a *Lecture_hall*? Is he going to both places one after the other? Or is he going somewhere else? The prediction module uses the history data about *Bob* and his usual habit on *Monday mornings* after passing by the *corridor*. Steps for demonstrating such predictions are given in Figure 4.12.

```
//P is conditional probability function.

//T is Threshold probability for selection that is obtained from
    the default threshold of the application, from the default
    threshold of the user or from the interface as a value
    entered by the user during HCoM initialization.

Calculate P1 = P([Bob, Classrooms, 9:00AM, Monday] | [PDA_01,
               Bob, Corridor_01, 9:00AM, Monday])

Calculate P2 = P([Bob, Library, 9:00AM, Monday] | [PDA_01, Bob,
               Corridor_01, 9:00AM, Monday])

If P1 < T prune Classrooms from the hierarchy otherwise collect
        Classrooms and its sub-entities as relevant.

If P2 < T prune Library from the hierarchy otherwise collect
        library and its sub-entities as relevant.
```

Figure 4-12:Steps in prediction

Using the above steps and assuming that most of the *Monday mornings* (80% of the time => P_1=0.80), *Bob* activates the service on his *PDA* while walking in a *corridor* before he goes to the *Classrooms*. On the other hand it is very rare that Bob goes to the Library on Monday mornings after passing by this corridor (rare means like 15% => P_2=0.15). Assuming also his default probability threshold value as 0.75, the prediction module, then, decides to load the *Classrooms* node and prunes the *Library* node from the hierarchy. A portion of the pruning process is shown in Figure 4.13. The figure shows the context entity tree in the context entity graph that has *person*, *activity*, *location*, *class*, *student*, *classroom*,

and *library* as context entity classes. It also shows entity instances like *Bob*, *AmphiRoom1*, etc. and some relations like *attendClass*, *reservedFor*, etc. Values of P indicated in the graph are used to determine the pruning point indicated in the graph.

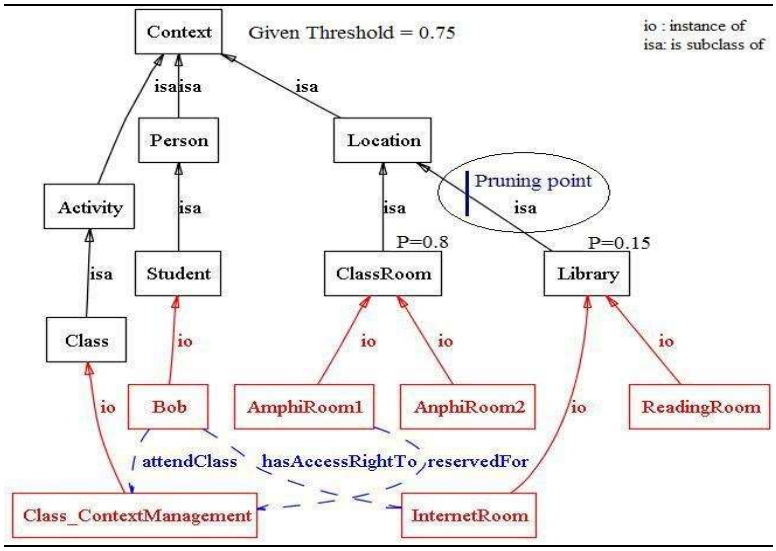


Figure 4-13: A pruning graph showing a portion of the context data space

A probability threshold value of 1 reduces the number of relevant entities. This guarantees the fastest response time but less accurate decisions. On the other hand, a probability threshold value of 0 loads all context entities as relevant. This guarantees the maximum possible accuracy but the worst response time. GCoM is an example of the case when threshold value is 0. This means applying selection/pruning on GCoM has no significance.

4.6.4.2 Selection/pruning algorithm

We will now develop the concept of selection/pruning from the context data space into a more generalized solution represented as a formalized algorithm. Figure 4.14 shows the algorithm for pruning non relevant or selecting relevant context entities (leaf nodes) and entity classes (intermediate nodes) in the hierarchy of context entities. In this algorithm, predictions and estimations are calculated based on the history information using prediction tools like decision-tree. The principle is similar to what we have shown in Mr. Bob's example above.

Lines 10-12 in the algorithm collect entities that are identified at the initialization stage of HCoM. In Mr Bob's example, *Bob* himself, his *PDA* and the *Corridor* in which he is located are what we call identified entities. Lines 13-18 collect user initiated data from the

user interface. These are optional data where users can enter or select list of entities of their interest depending on their intension. Lines 17 and 20 are about the threshold value that can either be set by the user through the user interface or estimated based on the initially identified or specified entities.

```
    //Algorithm for pruning non relevant entities (selecting relevant entities)
1.  Input:  T (set of all entities in the EHRAM hierarchy graph)
2.          E1 (set of entities identified at the time of initialization)
3.          E2 (set of entities specified at the time of initialization)
4.          getProbablity() (a function to predict the probability of occurrence based on history data)
5.          getTH(), estimateTH() (functions that set threshold value)
6.  Output: S (set of relevant entities selected from the set T) => (S ⊆ T)
7.
7.  Let τ ← 0 (threshold probability for selection of entity, value between 0 and 1 inclusive)
8.  Let S1, S2, S3, S4, S5, S', SS ← ∅
9.  Let δ ← DefaultDepth // Depth of search space for searching related entities  (δ>=1)
10. For every εi ∈ E1 do
11.         S1 ← S1 ∪ {εi}
12. End do
13. If (user has preferences)
14.         For every εi ∈ E2 do
15.                 S2 ← S2 ∪ {εi}
16.         End do
17.         τ=getUserTH() // user specified threshold cf. Fig. 4.12
18. End if
19.  S3 ← S1 ∪ S2
20. If τ=0 then τ = getApplicationTH() // default user threshold cf. Fig. 4.12
21. If τ=0 then τ = getDefaultTH() // default application threshold cf. Fig. 4.12
22. For every εi ∈ (T \S3) do
23.         ρ←getProbability(εi, S3) // conditional probability Fig. 4.12
24.         If (ρ ≥ τ)
25.                 S4 ← S4 ∪  {εi}
26.         End if
27. End Do
28. SS ← S3 ∪ S4
29. S←SS
30. Depth←1
31. Repeat
32.         For every λi ∈ SS do // SPARQL/SQL
33.                 S' ← S' ∪ {θi | hasRelation (λi, θi)}
34.                 S' ← S' ∪ {θj | hasRelation (θj, λi)}
35.         End do
36.         S ← S ∪ S'
37.         SS ← S'
38.         S' ← ∅'
39.         Depth++
40. Until Depth>δ
    Return S
```

Figure 4-14: Context selection/pruning algorithm

In Mr. Bob's demonstration example, the value 0.75 that is compared against the prediction probability 0.8 to select *Classrooms* stands for a threshold value. Lines 21-25 check, for every entity not already selected as relevant, whether it is relevant or not based on its prediction probability and the threshold value. In Mr Bob's example, *Classrooms* is an example of the selected entity class whereas *Library* is an example of a pruned node. Lines 27-39 collect all entities that have relation with entities that are already identified as relevant. The depth up to which the search for entities that have relations with entities that themselves are selected as related entities is determined by the repeat loop that starts at line 30. This loop depends on default-depth value that needs to be specified depending on the characteristics of the application domain. In our example, entities like *professor*, *course* and *student* have direct relation with *Classrooms*. Offices of the professors, designers of the courses, etc., on the other hand, are related to Classrooms only indirectly. All collected entity instances and entity classes are put in a set *S* as an output parameter for further processing and reasoning on the context data in the reasoning space.

4.6.4.3 Performance issues in the selection process

Context entities are parameters from which the contents of the reasoning space (context data) are defined. Names like *Device, Student, Bob, Library, PDA_Bob and Room_306* are context entities from which context data like *(Bob owns PDA_Bob*) or *(Student locatedIn Library)* are defined. With our algorithm, given the *whole* set of context data T_c= {c_1, c_2, ct}, pruning inappropriate context data from T_c, we get the set of *selected* context data S_c={c_1, c_2, c_s}. The two measures of performance of our algorithm are the accuracy (quality) of reasoning and the response time (speed). Given the set of ideally *appropriate* context data as A_c= (c_1, c_2, c_a), then theese performance values depend on the difference between the sets A_c and S_c.

$$\textit{Given} \quad \begin{cases} P \equiv A_c \setminus S_c \\ Q \equiv S_c \setminus A_c \end{cases} \quad \textit{then} \quad \begin{cases} \left.\begin{matrix} P \equiv \varnothing \\ Q \equiv \varnothing \end{matrix}\right\} \Longrightarrow \textit{Both speed and quality optimal} \\ \left.\begin{matrix} P \equiv \varnothing \\ Q \neq \varnothing \end{matrix}\right\} \Longrightarrow \textit{Quality optimal and speed low} \\ \left.\begin{matrix} P \neq \varnothing \\ Q \equiv \varnothing \end{matrix}\right\} \Longrightarrow \textit{Quality low and speed optimal} \\ \left.\begin{matrix} P \neq \varnothing \\ Q \neq \varnothing \end{matrix}\right\} \Longrightarrow \textit{Both quality and speed low} \end{cases}$$

The cardinalities of the sets P and Q are indicators for the quality and speed performances of our hybrid reasoning process. The smaller the value of |P| is the better the quality performance of the reasoning process and vice versa. On the other hand the smaller

the value of |Q| is the better the speed performance of the reasoning process and vice versa. Assuming that a given reasoning process always requires a proper subset of the total context data, i.e. Ac ⊂ Tc is always true, then the worst speed loss is when Tc≡ Sc, but this, on the other hand, guarantees optimal quality because all available information is used in the reasoning process. An example of a model with optimal quality in terms of loading complete data into the reasoner but the worst performance speed is our GCoM model.

$$T_c \equiv S_c \Rightarrow A_c \subseteq S_c \Rightarrow P \equiv \varnothing \quad (quality \;\; is \; optimal \;\;)$$

In order to improve both quality and speed, the selection (pruning) algorithm must be selected in such a way that the values of both |P| and |Q| are nearing zero. In real terms, it means that the performed selection/pruning matches with the users' intension. This on the other hand depends on the prediction module and how reliable the history data - or *the experience* - is. This is just like in human being where, under normal condition, experiences improve performances both in terms of quality and speed.

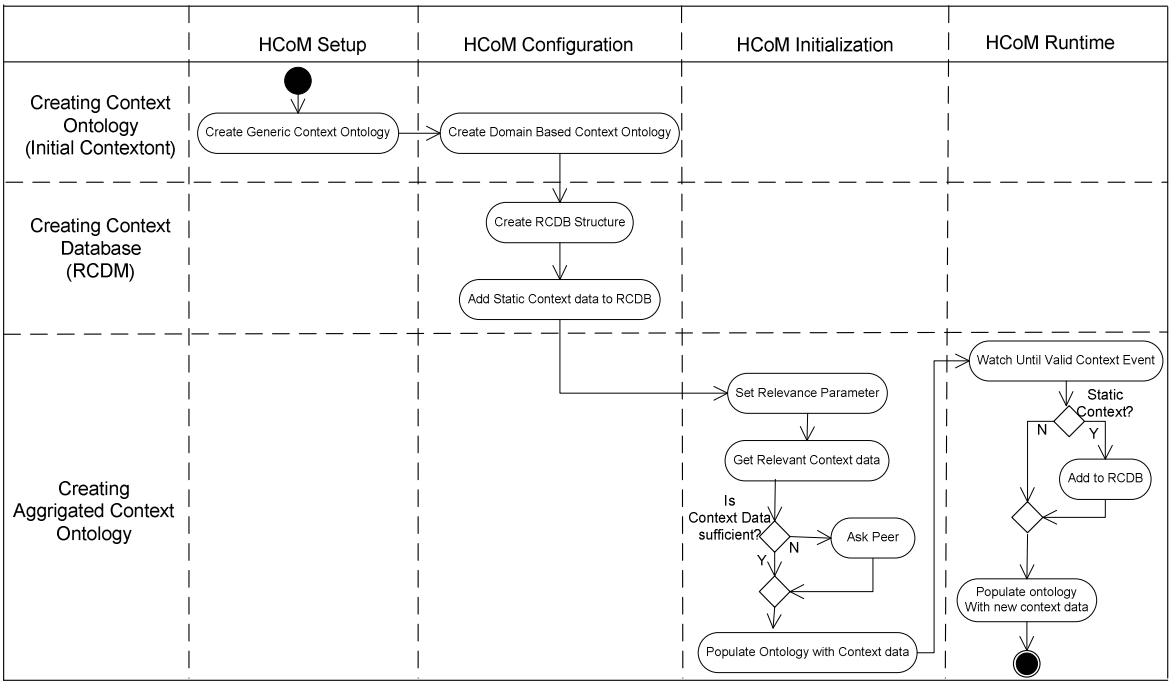


Figure 4-15: HCoM process flow

A practical example of this concept of how pruning can reduce the size of context data in the reasoning space and improve response time can be described as follows (using demonstration data from the PiCASO scenario chapter 6). In a campus scenario, for example, context data loaded for a particular student may contain information about *students*, *devices*, *courses*, *professors*, *libraries* and *class rooms*. Using our pruning

algorithm we can perform removal of less relevant data from the entire reasoning space. For example, *courses* that the student is not taking and *professors* of those courses can be pruned from the reasoning space. Similarly if the immediate intension of the student is to attend a class, all branches in the hierarchy that concern the *Library* can be pruned. This leads to an improved performance demonstrated in the evaluation section of chapter 6.

Figure 4.15 shows the HCoM system process flow diagram running from the initial setup up to the final execution and data delivery. Column wise partitions show the four states: setup, configuration, initialization and runtime. The box "*Get Relevant Context Instances*" in the HCoM initialization partition is where our *pruning algorithm* is executed. In this same partition, the query *"Is context data sufficient?"* is executed to check if there is some incomplete or inconsistent information about entities and relations. If the answer is no, it may need peer collaboration to get more information before proceeding to decision.

For example, if a device is located in the vicinity but its owner is not identified, this calls the *"Ask Peer"* module. Row wise partitions on the other hand show category of activities; creating ontology, creating context database and creating aggregated context model for delivery to the core context-aware service for reasoning and decisions. The aggregation partition is where our *pruning algorithm* is executed in order to load only relevant data for delivery to the core service.

4.7 Summary

In this chapter, we have presented our semantically rich novel approach for context management modeling. It uses the hybrid of ontology and database principles for modeling the management of both context data and context semantics. The important aspect of this approach is that in addition to separate processing of context data and context knowledge, selective loading of context data into the reasoner space ensures scalability.

Table 4-4: HCoM/EHRAM and appropriateness of modeling approaches

Requirements	Approaches					
	Markup Scheme	Graphical models	OO models	Logic based	Ontology (GCoM..)	Hybrid (HCoM)
Distributed composition	+	-	++	++	++	++
Partial validation	++	-	+	-	++	++
Quality of information	-	+	+	-	+	+

Incompleteness/ambiguity	-	-	+	-	+	+
Level of formality	+	+	+	++	++	++
Applicability	++	+	+	-	+	++
(Key: ++ Comprehensive + Partial - Limited or none)						

EHRAM is a conceptual context representation meta-model and HCoM is a hybrid model that uses components of EHRAM in ontology and relational schema. The ontology part represents the semantics aspect of the context data and the relational schema represents the context data itself.

HCoM model is hybrid and this means that we have all the grounds to claim HCoM model inherits all the important features from ontology, graphical, markup and relational modeling approaches. Hence, HCoM model responds best to the requirements [Strang04] of the context modeling approaches presented in section 2.2, distributed composition, partial validation, richness and quality of information, incompleteness and ambiguity, level of formality, and applicability to existing environments. This is presented in Table 4.4.

Chapter 5 COLLABORATIVE CONTEXT-AWARE SERVICES: THE CoCA PLATFORM

5.1 Overview on context awareness

In order to advance the operations of its functionalities, a context-aware system must be able to mimic human ability to recognize and exploit implicit information in the environment. Although identifying and deducing a human activity is a challenge, it is critical that context-aware applications should operate by conveying the appropriate information to the right place at the right time by inferring the user's intention. To accomplish this objective, a context-aware system must gather information from the environment or the user's situation, translate this information into the appropriate format, and combine context information to generate a higher context, take action based on the context information and make the information accessible to other applications and the neighborhood. The management model should handle context in a reusable manner to permit context from one source to be exploited by many distinct applications and devices in the neighborhood space that perform a variety of tasks.

This chapter presents our neighborhood based collaborative context-aware service platform (CoCA) that uses HCoM/EHRAM model as its basic data source. The role of collaborative computing in the CoCA neighborhood space is to share computing resources like context, rules, ontology, processor, memory, etc. to solve computing problems to provide comprehensive context-aware service, which would otherwise be difficult and sometimes impossible for a single pervasive device to solve. Among the basic requirements for collaborative computing between CoCA peers in the neighborhood space is the ability to self-organize into peer groups, discover each other and each other's services and resources. Sensors and IP-cameras, for example, track absolute and relative positions of mobile devices and humans involved and advertise this information for the neighbors (or peers) to use. Under some setting, mobile devices themselves may need to detect their contexts.

5.2 Acquisition of context data: Example on indoor positioning

Some context information can be provided to the context-aware system explicitly, such as a user's name or age; other context information can be obtained using sensors. Many types of sensor are already commonly in existence and can provide primitive physical information such as light, heat and pressure readings. Other types of context such as facial recognition depend on simple sensors such as cameras, but require considerable processing such as image recognition in order to make use of the information obtained. Location and identity are the most frequently sensed pieces of context. Active Badges [Want92] by Olivetti and AT&T emit infrared signals, which give a rough location and ID. Optical systems for context determination are also possible and research is underway in the areas of optical tracking and motion detection, stereo and 3D reconstruction and object recognition. Location is an important element of context information. Many different approaches have been taken to determining the location of agents within a context-aware system: GPS, infrared and radio signals have all been explored. In our work, we have investigated how a measure of signal strength from a general purpose WiFi (IEEE 802.11) [WiFi07] access point is used to detect locations using small computing devices. A summary of this work is shown as follows.

As part of the project PerSE (Pervasive Service Environment) [Gripay06] at LIRIS laboratory, INSA de Lyon, we have developed WiFi based system for capturing indoor locations. PerSE is a middleware that supports the interaction of independent and collaborating services to perform an intended action. As indicated in our work [Scuturici06], our indoor location tracker detects a room in a building where the holder of a mobile device is located. Like in any prediction process, modeling localization involves two basic activities: learning and prediction. During the learning phase, data about the real situation are collected, classified, and interpreted into knowledge. The prediction phase then uses this knowledge for location prediction based on the real-time data values. We plan our work in such a way that during the learning phase, a person holding a PDA moves around the rooms and other spaces in the building to capture the signal strength. All recorded data are associated with literal location names like room numbers. We have developed a WiFi-Spotter program that tracks received signal strength and then keep the record on a file for further analysis.

Such recorded data are further calibrated for variations that may happen due to the type of the tracking devices or other environmental influences. We have then planned a further step of processing for data and pattern classification using a datamining tool and a decision tree model. The result from this process is our working model that can later be used for real-time location detection. The model is represented in the Predictive Model Mark-up Language (PMML) format. Our positioning module (WiFi-Spotter) uses this model to predict locations on real-time bases. Figure 5.1 shows the architecture of our learning and prediction process. It also indicates the link from this prediction process to a context acquisition service.

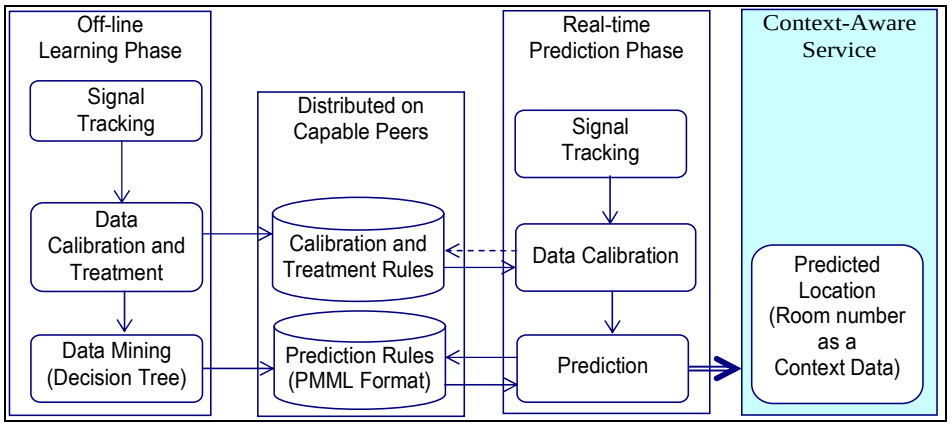


Figure 5-1: Architecture of our learning and prediction model

To provide positioning support, as shown on the architecture of the model, we have indicated the learning phase (off-line) and the prediction phase (real-time). During the learning phase, signal strength at selected locations of the rooms in the building including meeting halls, offices, common rooms, printing rooms and corridors are collected and classified for pattern identification and learning. The prediction or real-time phase uses patterns identified in the learning phase to detect location for the real-time values of signal strength.

5.2.1 **Learning phase**

The learning phase involves mobile clients that scan a list of radio frequency signal strength values from all n known access points at each tracking location. For each tracked point *k* we have a vector with the signal strength values and a label corresponding to the room/office where the point is situated:

$$\left(ap_k^1, ap_k^2, ..., ap_k^n, room_k\right)$$

The signal strength values are situated between -90 and -40. The number of APs that can be seen varies with the location. Corresponding to the data collected from the tracked points we build an attribute-value table. The attributes of the table corresponds to the access points (identified by the MAC address) and to the room/office label. We replace the missing values (corresponding to an AP not seen in a location) with the value -100. An example of a portion of the data is presented in Table 5.1.

Table 5-1: Sample measures of signal strength by room and access point

Room	_5A:40:0D:C6	_5A:40:0D:D7	_5A:10:0D:C6	_5A:10:0D:D7
501.317	-60	-60	-60	-57
501.317	-60	-60	-60	-57
501.317	-68	-63	-59	-65
501.319	-60	-62	-64	-100
501.319	-57	-57	-60	-100
501.319	-57	-66	-57	-100

We use a decision tree method for data classification and learning. Decision trees are popular knowledge representation, classification and learning tools as they are easy to use and interpret [9]. In decision trees, learned patterns are represented as a tree where nodes in the tree embody decisions based on the values of attributes and the leaves of the tree provide predictions. A new situation can be classified simply by tracing a path from the root of the tree to a leaf, with the path taken being determined by the input attribute values. These input values in our case are real-time array of signal strength values received from the APs.

5.2.2 **Prediction phase**

Among the number of algorithms and programs that implement decision tree, we used MCubiX [MCubiX07]. We found MCubiX to be well-structured tool whose output can be collected, viewed and stored in different forms. Demonstration of classification of a region into sub regions (into sub spaces for more than two dimensions) by the learning algorithm using a simple example of two APs and three rooms under a simple decision tree with depth two is given in Figure 5.2.

Prediction phase involves detection of the room in which a mobile client is located. The two important input parameters for the prediction module, therefore, are decision rules obtained from the learning phase and the real-time signal strength values collected at a specific location. The model is stored in Predictive Model Markup Language (PMML) format and contains the probability values of each prediction. PMML [PMML05] is an

XML-based language, which provides a way for applications to define data mining models and to share models between PMML compliant applications.

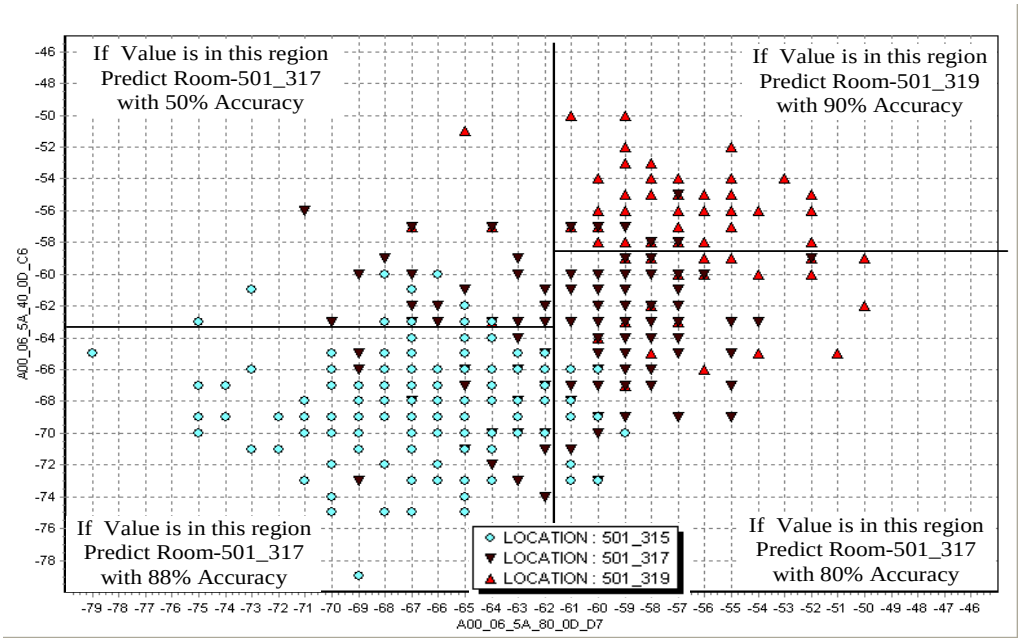


Figure 5-2: Effects of classification of a region into sub regions

Real-time signal strength values collected by the mobile client from the APs are given in the form of an array. For each of the m rooms involved, our prediction module calculates the probability that these values are observed in the room and then selects the room with the maximum probability. Given $P(R_k)$ as the probability that the given array of values is observed from room R_k, an expression for room R is given by:

$$R = R_k : P(R_k) = \max_{j=1}^{m} P(R_j)$$

5.2.3 **Experimental results**

We have used data from selected locations in 33 rooms of different size, from two floors of a three-storey building. We have collected some 13,500 records of arrays of signal strength values in all the 33 rooms, corresponding to four hours learning and calibration phase. The collected data contains information about 100 distinct access points. Only 20 of these access points are situated in the building. The layout of the two floors constituting our test-bed is depicted in Figure 5.3. As the rooms in our experimental test-bed are located side by side, knowledge about their width helps us to investigate the resolution of our detection mechanism. The average width of the small rooms is 3m for offices and 7m for

amphitheaters with an overall average of 5m interval between the corresponding points of references in the neighboring rooms. In most of the cases, it involves detecting small rooms with intervals as small as 3m. Using the collected data as a learning dataset, we have built a data-mining model using decision trees. The result has been tested using a cross validation technique. The results are very encouraging with the error rate is situated below 5%, corresponding to a 95% hit rate.

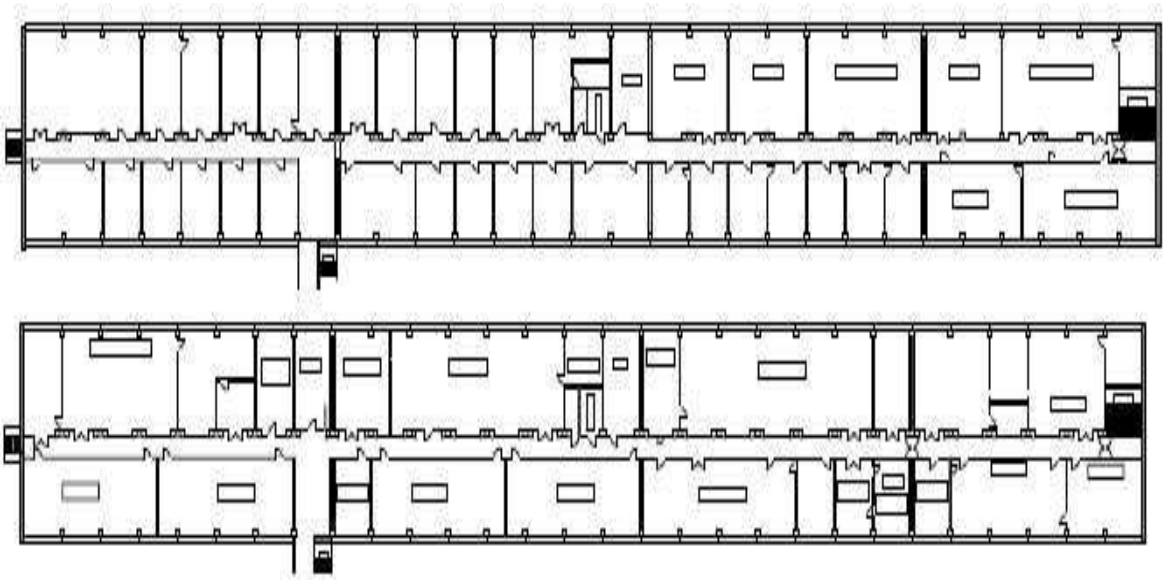


Figure 5-3: Layout of the floors used as a test-bed in our experiments.

5.3 The CoCA Service platform

A context-aware service platform in pervasive computing should aim at acquiring and utilizing context information to provide appropriate services without user supervision. For this purpose, we propose a neighborhood based Collaborative Context-Aware service platform (CoCA). CoCA is aimed to be domain independent middleware that enables application developers to use context information without the overheads of caring on how to manage it. The reasoning engine in the platform accepts a set of aggregated context data, rules and their semantics and changes it into concrete knowledge necessary for reasoning and decisions. Decisions in turn are used by applications to take appropriate actions.

CoCA platform is built from five layers: capturing layer, pre-processing layer, management modeling layer, context-aware core service, and application layer that executes the actions. Figure 5.4 shows layered CoCA architecture.

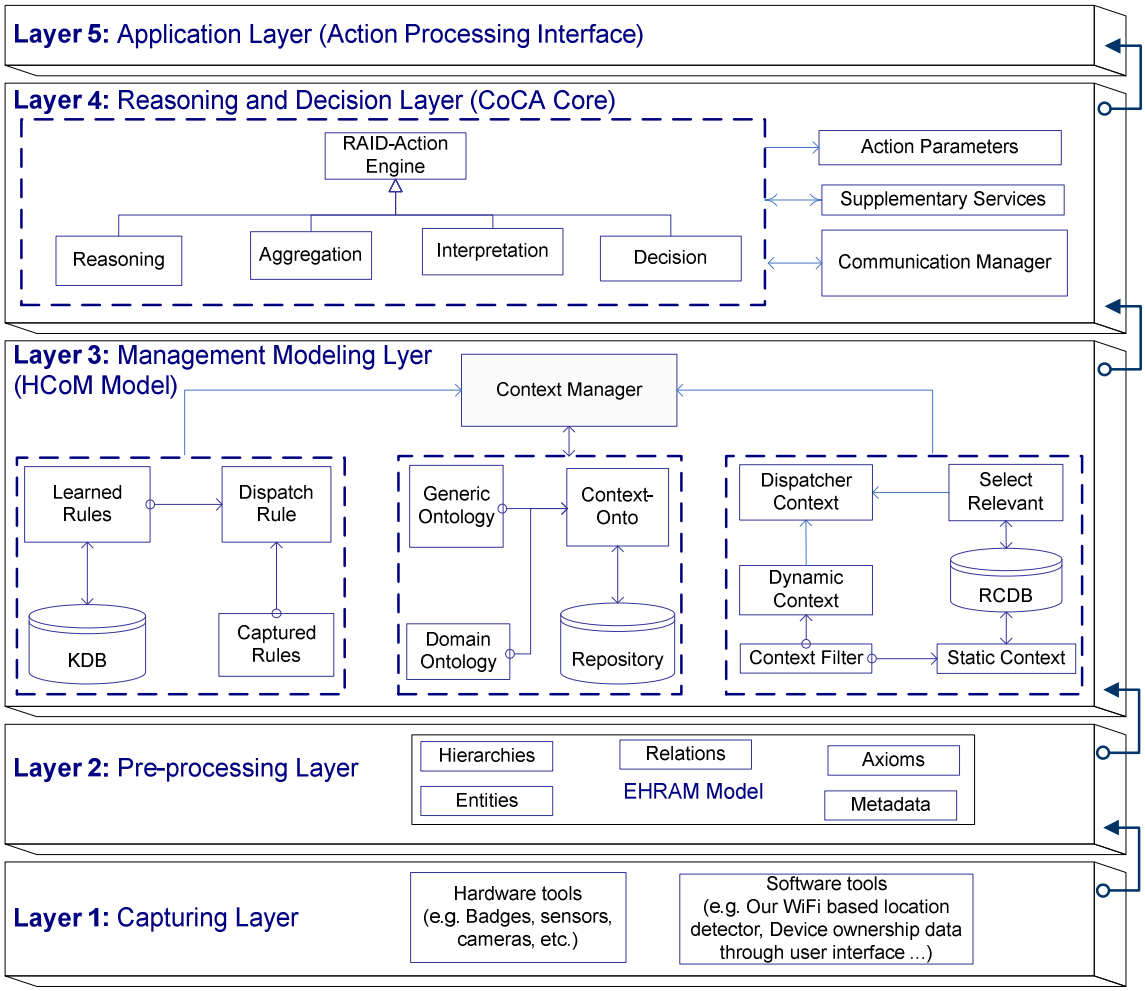


Figure 5-4: CoCA layered architecture

Layer 1, the capturing layer, deals with data acquisition tools in the form of either hardware or software. It involves data capturing hardware like wearable badges, sensors and cameras. Among the software tools are our indoor location tracker discussed in earlier section. It converts a general-purpose WiFi signal normally designed for networking into a meaningful location name. CoCA being a platform for applications in multiple domains, the interface is built based on APIs.

Layer 2, the pre-processing layer, is used to formalize and prepare the captured data for further processing. It deals with a conceptual modeling of the captured data according to context representation formalism. It performs separation of context related data into entities, hierarchies, relationships, axioms and metadata. This allows us to organize and process the context data and the context semantics separately. Details of activities involved in this layer are given in the chapter about our EHRAM context representation model.

Layer 3, the context management modeling layer, deals with how we organize context resources useful for reasoning. Components in this layer provide the necessary data to the core service. Formal representation of this data can be given using the HCoM model. The context filter filters and sends the static context data to RCDB and the dynamic context data to the context dispatcher. Context ontology consists of domain dependant and domain independent (generic) ontology. Context-onto and its repository are used to store and manage the ontology. Context dispatcher manages learned and captured rules. Major works involved in this layer are discussed in the chapter about the HCoM model.

Layer 4, the CoCA core layer, is where the final context-aware reasoning and decisions are performed. It provides the core context-awareness service after reasoning on the components. It consists of the *RAID-Action engine (Reasoning, Aggregation, Interpretation, Decision and Action engine)* that populates the ontology with the context data and then applies rules and axioms for reasoning and decision on the actions to be triggered. Coordinate based location values, for example, are interpreted to street names. It also performs aggregation by combining two or more low level contexts to one meaningful high-level context. Aggregation of body temperature, heart rate and blood pressure of a patient can be used to tell patient's health condition. The RAID Action Engine being at the heart of the service platform uses combined contexts, rules and ontology as an input.

The supplementary services in this layer consist of some CoCA elements outside the CoCA core service. This includes services like the knowledge discovery (rule-mining) service that adds features to enhance learning capacity of the CoCA platform, privacy and security management service, etc. Collaboration-Manager works based on peer-to-peer negotiation and communication protocols to get context and reasoning support. If the resources in the current device are not sufficient for the operations, the collaboration manager is responsible to contact the neighborhood space for the necessary support. The neighborhood space consists of computing devices with varying capacities. Each device in the neighborhood space is assumed to have a minimum configuration of the CoCA service platform. When requested, each peer in the CoCA neighborhood deals with the query and sends its response back to the source. Details about this layer will be discussed in this chapter. Security manager looks up security and privacy policy of the identified context entity before exchanging any resource with the entity.

Layer 5, the application layer, is the application domain dependent layer where actions are triggered reactively or proactively. It also hosts action-triggering process depending on the specific application domain in which the platform is used.

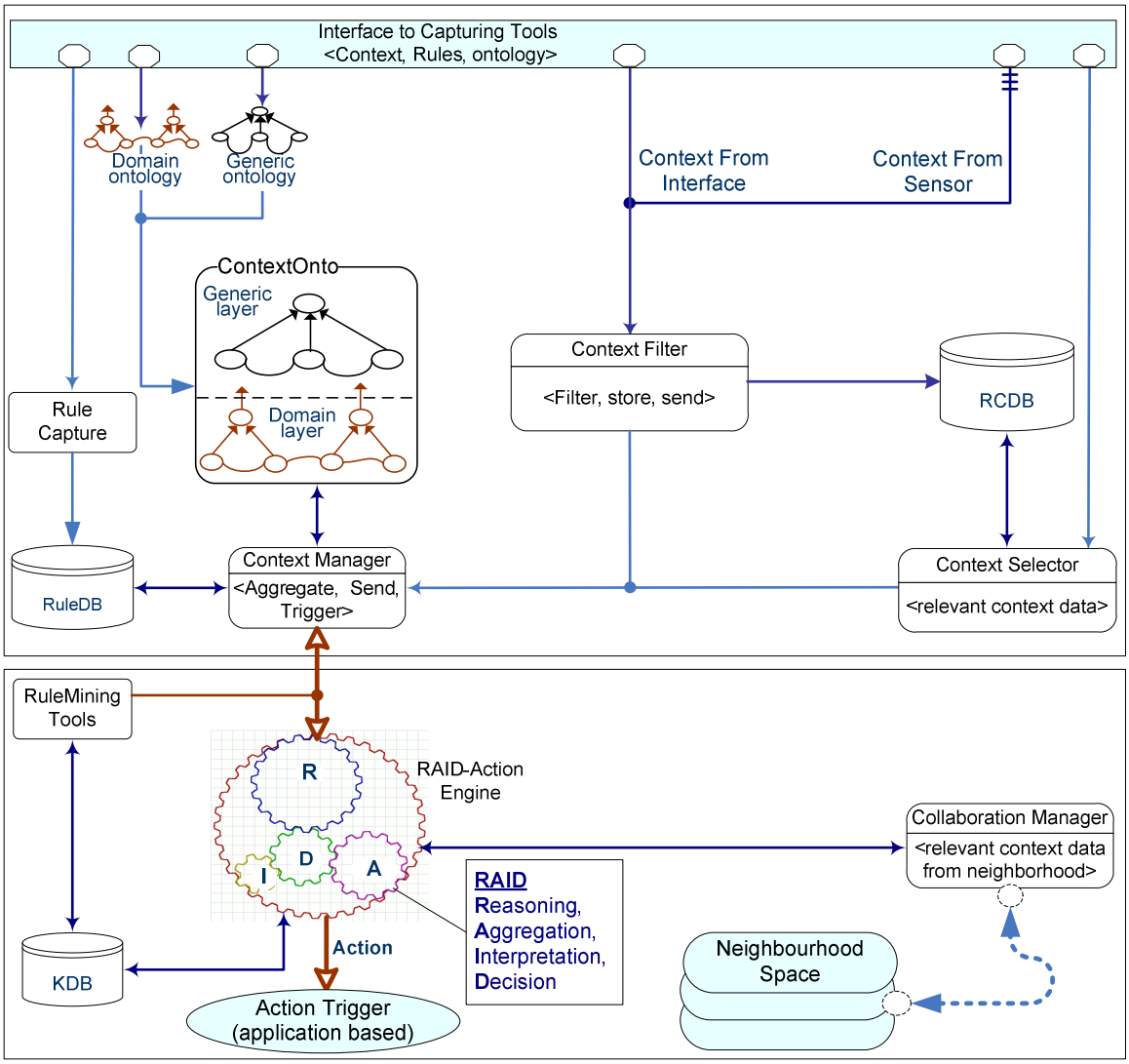


Figure 5-5: Component view of the CoCA platform

Detailed component view of CoCA platform is given in Figure 5.5. The upper box in the figure indicates HCoM and its components as a data source in the CoCA platform. The lower box indicates the core components of the CoCA platform. A flow diagram showing data exchange among CoCA components and importable APIs is given in Figure 5.6.

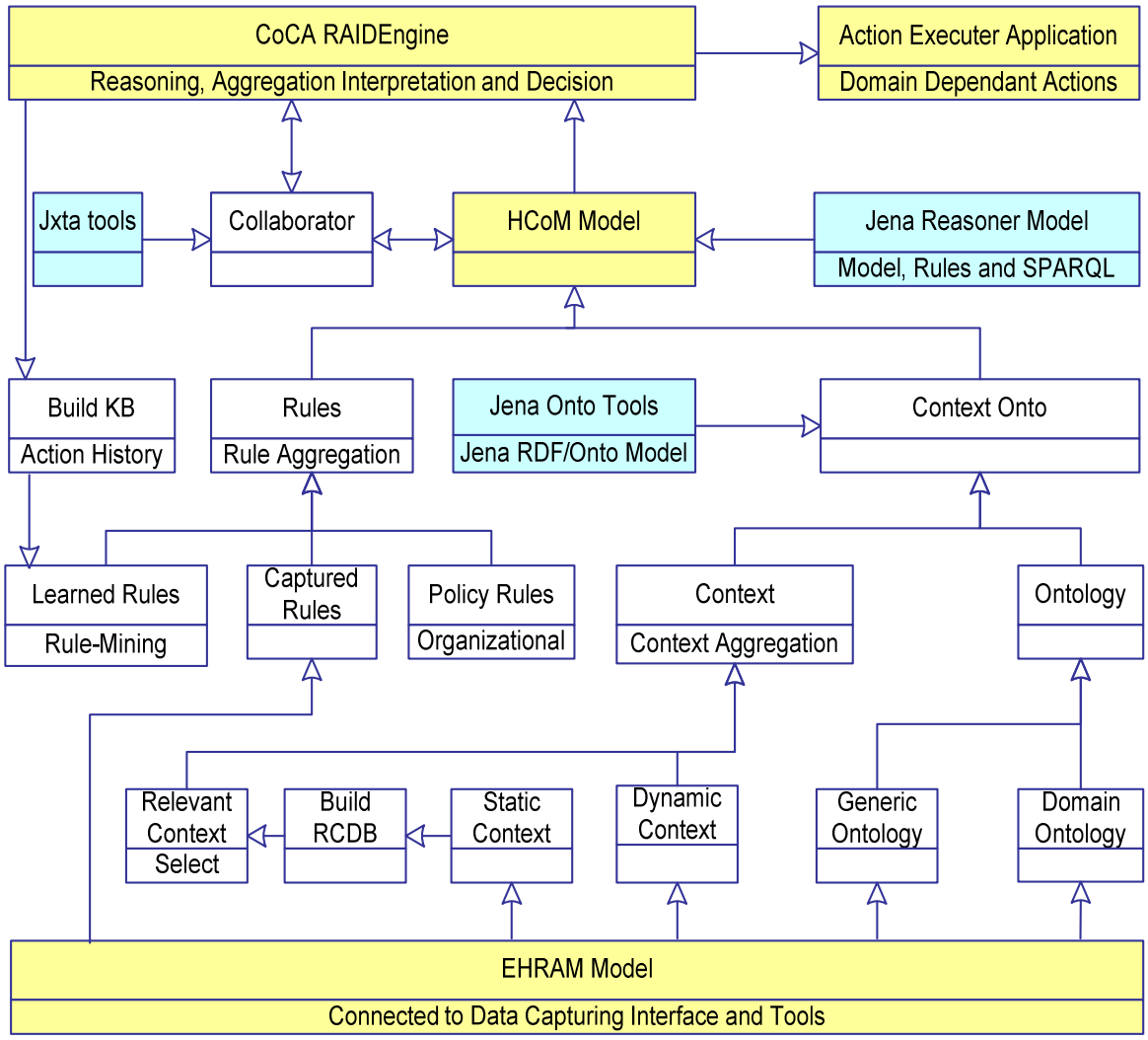


Figure 5-6: Internal data exchange among CoCA components

5.4 RAID-Action engine in CoCA

RAID-Action engine in CoCA stands for Reasoning, Aggregation, Interpretation, Decision and Action. RAID depends on the three basic types of data from the HCoM model (context, ontology, rules) that are aggregated using Jena reasoner and SPARQL query tool. Data *aggregation* and *interpretation* in the CoCA platform is done at multiple places with different abstraction levels. First aggregation and interpretation is done by the context-filter tool where irrelevant context are discarded. Second level aggregation and interpretation is done by the context manager and the third and last context aggregation and interpretation is done by the RAID-Action engine before reasoning and final *decision*s. Interpretation and aggregation may involve integrating numerous contexts into one to provide a higher-level context, thus the interpreter alters context information by raising its level of abstraction.

In our indoor location detector module discussed earlier in this chapter, it aggregates and converts array of radio signal-strength data collected from WiFi access points at a particular point in the building to a meaningful room name or room number.

One approach to context aggregation is to use context fusion [Chen04] that converts the lower level context into higher-level context usable by applications. Context fusion can be used to synthesize context from the same type of sources in order to increase the validity of the information so that erroneous sensors or reading are detected to avoid improper decisions by the system. Context fusion is also deemed to be the aggregation of context of varying types from a different variety of sources to produce a context that is exploitable by the system. This minimizes the need for the context-aware application to gather the required context from different sources that would otherwise be obligatory because of the distributed nature of the context-aware systems in pervasive environment. Aggregators support the delivery of particular context to an application, by accumulating related context that the application seeks into one logical placement. An aggregator facilitates interpretation of context hence it will aggregate diverse context information for different requesting applications.

Table 5-2: Sample ontology based and user defined rules

Rule	Category
(?a property ?b) (?b property ?c) → (?a property ?c) E.g. locatedIn, subClassOf, contains… (transitive property)	Ontology
(?a property1 ?b) → (?b property2 ?a) E.g. ownerOf and OwnedBy, locatedIn and contains … (inverse property)	
(?a property ?b) → (?b property ?a) coLocatedWith, friendOf …(symetric)	
(?device locatedIn ?location) (?device ownedBy ?person) → (?person locatedIn ?location)	User defined
(?student locatedIn Library) (?student owns ?phone) → (?phone “switchMode” “Silent”)	

Rules play an important role in the process of *reasoning* about contexts. Reasoning is performed based on two reasoning sources:

- Ontology reasoning that is based on rules that are integrated in the OWL semantics, e.g. using *transitive* and *inverse* relations.

- User-defined reasoning which are stated outside of OWL, e.g. if *person* is *locatedIn bedroom* and *electricLight* is *dim* conclude the *person* is *sleeping*.

Implicit rules are derived from the ontology and explicit rules are defined by the user based on the specific domain of application. Table 5.2 shows an abridged form of some of these rules grouped into two categories: ontology based and user defined rules.

5.4.1 **Action Trigger**

Actions are major outcomes from the CoCA services. Algorithms for action triggering in CoCA are based on multifaceted action processing approach [Rarau05]. Multifaceted action processing is based on *decision*s from CoCA-RAID and some other factors such as priority or existence of some other actions currently triggered. It is based on the idea that an application consists of both components that are context sensitive and components that do not depend on the context. Context sensitive component can be seen as an item with many facets that behaves like a switch. If the condition is true then the facet is exposed otherwise the facet is hidden.

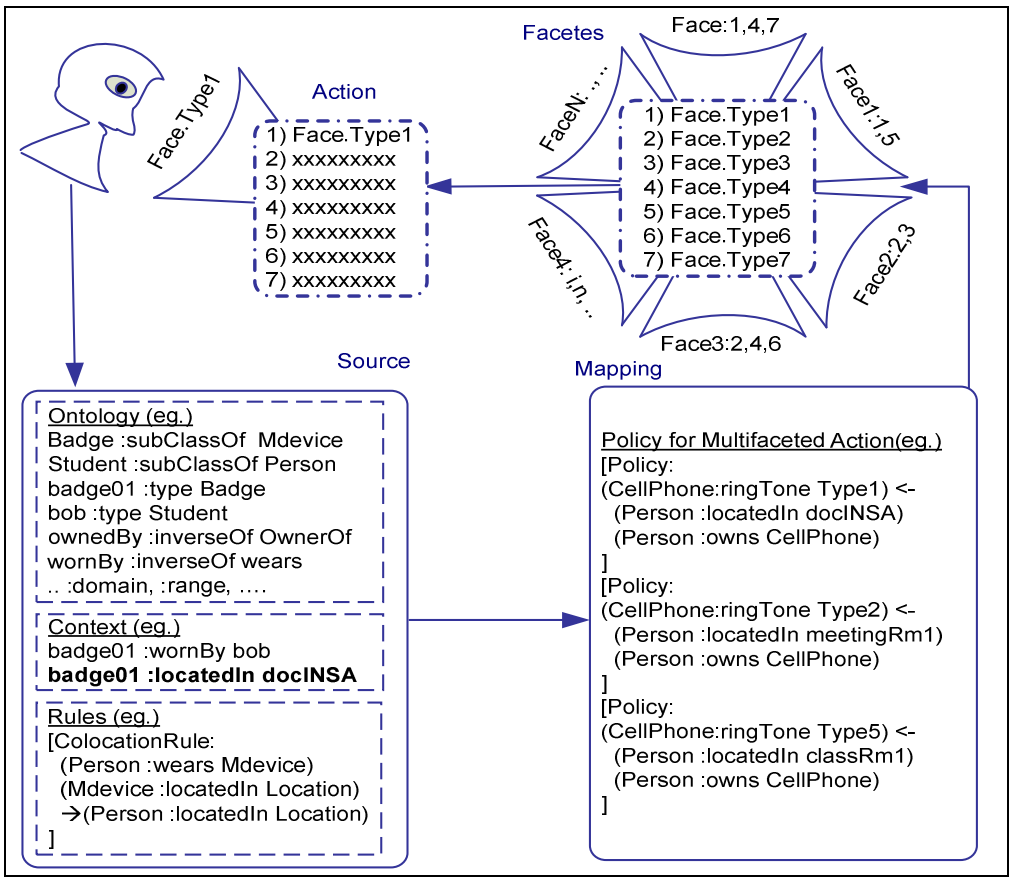


Figure 5-7: Principles of multifaceted action processing in CoCA

The principle of multifaceted action trigger is demonstrated in Figure 5.7. The figure shows four blocks of the process: the source block, the mapping block, the facet block, and the action block. Under the source block, we have three elements that demonstrate on ontology, context and rules involved. In ontology, we have the following set of semantics that describe meta-knowledge about the environment of the person in question.

```
1. Badge :subClassOf  Mdevice
2. Student :subClassOf Person
3. badge01 :type Badge
4. bob :type Student
5. ownedBy :inverseOf OwnerOf
6. wornBy :inverseOf wears
```

Lines 1 to 4 define that a *Badge* as a subclass of a class *Mdevice* (mobile device), *Student* as sub class of a class *Person*, *badge01* as an instance of the class *Badge*, *bob* as an instance of the class *student*. Lines 5 and 6 define the property *invrseOf* between *ownedBy* and *ownerOf* and between *wornBy* and wears.

This means that while in process reasoning, they obey the *inversOf* axiom. For instance, in the context definition of the source block, we have context data defined using *wornBy*, *i.e.* "*badge01: wornBy bob"*. However, in the definition we only have a rule defined on the relation *wears, i.e.* "*Person: wears Mdevice*". The reasoner uses the *inverseOf* axiom to interpreter and maps the two statements.

In the context definition, we have one static context data that holds stored information about the relation between badge01 and bob, i.e. "*badge01: wornBy bob"* and one timely context that is sensed or derived currently, i.e. "*badge01: locatedIn docINSA*". It is highlighted in the figure because this is where all the action-trigger process starts.

The rules indicate that if a badge is located somewhere then we conclude that the person wearing the badge is located there.

```
[Person :wears Mdevice)
  (Mdevice :locatedIn Location)
  →(Person :locatedIn Location)]
```

In the mapping block, we have multifaceted action mapping (switching) policy (rules). It shows when the type of the *cellphone* ringing tone is changed to *ringtone* type1 (e.g. vibrating), *ringtone* type 2 (e.g. decent ringtone), *ringtone* type5 (e.g. music ringtone), *ringtone* type7 (silent mode), etc.

```
[(CellPhone:ringTone Type1) <-
  (Person :locatedIn docINSA)
  (Person :owns CellPhone) ]
[(CellPhone:ringTone Type2) <-
  (Person :locatedIn meetingRm1)
  (Person :owns CellPhone) ]
[(CellPhone:ringTone Type5) <-
  (Person :locatedIn classRm1)
  (Person :owns CellPhone)]
```

The facet block is a demonstration of the multi facet (multiple faces) that represents action types (in our case different ringtone types). Some of the faces show the presence of more than one action which is natural like setting the ringtone of a telephone both to decent ringtone and vibrating mode at the same time.

The action block is about the visibility of the effect of the action to the user. In the figure, all except the face for Action1 are hidden (marked as *xxx*).

5.5 Proactivity in CoCA

Push and pull methods are the two options through which context-aware systems extract the necessary context information from context sources and perform relevant action. In the push method, context information is sent to the application in the push fashion (proactive). This means that context information is collected before it is needed, which may result in a better performance. The shortcoming of this approach is the consumption of resources for gathering and disseminating context that may never be exploited by the context service. The pull (reactive) approach on the other hand gathers only context information that is required by the service. Naturally, pervasive context-aware systems should be proactive or push type so as to satisfy the self-triggering property. Event based proactivity or push type service activation is used for triggering the context-aware service in CoCA. The context filter tool sends the context event notice to the context manager, which in turn activates the CoCA core, indicating occurrence of the new context. Then, the CoCA core reads the new context data, performs reasoning on the context, and suggests the action to be performed by the application. Applications are responsible to execute the actions.

5.6 Collaboration in CoCA

Storage, processing and reasoning of context data to knowledge is highly resource intensive while on the other hand most ubiquitous devices in the pervasive world have scarce resources. Mobility and anytime/anywhere access requirement of pervasive users

make the problem more challenging. To overcome this problem, we propose a collaborative approach where devices collaborate and combine their resources towards solving the problem. In this process, capable devices, like standard PCs, play an important role of “big” brothers to support tiny devices like PDAs and smart phones. With the current trend of PC availability, we can assume that, in the neighborhood space, we always have capable devices that play this role.

JXTA as a supporting technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs, servers and super computers to communicate and collaborate in a peer-to-peer manner. Figure 5.8 shows JXTA based CoCA collaboration architecture. It consists of the JXTA Core layer, the JXTA service Layer and the application layer where the CoCA collaboration service is placed.

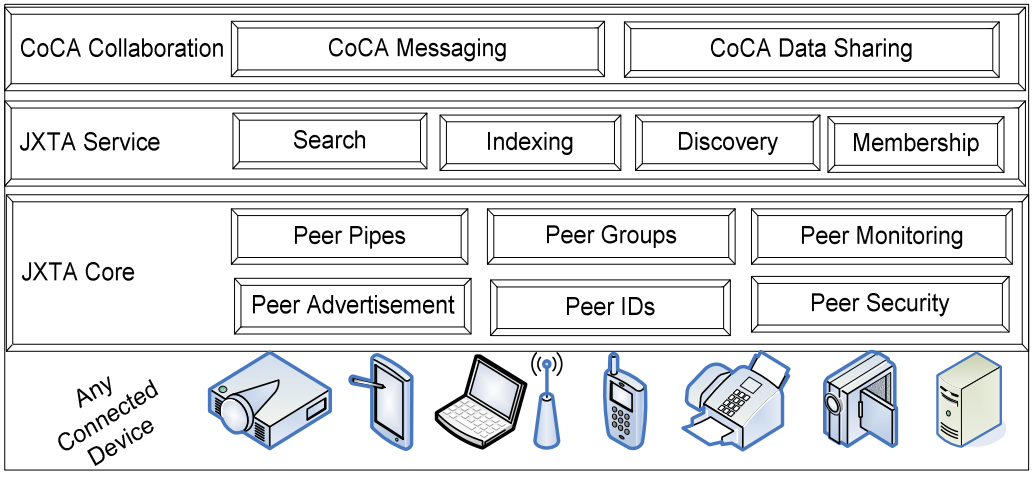


Figure 5-8: CoCA collaboration architecture

The JXTA Core layer is located at the bottom of the service hierarchy. In the JXTA Core, all the basic peer-to-peer functionality that peers can implement and use is provided. This layer involves notions of peer groups, pipes, peer monitoring, security as well as other functionality for discovering resources, creation of advertisement documents etc. It is important to note that since JXTA was designed in favor of peer-to-peer networks composed not only of PCs but also of smaller devices, the core's size has to remain as compact as possible. Indeed a peer does not need to run a complete implementation of the JXTA platform in order to participate in the network but only those protocols necessary for its smooth operation.

The JXTA Services layer is located above the core layer. This layer contains various services, which are implemented with calls to the JXTA Core. A JXTA service can be viewed as a library offering certain functionality that JXTA peer-to-peer applications may use if necessary. An example of a JXTA service is the JXTA Search system, which offers Information Retrieval functionality to a network of peers.

The CoCA Collaboration service takes the role of a JXTA Application layer. Peer-to-peer applications make use of only some core services. CoCA platform uses data sharing, messaging and possibly other services depending on the purpose of collaboration among CoCA peers. In such collaborative peer-to-peer applications, we need to support different levels of security and resource access. *Security* between peers in the CoCA service environment can be achieved by the JXTA peer security functionality in which peers operate in a role-based trust model, in which an individual peer acts under the authority granted to it by another trusted peer to perform a particular task. There is an ongoing PhD research work on such trust based security management for pervasive systems in our team [Saadi07]. We are hopping to incorporate these modules at the end.

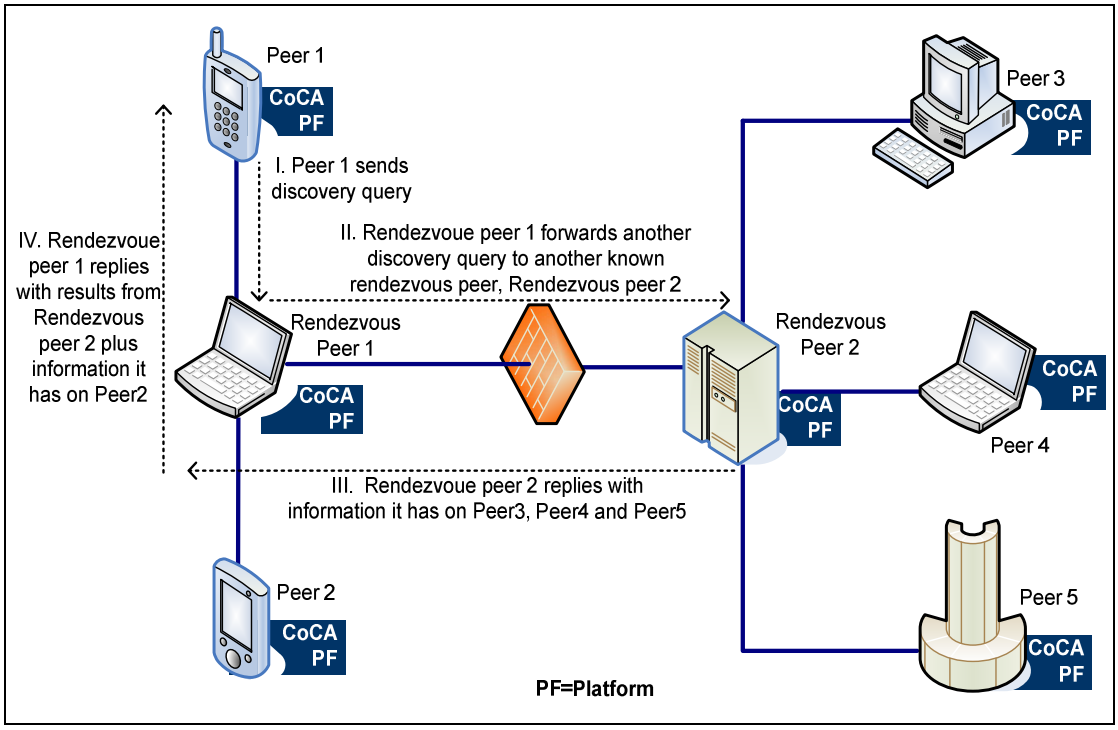


Figure 5-9: CoCA peer collaboration and discovery principles

The role of collaboration manager in the neighborhood space of the CoCA service platform is to share computing resources like context, rules and ontology. Among the basic requirements for collaborative computing between CoCA peers in the neighborhood space,

therefore, is the ability to self-organize into peer groups, discover each other and each other's services and resources. This principle is demonstrated in Figure 5.9. It shows how resource discovery and peer collaboration works using JXTA based peers and rendezvous peers. Each device in the collaboration space should have a minimum configuration of the CoCA platform installed on it in order to participate in the collaboration.

Peer discovery gets information like identity, security level, availability, willingness to participate in the group process, time to live (TTL), etc…of the peer. We suggest that all such information be made available to the rendezvous peer by all peers at the time it joins the group. For example, we can set a simple requirement that *"every peer must provide its TTL in order to take part in the peer group"*. The logical step that follows peer discovery is exchanging messages and resources by first discovering who owns what.

An algorithm for the processes involved in the CoCA collaboration manager is shown in Figure 5.10. The algorithm represented in the form of activity diagram indicates steps involved to discover the appropriate peer and establish a peer-to-peer connection between the collaboration modules in the two peers according to our JXTA based architecture discussed earlier. The diagram has five vertical activity partitions indicating the type and category of peers involved in the collaboration process. The first partition (InttialPeer) indicates activities involved in the peer that initiates the collaboration. The second partition (Nearby Rendezvous peer) indicates activities involved in the rendezvous peer of the group in which the initial peer is registered. The third partition involves activities involved in peers within the same group with the initial peer. The fourth partition (FarRendezVous peer) is for the activities involved in the rendezvous peer accessible by the near by rendezvous peer. Finally, partition five represents activities in peers that belong to other groups.

Figure 5.11 shows a use case of a JXTA based discovery protocol to handle messaging and queries during CoCA collaboration. The demonstration example has six peers and three rendezvous peers organized into three groups. For this demonstration, at a given particular time, the first group has three peers joined to participate in the collaboration, the second group two peers and the third group one peer. Each group has one elected rendezvous peer. The use case assumes that the required resource is found in Peer P22. The process is demonstrated in the steps described below:

(1) Peer P11 sends a query message *(messaging)* "Who knows about smart phone SP001?" to a rendezvous peer RP10. (P11→RP10).

(2) RP10 checks in its repository. On unsuccessful check, re-routes the query to the peers within the group. (RP10→P12, RP10→P13).

(3) P12 and P13 respond that they do not know SP001. (P12→RP10, P13→RP10).

(4) RP10 again re-routes the query to another rendezvous peer RP20. (RP10→RP20).

(5) RP20 checks in its repository and, on unsuccessful check, re-routes the query to the peers in its group. (RP20→P21, RP20→P22).

(6) P21 responds that it does not know SP001. (P21→ RP20).
P22 responds that it knows about SP001. (P22→P20).

(7) RP20 passes over the good news to RP10 with all the necessary information to get connected with P22 for more information. (RP20→RP10).

(8) RP10 passes over all the information to P11. (RP10→P11).

(9) P11 connects with P22 for *(data sharing)* the details about SP001. (P11→P22).

If for example P11 detects a smart phone SP001 in a meeting room and wants to decide the person holding this phone so as to use this information to conclude the presence of the person in the meeting, it may send the following SPARQL Query to P22: (CoCA:SP001 CoCA:ownedBY ?Person).

(10) P22 responds what it knows about SP001 to P11. (P22→P11).

The response by P22 to the query from P11 can be presented as (Person=Bob).

A sample of Jxta based code and data segment for context advertisement and discovery used in CoCA/HCoM is shown in Figure 5.12. Context advertisement data comes from different sources. Among these is our location prediction service detailed in [Scuturici06]. This code segment shows three parts: Context advertisement from lines 1 to 6, context discovery from lines 7 to 17 and sample format of advertised data lines 18 to 23.

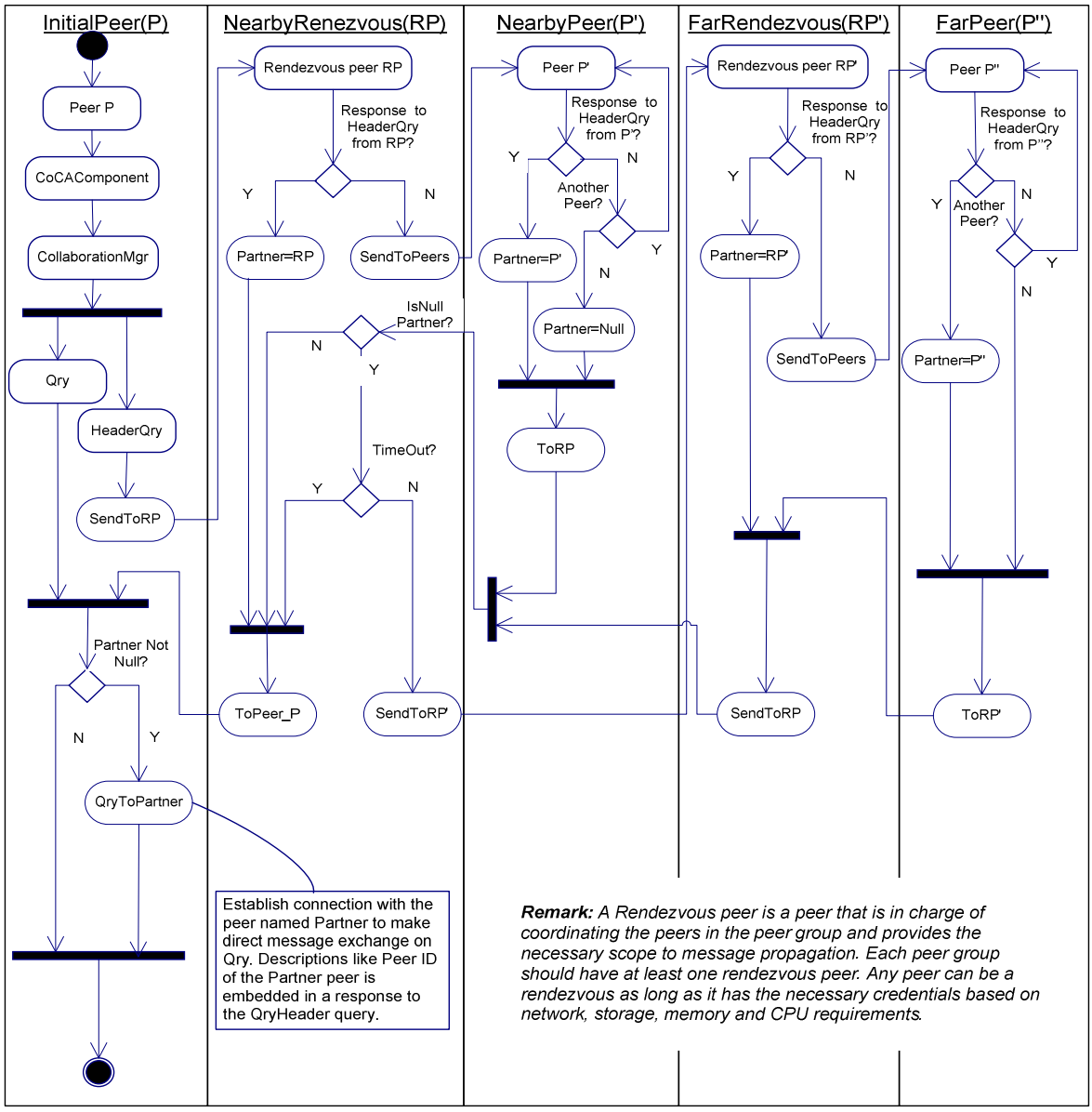


Figure 5-10: CoCA collaboration algorithm

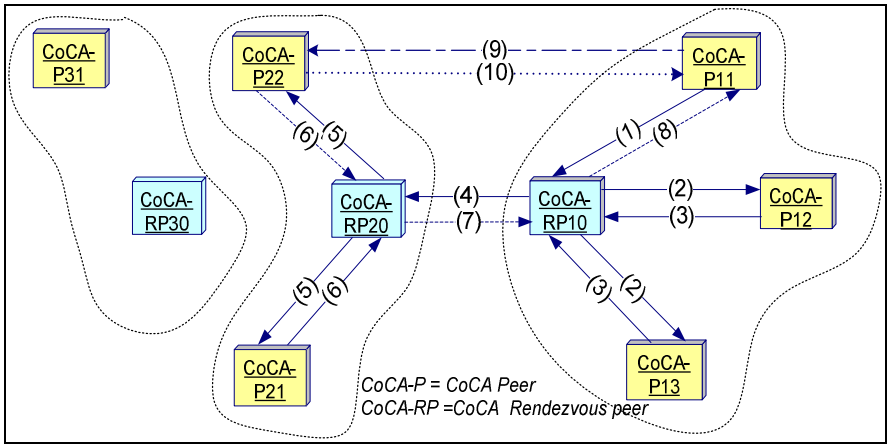


Figure 5-11: CoCA collaboration process

```
// Publishing and advertising context data
1  LocatioAdvertisement adv = new
   LocationAdvertisement();
2  adv.setID(myID);
3  adv.setName("PDA001");
4  adv.setOwner("Bob");
5  adv.setLocation("DocINSA");
6  discoveryService.publish(adv, DiscoveryService.ADV,
   lifeTime, expTime);
//Retrieve by discovering context advertisements
7  discoveryService.getRemoteAdvertisements(null,
8  DiscoveryService.ADV,"Name", "PDA001*");
9  Enumeration advs =
   discoveryService.getLocalAdvertisements(
10 DiscoveryService.ADV, "Name", "PDA001*");
11 LocationAdvertisement adv = null;
12 while (advs.hasMoreElements()) {
13 adv = (LocationAdvertisement) advs.nextElement();
14 String ownedBy = adv.getOwner();
15 String locatedIn = adv.getLocation();
16 break;
17 }
//Sample advertisement file that contains context data
18 <jxta:LocationAdvertisement
   xmlns:jxta="http://jxta.org">
19 <ID> urn:jxta:uuid 59616261646162614A7874615… … </ID>
20 <Name> PDA001 </Name>
21 <Owner> Bob </Owner>
22 <Location> DocINSA</Location>
23 </jxta: LocationAdvertisement>
```

Figure 5-12: Sample code and data for advertisement and discovery

5.7 Summary

CoCA is a collaborative middleware platform that is based on HCoM model. Evaluation of CoCA with respect to the context-aware system requirements presented in chapter 2 of is given as follows:

- *Support for heterogeneity:* CoCA supports heterogeneity. Its architecture is designed to work with devices of any sort and application of any domain. The semantic ontology used in the CoCA reasoning can also be extended to enhance interoperability so as to improve heterogeneity.
- *Support for mobility:* The core communication principle in the CoCA platform is based on Jxta protocol. Jxta protocols provide mobile peer-to-peer computing support [Maibaum02] and CoCA platform is, therefore, well suited for mobility.
- *Scalability:* The use of hybrid approach in the HCoM model ensures scalability of the CoCA platform by limiting the amount of context data in the reasoning space. This principle remains the same for any change in participating number of peers. After the partner for the data source is identified, a direct link is established between the initial

peer and the partner peer. As presented in chapter 6, experimental result on scalability of the CoCA platform using PiCASO confirms to this.

- *Support for privacy:* There are works remaining to be done on privacy issue in CoCA platform. For the time being, collaboration is based on voluntary and trust.
- *Traceability and control*: Every action in CoCA platform is stored in the event log for any control and future reference.

Table 5-3: Comparison of performance of CoCA platform with other relatede works

Requirement by	ContextToolkit	CFN	ConFab	Gaia	RCSM	CoCA
Heterogeneity	+	-	+	+	+	+
Mobility	+	++	-	+	-	++
Scalability	-	++	-	-	-	++
Privacy	-	-	++	-	-	-
Traceability	-	-	+	-	-	++
Tolerance	-	++	-	+	-	+
Deployment	-	+	-	+	+	-
Decision	-	-	-	-	-	++
(Key: ++ Comprehensive + Partial - Limited or none)						

- *Tolerance for component failures:* CoCA collaboration algorithms are dynamic in the sense that, if a partner with which the initial peer created a link fails for some reason, the other candidate partner returned during the initial partner search over takes the collaboration.
- *Ease of deployment and configuration:* The design of CoCA platform allows an easy deployment and configuration of the middleware to be used in any domain of application. From the implementation point of view, there is some work remaining to make CoCA deployable in the hand held computing devices.
- *Decision support:* Decision support in CoCA platform is provided in a proactive or reactive fashion. It provides a decision support and action trigger functionality to minimize user intervention.

Based on these requirements, Table 5.3 shows comparison of CoCA platform to other related works. This summary shows that CoCA is a promising middleware platform for the development of context-aware applications in pervasive environment.

Chapter 6 IMPLEMENTATION AND DISCUSSION

6.1 Implementation plan

In this section, we present implementation plan for the demonstration version of the CoCA platform and the HCOM model. Figure 6.1 shows flow of processes in the overall states of the CoCA platform development starting from the initial context acquisition state up to the final action trigger state. Column wise partitions (blocks) in the diagram show four major stages in the CoCA platform development process: context capturing (Interface), context representation modeling (EHRAM Model), context processing and management modeling (HCoM Model) and context-aware core service (CoCA Core). These partitions correspond with the first four layers (layers 1 to 4) of our layered CoCA architecture.

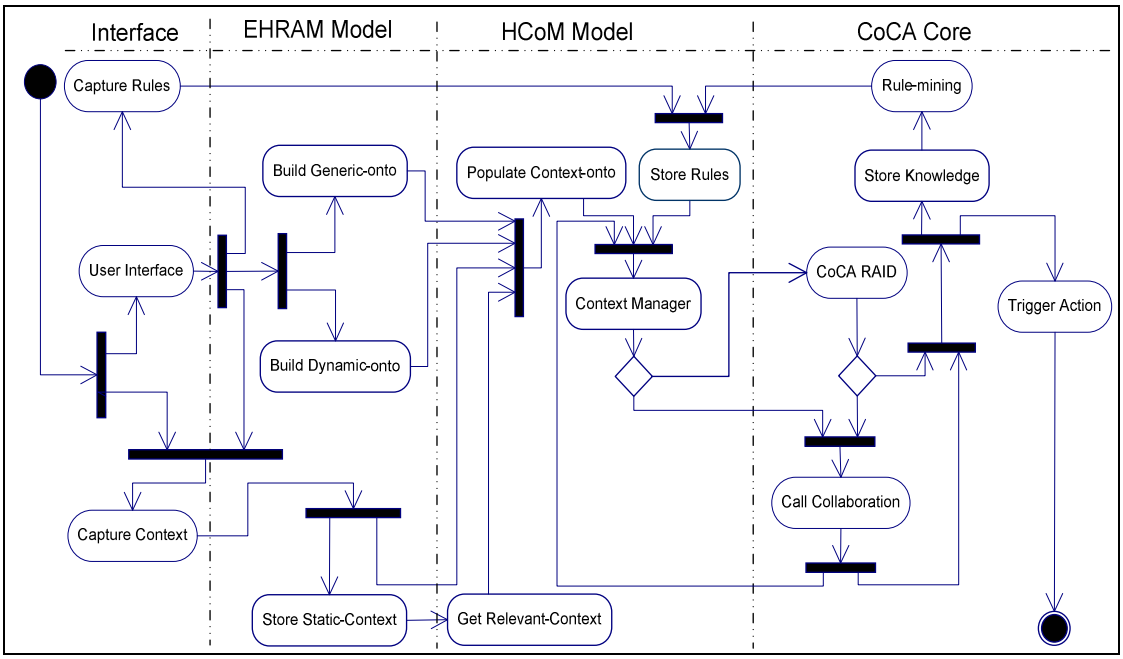


Figure 6-1: Generalized CoCA implementation algorithm

Interface Partition is a collection of hardware and software tools for capturing context data and related decision support rules and pass it to the next component for proper representation. In the diagram, this block consists of *context-capture*, *user-interface* and *rule-capture* with a fork and a join data flow connections indicating states of activities.

EHRAM Model Partition is a collection of data representation tools that exist in the form of EHRAM model that is based on ontology structure and relational schema. In the diagram, this block consists of *Build-Generic-onto, Build-Dynamic-onto,* and *Store-Static-context* with a fork and a join flow connections indicating states of activities.

HCoM Model Partition is a collection of the data management tools in our hybrid context management model (HCoM). It is based on the semantic mapping and aggregation of context data, rules and ontology. Retrieval of only relevant context data from the repository into the context-onto is done in this block. The block consists of *Get-Relevant-context, Populate-Context-onto, Store-Rules* and *Context-Manager* with number of join connections and a decision indicating states of activities.

CoCA Core Partition is a collection of CoCA core service tools. It performs interpretation, aggregation, reasoning, decision and then triggers action (RAID action). CoCA core uses the HCoM context model and a neighborhood collaboration mechanism for devices with scarce reasoning resources. The block consists of CoCA-RAID, Call-Collaboration, Store-Knowledge, Rule-Mining and Trigger-Action with two joins, two forks and a decision indicating states of activities.

6.2 Implementation

Implementation of the demonstration version is intended primarily as a *proof of concepts* in the CoCA/HCoM architecture. We decided to use the Java language environment because in addition to its portability to multiplatform environment, we have open source Java development API tools (Jena and Jxta) that facilitate implementation of the reasoning and collaboration features of the CoCA platform. The implementation consists of approximately 2,000 lines of Java code with intensive use of numerous inherited API classes from the Jena reasoner and the Jxta protocols.

Flexibility, modularity, expandability, and efficiency are among the issues considered during this implementation phase of the proposed platform and context model. To achieve this goal, we structured the classes using Java's *package* concept. Currently, we have one top-level package named as *CoCA* with classes: *CoCA_Interface, CoCA_Collaboration, CoCA_Trigger, CoCA_RAID, CoCA_HCoM, CoCA_RCDB, CoCA_ContextOnto, and CoCA_Capture.* These components are carefully abstracted to make the system independent of the application domain.

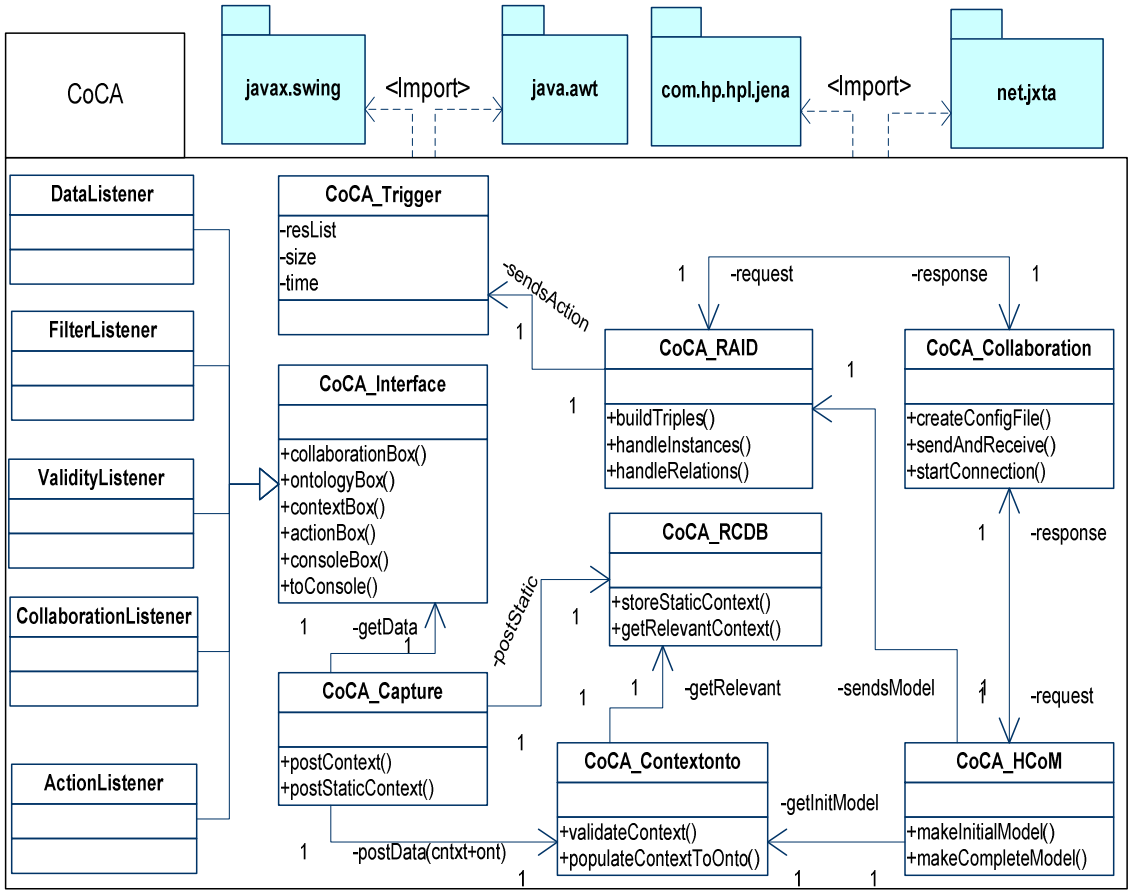


Figure 6-2: CoCA Class Diagram

Table 6-1: Mapping layers in CoCA architecture and the implementation classes

CoCA Layers		Implementation classes	
Application Layer *(Interface)*	↔	CoCA_Interface	*Interface and data/event listeners*
Reasoning and Decision Layer *(CoCA core)*	↔	CoCA_Collaboration	*Connect, get and or receive data*
		CoCA_Trigger	*Trigger actions*
		CoCA_RAID	*Reasoning, decision and action*
Context Management Modeling Layer *(HCoM model)*	↔	CoCA_HCoM	*Create and initialize memory based models*
		CoCA_RCDB	*Context storage and selective retrieval*
Preprocessing Layer *(EHRAM)*	↔	CoCA_ContextOnto	*Context and ontology filtering and merging*
Context Capturing Layer *(capturing tools)*	↔	CoCA_Capture	*Context capturing interface*

Figure 6.2 shows class diagram of our major implementation modules. There are two important APIs used in this implementation: APIs from the Jena framework that supports context reasoning and APIs from Jxta protocols that are used for management of peer collaborations. Mapping between components of the CoCA platform architecture discussed in chapter 5 and the list of CoCA implementation classes is given in Table 6.1.

A demonstration version of the CoCA platform is implemented and is readily available for testing. It can be used to perform reasoning on context data from different domain of applications. All context data that is intended for use in this platform should be organized into the EHRAM/HCoM model. Descriptions of the CoCA package and its class hierarchies are given in Annex II.

6.3 Use case scenarios

We use Protégé ontology editor [Protégé07] to build our ontology, MySQL database management system [MySQL07] with its ODBC-JDBC [ODBC07] as backend data storage for static context data, and Jena framework [Jena07] reasoning and inference tools. We also use Jxta [JXTA07] peer-to-peer protocols for collaboration management. In subsequent sections, we try to show how different smart pervasive scenarios are implemented under the CoCA platform. Details of these implementation tools are discussed in chapter 2 (section 2.4). We use scenarios from three different domains: smart university campus, smart hospital, and dynamic adaptation of computer applications. We finally show a performance evaluation of CoCA and its components.

6.3.1 **Smart campus scenario: PiCASO**

In this section, we discuss the internal functions of CoCA and its RAID Action engine by experimenting on the prototype of our architecture and its context management model, HCoM. We use a *Pervasive Campus-Aware Smart Onlooker (PiCASO)* example.

PiCASO is based on the scenario of a university campus where research students and professors are involved. Besides the scheduled regular meetings among students and professors, informal and spontaneous meetings and discussions are important for the advancement of their work. Discussion can take place among two or more of the researchers depending on the relevance of their work.

The questions that can be raised in this scenario are: When do they make such a meeting? How only those available can be informed about someone else's interest to discuss about a specific subject matter during the tea break. How can a student know that his professor is available for the coming 30 minutes? How can a student know when his professor is in the tearoom and is available, in his office or in the corridor passing by the office of the student? What type of messaging method is appropriate to send such information to a particular person located at a particular place at a particular time? If telephone is used to accept such a message, what should its call mode be (vibrating, ringing)? And so on.

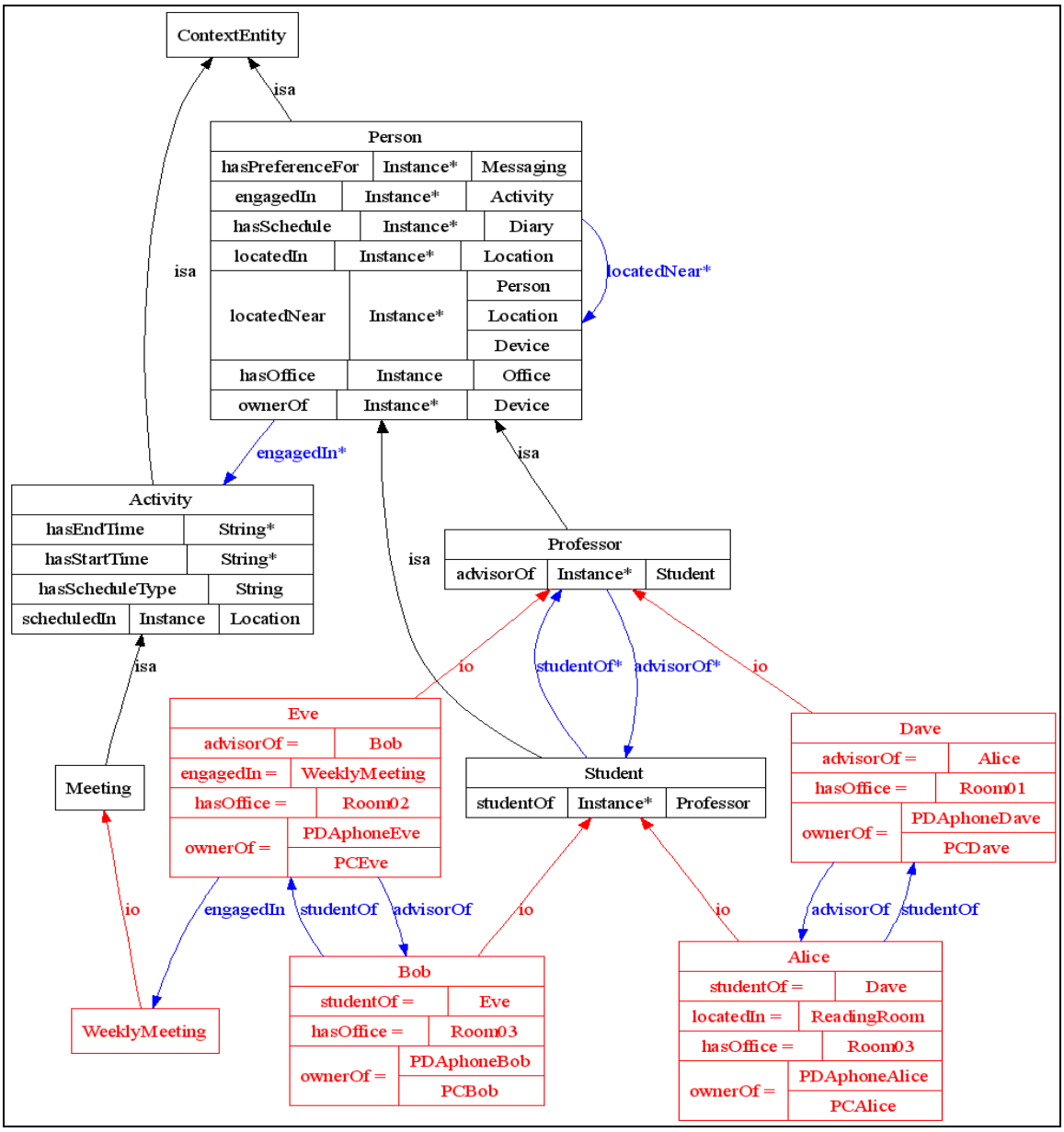


Figure 6-3: Part of context ontology graph for PiCASO scenario

In ontology, we have semantics about all components and related concepts. It also defines relations between the components and the concepts. For example, a person can be defined as an *ownerOf* a device and then the relation *ownedBy* is automatically granted to the reverse relation because both concepts are defined, in the ontology, as being the inverse of one another. One example of such context instance data can be given as: *Bob is ownerOf PDA001*. This means Bob is *ownerOf* a PDA device called PDA001, from which, the ontology reasoner can easily deduce that PDA001 is *ownedBy* Bob. In the GCoM model, persistent data about static contexts (e.g. ownership relationship of persons to devices like telephone or PDA) can be stored as profiled or static context in any standard database format, which can then be selectively populated as context instances into the ontology structure at runtime.

```
1.  Ontology
2.  <xml version="1.0"?>
3.  <rdf:RDF …….
4.    <owl:Class rdf:ID="Student">
5.       <rdfs:subClassOf> <owl:Class rdf:ID="Person"/> </rdfs:subClassOf>
6.    </owl:Class>
7.    <owl:Class rdf:ID="Library">
7.       <rdfs:subClassOf> <owl:Class rdf:about="#Location"/> </rdfs:subClassOf>
8.    </owl:Class>
9.    <owl:ObjectProperty rdf:ID="ownedBy">
10.      <rdfs:range rdf:resource="#User"/>
11.      <rdfs:domain rdf:resource="#Device"/>
12.      <rdf:type rdf:resource="http://www.w3.org/.../owl#FunctionalProperty"/>
13.     <owl:inverseOf> <owl:ObjectProperty rdf:ID="ownerOf"/> </owl:inverseOf>
14.   </owl:ObjectProperty>
15.   <Student rdf:ID="Bob">
16.      <ownerOf>
17.           <PDA rdf:ID="PDA001">
18.             <hasScreenSize
19.      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Medium
20.        </hasScreenSize>
21.      </PDA>
22.      </ownerOf>
23.      <ownerOf rdf:resource="#Cellphone001"/>
24.   </Student>
25. …...
26. </rdf:RDF>
```

Figure 6-4: An excerpt from the PiCASO ontology

To demonstrate context data representation in PiCASO, we use the ontology based generic context management model GCoM and its enhanced version HCoM. The three important data sources related to the data model are context ontology, context data and rules. Graphical representation of part of the context ontology for the PiCASO is given in

Figure 6.3. An equivalent OWL representation of a part of the context ontology for PiCASO is given in Figure 6.4.

```
1.  //Context
2.  <xml version="1.0"?>
3.  <rdf:RDF …..
4.  <Professor rdf:ID="Dave">
5.       <locatedIn rdf:resource="#Room01"/>
6.       <engagedIn rdf:resource="#FormalMeeting"/>
7.  </Professor>
8.  <Student rdf:ID="Carol">
9.       <locatedIn rdf:resource="#ReadingRoom"/>
10. </Student>
11. <Student rdf:ID="Alice">
12.      <locatedIn rdf:resource="#Room03"/>
13. </Student>
14. <Professor rdf:ID="Eve">
15.      <locatedIn rdf:resource="#Room02"/>
16.      <engagedIn rdf:resource="#Browsing"/>
17. </Professor>
18. <Student rdf:ID="Bob">
19.       <owns rdf:resource=”#PDA001”/>
20.       <owns rdf:resource=”#CellPhone001”/>
21. </PDA>
22. <PC rdf:ID="PCAlice">
23. </PC>
24. <Messaging rdf:ID="MessengerService">
25. </Messaging>
26. …….
27. </rdf:RDF>
```

Figure 6-5: An excerpt from PiCASO context representation

Sensed context is to be communicated using XML/RDF-triple representation format as indicated in Figure 6.5. Sample rules for representing explicit wishes are given in Figure 6.6. Such data are stored in a disk file that can be in any suitable format: text or any Jena compatible database format like MySQL. After semantically combining data from all the three sources, the reasoner can draw parameters for the actions in the PiCASO.

In this example, we use SPARQL, the RDF query language supported by the Jena framework. SPARQL is becoming more and more accepted by the user community due to its stronger definition and better implementations. W3C is working towards standardizing SPARQL.

Sample SPARQL query to select the phone for which the action has to be triggered and set its ringing tone to “Silent” mode setting can be given as:

```
Query
SELECT  ?phone
WHERE{
   coca:Bob coca:owns ?phone.
   ?phone coca:setRingTone coca:Silent.
}
```

```
1.  //Rules
2.  //Ontology based derived rules
3.  [Transitive_Rule: (?a coca:locatedIn ?b)
4.    (?b coca:locatedIn ?c)
5.  ->(?a coca:locatedIn ?c)]
6.  [Inverse_Rule: (?a coca:ownerOf ?b)
7.       ->(?b coca:ownedBy ?a)] //inverse
8.  …..
9.  // Domain based phone management rules
10. [locatedRule:(?device coca:locatedIn ?location)
11.  (?device coca:ownedBy ?person)
12.  -> (?person coca:locatedIn ?location)
13. ]
14. [libraryRule:(?student coca:locatedIn coca:Library)
15. (?student coca:owns ?phone)
16.  -> (?phone "setRingTone" "silent")
17. ]
18. [classRule:(?student coca:hasSchedule ?class)
19. (?class coca:isScheduledIn ?classRoom)
20. (?class coca:startTime ?t1)
21. (?class coca:endTime ?t2)
22. ((?Student coca:locatedIn  ?classRoom) coca:hasTime ?t)
23. (?t sys:greaterThan ?t1)(?t sys:lessThan ?t2)
24. (?student coca:owns ?phone)
25. → (?phone "switchMode" "Vibrating")
26. ]
27. [meetingRule:(?student coca:hasSchedule ?meeting)
28. (?meeting coca:scheduledIn ?meetingRoom)
29. (?meeting coca:startTime ?t1)
30. (?meeting coca:endTime ?t2)
31. ((?student coca:locatedIn ?meetingRoom) coca:hasTime ?t)
32. (?t sys:greaterThan ?t1) (?t coca:lessThan ?t2)
33. (?student coca:owns ?phone)
34. →(?phone "switchMode" "Silent")
35. ]
36. xcampusRule:(?Student coca:locatedIn OutSideCampus)
37. (?student coca:owns ?phone)
38. → (?phone "switchMode" "MusicRingingTone")
39  ]
```

Figure 6-6: An excerpt from PiCASO rule representation

Table 6-2 shows a simple trace of a library rule in the above example as demonstration of the reasoning process. It shows an output message with parameters *(CellPhone001, setRingingTone, SilentMode)* that are used in a call to a customized action trigger module.

In this particular case, the action is to set the ringing mode of Bob's cellPhone to silent mode.

Table 6-2: Trace of reasoning process using the LibraryRules

(PDA001 locatedIn ReadingRoom)	Sensed context
=>(PDA001 locatedIn Library)	By rules in lines 3-5, Fig. 6.6.
=>(Bob locatedIn Library)	By ontology in lines 9-14, Fig. 6.4 (inverse property owns and ownedBy defined in the ontology), context in lines 18-21, Fig. 6.5 and rules in lines 10-13, Fig. 66.
=>(CellPhone001 SetRingTone silentMode)	By context in lines 18-21, Fig. 6.5 and rules in lines 14-17, Fig. 6.6.

Figure 6.7 shows a java code that put together all the major components of the CoCA service platform for reasoning, inferences and decisions. It also indicates how SPARQL queries are used to draw parameters for action triggering. This simple example demonstrates how CoCA platform is used in building a context-aware service.

All input components used in our platform (contexts, ontology and rules) are not hard coded into the system and are stored separately. This indicates that the platform can be used in multiple domains by simply changing the inputs.

PiCASO scenario implementation involved varying number of handcrafted context instances (200 up to 6,000) created from the combination of context entities and relations defined in the campus domain. We have used up to 100 defined relations, 80 activity instances, 100 device instances, 100 location instances, 3 network instances, 100 instance of persons and 30 instances of services to generate this context data set. Such data instances are stored in the relational context database, RCDB. Only relevant context data is then loaded from RCDB into the reasoner space during initialization. It also use varying number of rules (20 to 200 lines) created from domain policies and user needs. We have also used varying lines of dynamic context instances that not only guide the reasoning process but also trigger the reasoner proactively. Combining all these with the owl ontology schema, the PiCASO HCoM model produces up to 9,000 context triples in the entire reasoning space. Figure 6.8 shows sample screen shots of the CoCA SMART assistant for the PiCASO scenario. Comparison on the effect of using varying data size in HCoM and GCoM approaches will be given in the evaluation section of this chapter (section 6.5).

```
//import APIs…
.. ..
public class CoCASupport{
  public static void main(String[] args) {

   //Section for loading and configuring CoCA model

   //Load rules
   GenericRuleReasoner reasoner = new GenericRuleReasoner((List)
          Rule.rulesFromURL(    "file:ForumRules.rules"));
   //Load context ontology with an already selected populated context
   //data and create a memory-based model ready for reasoning application
   OntModel tempModel = ModelFactory.createOntologyModel(
               OntModelSpec.OWL_MEM_MICRO_RULE_INF,
     ModelLoader.loadModel("file:ForumOntology.owl"));
   //Combine and map rules into ontology
   InfModel cocaModel=ModelFactory.createInfModel(reasoner,tempModel);

   //Example usage
   //Section for query formtion

   //Definition of the CoCA name space
   String queryString = "PREFIX coca: <http://www.owl" +
   "-ontologies.com/unnamed.owl#> "
   "SELECT ?phone WHERE {?phone coca:setRingTone coca:Silent.}";
   Query cocaQry = QueryFactory.create(queryString) ;
   QueryExecution qexec=QueryExecutionFactory.create(cocaQry,cocaModel) ;
   ResultSet cocaResults = qexec.execSelect() ;

   //Extract query results
   for ( ; cocaResults.hasNext() ; )
   {
     QuerySolution res = cocaResults.nextSolution() ;
     RDFNode phone = res.get("phone") ;
     System.out.println("Setting ringing tone of "+ phone +" to silent");
     fireAction("RingingTone", phone,"silent"); //Module Call
   }
   qexec.close();
 }
}.. ..
```

Figure 6-7: An excerpt of code for creation and initialization of the CoCA data model

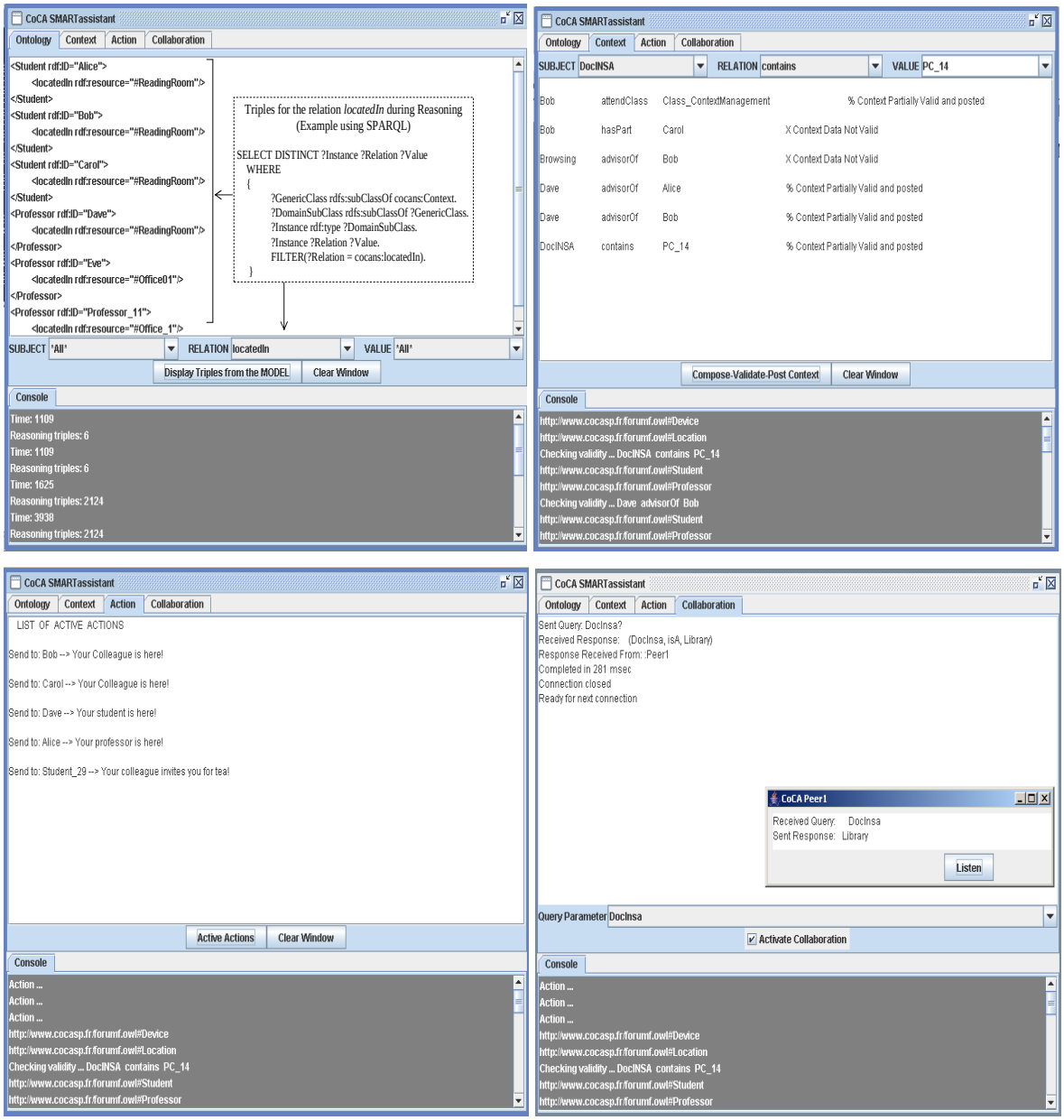


Figure 6-8: Screen shots from the PiCASO scenario in the CoCA platform

6.3.2 **Smart hospital scenario: patient monitoring and follow-up**

Consider a smart medical ward in a hospital (section 1.2) where patients, nurses and physicians, etc. are involved. The ward is equipped with context sensor technologies in its rooms, corridors and garden at the disposal of individuals involved. Patients admitted to the hospital may need intensive follow up which may create staff shortage and may result in inappropriate care to the needy ones due to overloading. A context-aware monitoring and follow up system helps to minimize the engagement of human assistants to the less important activities.

Human interventions may be needed only when alerted by the system. Live multimedia recording and transmission of an event that the system has found important may also be used for monitoring purposes. Delivery can be made to those who are concerned after adapting such contents to their context. Figure 6.9 shows part of the context ontology for the hospital scenario.

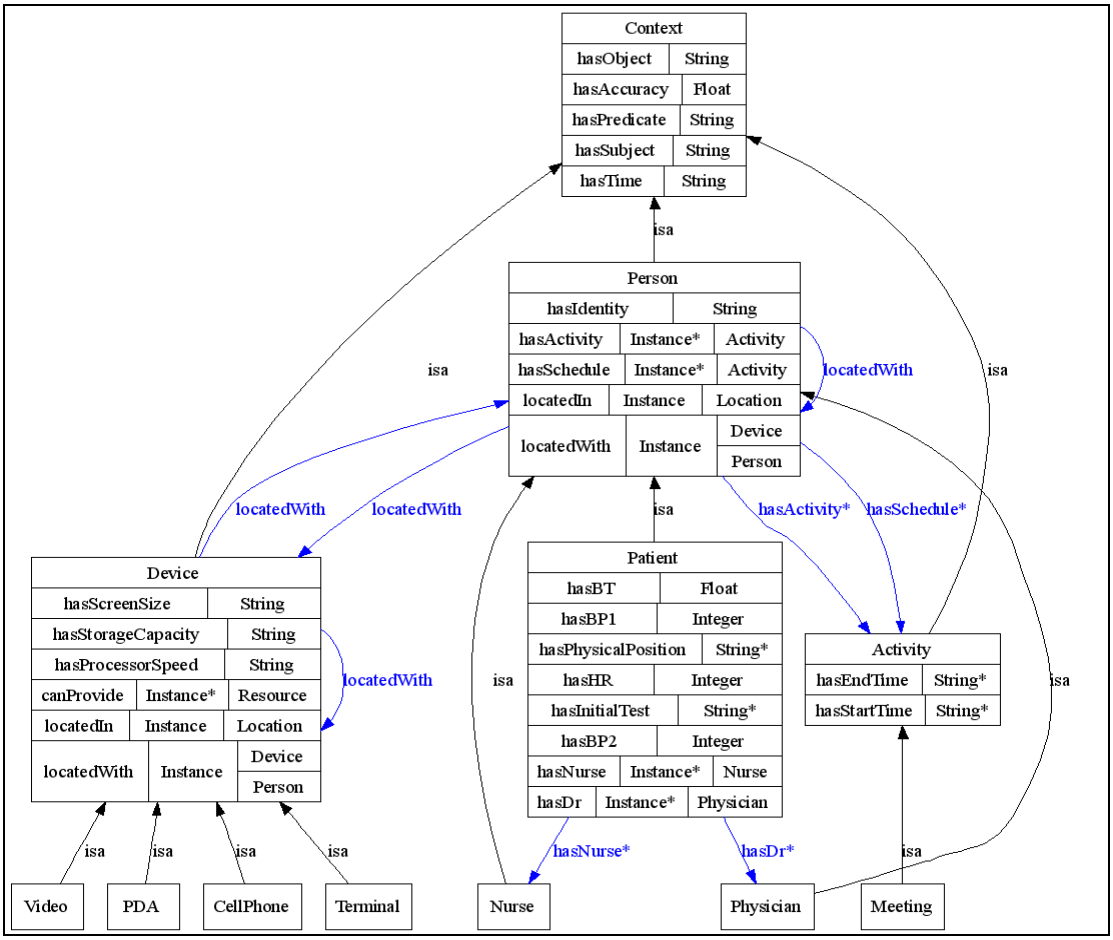


Figure 6-9: Part of context ontology for the hospital scenario

The implementation of this scenario is one step towards a real application because we have used the real world clinical concepts. The ontology of the scenario is built on top of the medical terminologies from the OpenGALEN project [OpenGalen07]. OpenGALEN is a not-for-profit organisation that provides downloadable open source medical terminologies and tools. An early phase of the GALEN programme was funded by the EU (the SESAME & OAR projects) in late 1980s. The OpenGALEN project is now a Dutch Foundation that runs jointly by the Victoria University of Manchester in the United Kingdom and the University of Nijmegen in The Netherlands.

Actors in our medical ward scenario are linked to the ontology as part of the ontology classes or class instances. For example, *medicalward* and *garden* are sub classes of the *location* class where as specific *garden-name* or *room-number* in the scenarion is the location instance. *Doctor, nurse, patient* and *supportstaff* are examples of sub classes of the person class where as *Michel, Pascal* and *Ada* are instances of the class person. Each instance inherits its role and property from the immediate parent class and similarly each class inherites properties and roles from its parenet class.

Context data and rules for patient care in the scenario like below are semantically aggregated with the ontology for appropriate actions and decisions.

```
<Doctor rdf:ID="Pascal">
   <locatedIn rdf:resource="#MeetingRoom_01"/>
   <engagedIn rdf:resource="#WeeklyMeeting"/>
</Doctor>
<Patient rdf:ID="Michel">
   <locatedIn rdf:resource="#BackDoorGarden"/>
</Patient>

[EmergencyRule:
   (?Patient coca:locatedIn ?loc)
   (?loc  rdf:type  coca:Garden)
   (Patient coca:hasStatus ?Status)
   (?Status rdf:type coca:Emergency)
   (?status coca:hasMessage ?Msg)
   (?Patient coca:hasDoctor ?Doc)
→(?Msg coca:sendMessage ?Doc)]
```

The advantage of embedding the OpenGALEN clinical concepts into the HCoM model extends the application of the scenario to a more complex decision support tool even for the doctors and the nurses in their regular clinical process. Knowing about the principal clinical procedures and concepts like pathological, anatomical and pharmaceutical semantics, CoCA can be used for reasoning and suggestion of actions towards patient care and treatment. Both textual and graphical representations of the extract of the ontology are given in Annex VI.

6.3.3 **Adaptation scenario: adaptation of applications to context**

The importance of context information in the process of content adaptation has been well studied and demonstrated by a colleague at our research team in his PhD work [Berhe05]. Adaptation of software applications to context is another aspect of context-aware computing in pervasive systems. Details of the work we use here as one of the application scenarios of the CoCA platform is given in our article [Chaari07]. For example, if a display

unit of a device doesn't support images, and if the user selects to view an image on this device, the application should automatically lead the user to the textual description of the image which means the displayImage() module in the application must be locked by the context-aware service. In our case study, to use the context ontology to represent the concepts in the process of adapting applications to new context situations, we have added a new domain class (sub entity) named *Application* under the *Service* base ontology class (entity) of context-onto in HCoM.

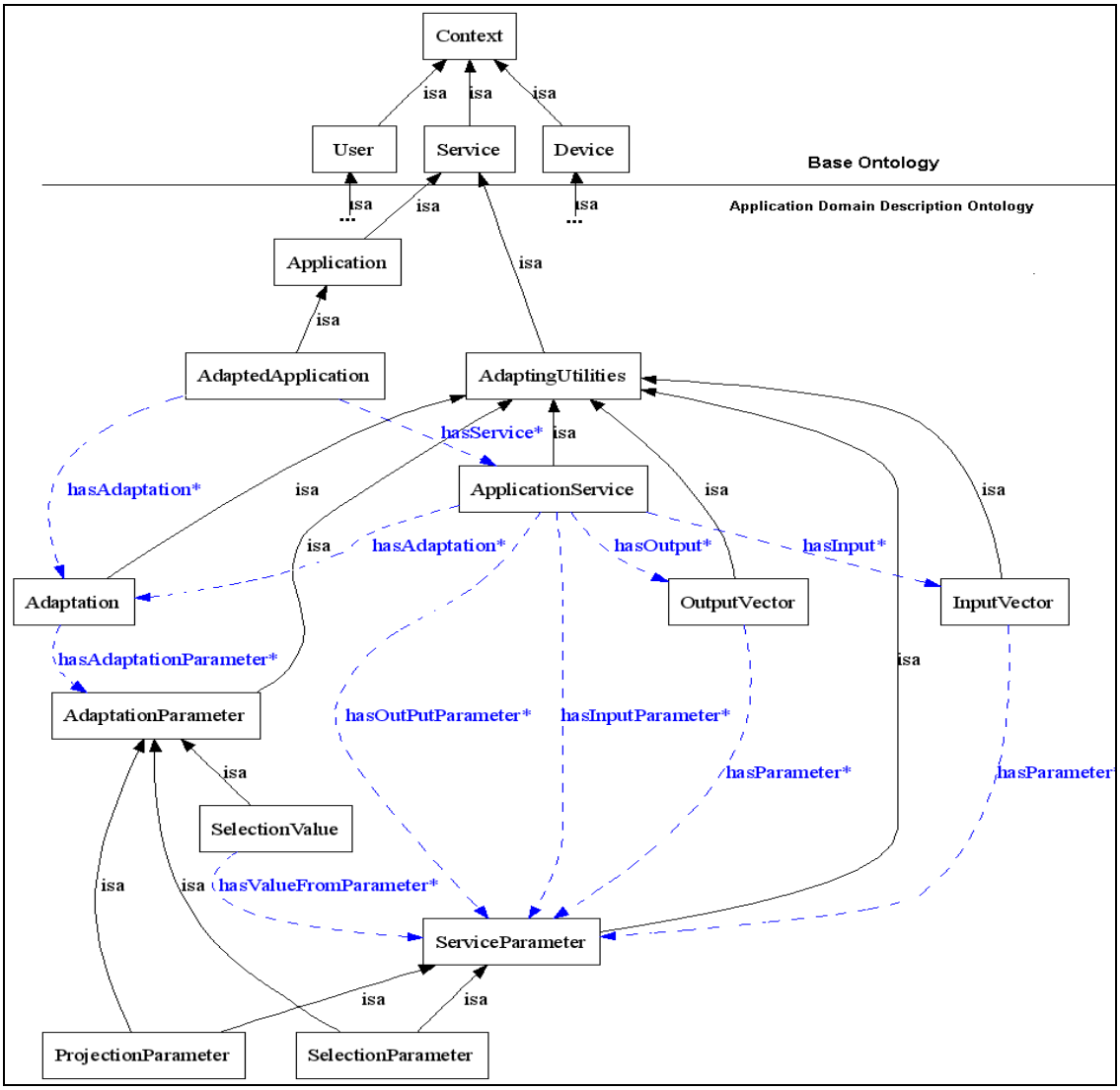


Figure 6-10: Example of context ontology for adaptation of application

Figure 6.10 shows the relationship between the *generic context ontology* part and *the application domain ontology* part. This relation keeps hierarchy of context entities in our adaptation process to describe the services of the application and the different adaptation operators that we have defined in previous sections. In addition to the components indicated in this figure, we have also defined other subclasses under the base ontology classes and

their relationship with one another. Under *Device* class, for example, we have a sub class *Terminal* which possesses the *hasSize* property.

In general, it is up to the application designer or the user to add or remove the domain specific classes, sub classes, domain based class properties and relationships using domain definition interface. Adaptation rules based on changing context can also be defined by designers or users using this interface. The context manager module is then responsible to use the knowledge in the ontology to activate adaptation.

The adaptation processes that can be applied in a specific context situation are defined by a combination of predicates and facts on the *application domain description part* and its components (application, services and their output and input parameters). To integrate the knowledge required for adaptation within the context ontology, rules in our application are described using the Jena generic rule. These rules are easily transported and integrated into OWL. The Jena parser and query engine tools give an easy interpretation and access to the content of the OWL representation of the context ontology, rules and context data.

For example, if a display unit of a device doesn't support images, and if the user selects to view an image on this device, the application automatically leads the user directly to the textual description of the image. The displayImage() service of the application is locked by using some domain based rules.

6.4 Demonstration on reasoning in CoCA using PiCASO

Given the PiCASO ontology with its context data and some inter PiCASO-community messaging rules, we will try to show how the occurrence of a simple location based contexts trigger a messages dispatch (action). Figure 6.11 is a segment of PiCASO context ontology showing the *advisorOf* relationship that exists among the three professors (Eve, Professor_11, Dave) and the five students (Carol, Bob, Student_23, Alice, Student_21). Some of the nodes in the graph show slots indicating currently existing relationships. For example, the node for Professor Dave shows the OfferClass, advisorOf, engagedIn, hasOffice and ownerOf relationships and their values. The term *isa* in the graph represents the built in rdfs:subClassOf relationship and similarly *io* (instance of) represents the built in rdf:type relationship. In the ontology, the relationships *advisorOf* and *studentOf* are defined to be the inverse of one another. This means wherever the relationship *advisorOf* exists, it also holds true for *studentOf* in the reverse direction.

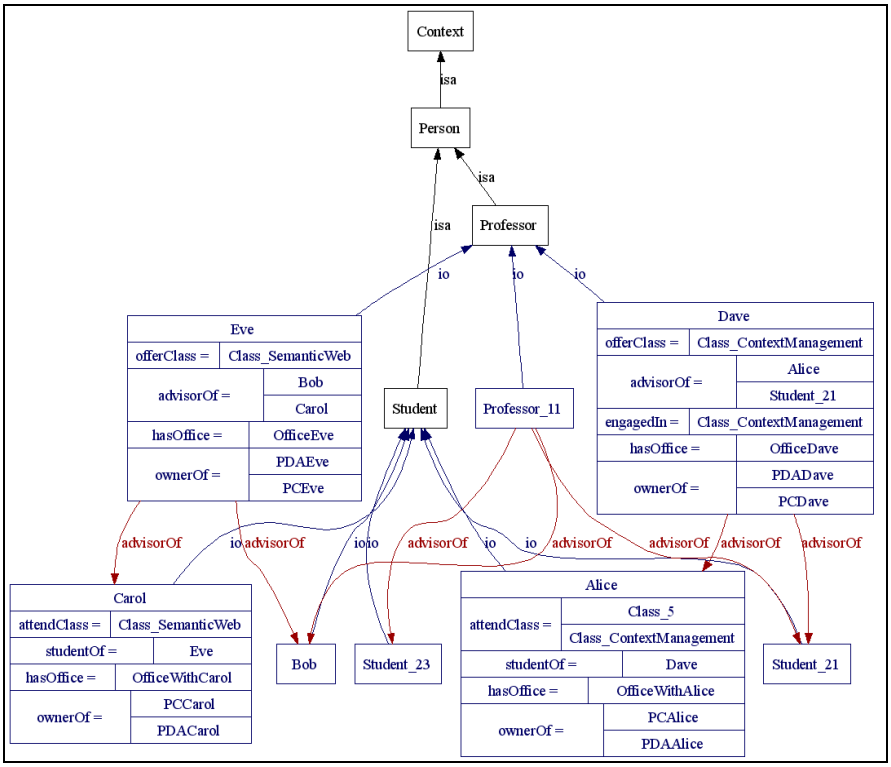


Figure 6-11: PiCASO context ontology showing some advisorOf relationships

```
# Send trigger message
[InformRule1:
   (?S rdf:type pre:Student)
   (?P rdf:type pre:Professor)
   (?S pre:studentOf ?P)
   (?S pre:locatedWith ?P)
->(?S pre:hasMessageTogo
      pre:ProfessorHere)
]
[InformRule3:
   (?S1 rdf:type pre:Student)
   (?S2 rdf:type pre:Student)
   (?P rdf:type pre:Professor)
   (?S1 pre:studentOf ?P)
   (?S2 pre:studentOf ?P)
   (?S1 pre:locatedWith ?S2)
   notEqual(?S1,?S2)
->(?S1 pre:hasMessageTogo
      pre:ColleagueHere)
[InviteRule1:
   (?S1 rdf:type pre:Student)
   (?S2 rdf:type pre:Student)
   (?P rdf:type pre:Professor)
   (?P pre:advisorOf ?S1)
   (?P pre:advisorOf ?S2)
   (?S1 pre:engagedIn ?A1)
   (?A1 rdf:type pre:TeaBreak)
   (?S2 pre:engagedIn ?A2)
   (?A2 pre:hasScheduleType ?t)
   equal(?t,pre:relaxed)
   notEqual(?S1,?S2)
->(?S2 pre:hasMessageTogo
      pre:ColleagueForTea)
]
```

```
#Set mobile ringing tone
[MobilePhoneRule1:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:Library)
  ->(?d pre:setNotificationMode
  pre:SilentMode)
]
[MobilePhoneRule2:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:ClassRoom)
  ->(?d pre:setNotificationMode
  pre:VibratingMode)
]
 [MobilePhoneRule3:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:BreakRoom)
  ->(?d pre:setNotificationMode
  pre:MusicTone)
]
 [MobilePhoneRule4:
   (?d pre:ownedBy ?p)
   (?d rdf:type pre:PDA)
   (?p pre:locatedIn ?l)
   (?l rdf:type pre:Office)
  ->(?d pre:setNotificationMode
  pre:RingTone)
]
```

Figure 6-12: Some PiCASO rules for message trigger and ringing tone

Column 1 in Figure 6.12 is about the rules that govern message triggers. If we look at the first rule in this figure, it says "*if the set of captured contexts are aggregated to give us a new context about the presence of a student and his/her professor in the same location then set value of MessageToGo for the student to the a message named ProfessorHere."* Context aggregation process, for example, works by checking location of the professor and location of the student and decide if they are located in the same place or not. The message *ProfessorHere* by itself is an object in which a value, an executer module, etc. are defined. The content of the message in this particular case is *"Send to: Student --> Your professor is here!"*. We can have plenty of such application based policies or rules. Another set of rules that we use in this demonstration is about cell phone ringing tone management, column 2 in figure.

HCoM is then used to create a memory based reasoning model that can be queried using the SPARQL query language for final action. As a demonstration of this concept, in this particular example (Figure 6.13), a segment of SPARQL query (lines 9 to 12) embedded into a java code to extract the content of *hasMessageToGo.* Lines 2 to 7 are standard query headers indicating URLs for RDF, OWL, RDFS, etc. Line 8 indicates URL for the CoCA model on which the query is applied. The rest indicates part of the code that deals with utilization of the result from the query.

Table 6-3: Examples of new context data and the resulting set of action triggers

New context	Expected action to follow from the ontology, rules and context data given in this demo
- Carol engagedIn Email - Bob engagedIn MorningTea	- Send to: Carol --> *Your colleague invites you for tea*
- Alice locatedIn ReadingRoom - Student_21 locatedIn ReadingRoom	-Send to: Student_21 --> *Your Colleague is here* -Send to: Alice --> *Your Colleague is here* -Send to: PDA_21 --> *No ringing tone on arrival of call* -Send to: PDAAlice --> *No ringing tone on arrival of call*
- Bob locatedIn AmphiRoom1 - Eve locatedIn AmphiRoom1	Send to: Eve --> *Your student is here* Send to: Bob --> *Your professor is here* Send to: PDAEve --> *Vibrate on arrival of call* Send to: PDABob --> *Vibrate on arrival of call*

```
1. public String[] getActionsToGo() {
2. String queryString = ""+
3. "PREFIX rdfsyntax: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> "+
4. "PREFIX xmlschema: <http://www.w3.org/2001/XMLSchema#> "+
5. "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> "+
6. "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
7. "PREFIX owl: <http://www.w3.org/2002/07/owl#> "+
8. "PREFIX coca: <http://www.cocasp.fr/forumf.owl#> "+
9. "SELECT DISTINCT ?Object ?Cntnt "+
10."WHERE "+
11."{ "+
   11.1.  "{ ?Object coca:hasMessageTogo ?Msg."+
   11.2.  "?Msg coca:hasContent  ?Cntnt.} "+
   11.3.  "UNION "+
   11.4.  "{ ?Object coca:setNotificationMode ?Msg."+
   11.5.  "?Msg coca:hasContent  ?Cntnt.}"+
12."}";
13.Query query = QueryFactory.create(queryString) ;
14.QueryExecution qexec=QueryExecutionFactory.create(query,CoCAsystem.model);
15.ResultSet results = qexec.execSelect();
16.int k=0;
17.String[] qryResult=new String[MaxSize];
18.while (results.hasNext())
19.{
   19.1.  QuerySolution res = results.nextSolution() ;
   19.2.  RDFNode P = res.get("?Object");
   19.3.  RDFNode C = res.get("?Cntnt");
   19.4.  String person=trimResult(P.toString());
   19.5.  String content=trimResult(C.toString());
   19.6.  qryResult[k]= "Send to: " + person + " --> " + content+"\n\n";
   19.7.  k++;
20.}
21.qexec.close();
22.CoCAsystem.size=k;
23.return qryResult;
24.}
```

Figure 6-13: Segment of code that shows SPARQL query on HCoM model

Table 6.3 shows demonstration of sets of new context data (left) and the resulting set of action triggers (right). In order to provide a proactive service on every occurrence of new context, the CoCA decision engine goes through aggregation and reasoning process. The aggregated context data is buffered along with the initial context data for the next use.

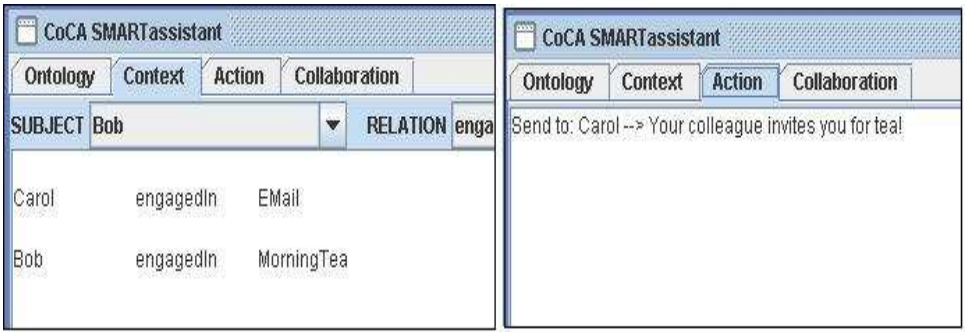


Figure 6-14: Part of the screen showing new context data and the resulting action

The CoCA platform provides an interface for running and testing decisions and actions of such type. We have run and verified that all the expected actions are also executed in a similar manner in the platform. Figure 6.14 shows screenshots of a demonstration of one of the examples.

As a conclusion, we have used a walkthrough example from PiCASO to demonstrate how the CoCA reasoning process works. The effect of the occurrence of a new context data on action trigger is demonstrated using examples from messaging service and mobile phone ringing tone management service.

6.5 Measuring performance

Machine based reasoning is a time intensive process that has exponential time complexity with respect to the size of data instances in the reasoning space [Zuo06]. To overcome the expensive time complexity problem, HCoM uses pruning technique (section 4.6.4) for selective loading of only relevant reasoning resources or pruning of the irrelevant branches from the hierarchy of the reasoning graph. This means, the selection module loads only relevant context information from the HCoM context repository, RCDB, into the reasoning space during every initialization of the CoCA system.

To evaluate HCoM and GCoM with respect to their response time and to compare their performances against one of the similar works, CONON from Wang et al, we have conducted the following experiment.

Our experimental data comes from the PiCASO implementation (Section 6.3.1). The experiment is carried out both for GCoM and HCoM. Experiment for GCoM is concerned with reasoning on an occurrence of a new context instance using the entire context data loaded into the reasoning space (with out pruning). Triggering of a *message-dispatcher* based on the presence of two students having the same professor in the same location (demonstrated in section 6.3.4) is one example of an action resulting from reasoning. During our experiment, such an occurrence of a new context and the response time to get the action message is recorded repeatedly by varying the data size in the reasoning space.

Table 6-4: Summary of response time from the experiment

Experiment		Data size ranging from 200 to 9,000											
GCoM	Size	200	800	1600	2300	3000	3800	4535	5268	6000	6733	7466	9010
Time (ms)	Round1	411	1032	1527	1906	2209	2427	2942	3365	3931	4576	5355	5914
	Round2	434	971	1465	1888	2258	2481	2944	3373	3931	4582	5384	5911
	Round3	483	1014	1520	1904	2255	2456	2960	3389	3964	4625	5350	5877
	Round4	486	1001	1518	1864	2248	2463	2966	3422	3960	4614	5333	5924
	Round5	436	982	1470	1888	2235	2433	2908	3426	3949	4593	5328	5879
	Average	450	1000	1500	1890	2241	2452	2944	3395	3947	4598	5350	5901
	St. Dev.	33	24	30	17	20	22	23	28	16	21	22	22
HCoM	Size	200	800	1600	2300	3000	3800	4535	5268	6000	6733	7466	9010
Time (ms)	Round1	525	788	725	799	869	992	1013	1089	1189	1249	1293	1419
	Round2	514	705	748	825	931	936	1068	1083	1189	1238	1363	1374
	Round3	470	742	705	858	871	986	1023	1136	1176	1284	1326	1424
	Round4	478	743	711	805	897	977	1028	1121	1205	1274	1353	1373
	Round5	513	722	761	843	922	959	1078	1141	1171	1245	1315	1420
	Average	500	740	730	826	898	970	1042	1114	1186	1258	1330	1402
	St. Dev.	24	31	24	25	28	23	29	27	13	20	28	26

We have repeated similar experiment for the HCoM model. In HCoM, the context data are loaded into the reasoning space selectively by pruning the irrelevant part of the data. We have made repeated experiments with varying data size (ranging from 200 up to 9000 RDF triples incremented by progressive intervals – sample of context data, rules and ontology generated for our experiment are shown in Annex III, IV and V).

The average response time collected for different context data instances is used for both GCoM and HCoM. Table 6.4 shows a summary of data for response time (in milliseconds) with calculated average and standard deviation for each round of the experiment. The overall average standard deviation (24ms) is small enough to conclude that the system is stable except due to some other jobs sharing the processor at the time of the experiment. We have collected data from five of our experiments and twelve progressively incremented data size both for GCoM and HCoM models.

The use of GCoM model gives a response time that grows fast with the increase of volume and complexity of reasoning data. The use of HCoM, on the other hand gives a response time that tends to remain nearly constant with the growth in the volume and complexity of context data. This exiting improvement in a response time, however, is not for free, it is at the expense of building an efficient search mechanism (heuristic based) that estimates the appropriate user intension or, other wise, the cost will be a compromise on the quality of the reasoning process. Details on this issue is presented in section 4.6.4. Figure 6.15 shows a graph for the result of our experiment using both the GCoM and the HCoM modeling approaches.

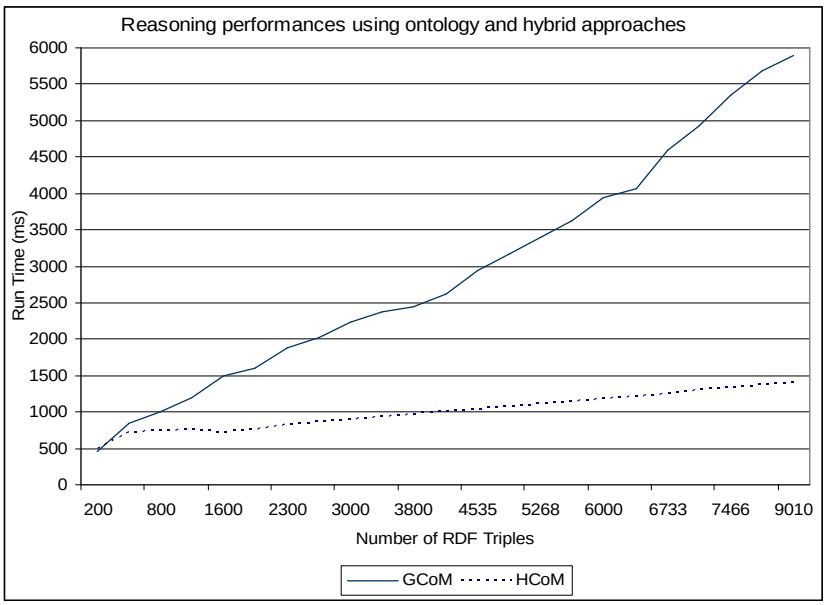


Figure 6-15: Reasoning performance for GCoM and HCoM (2x1.83 GHz CPU)

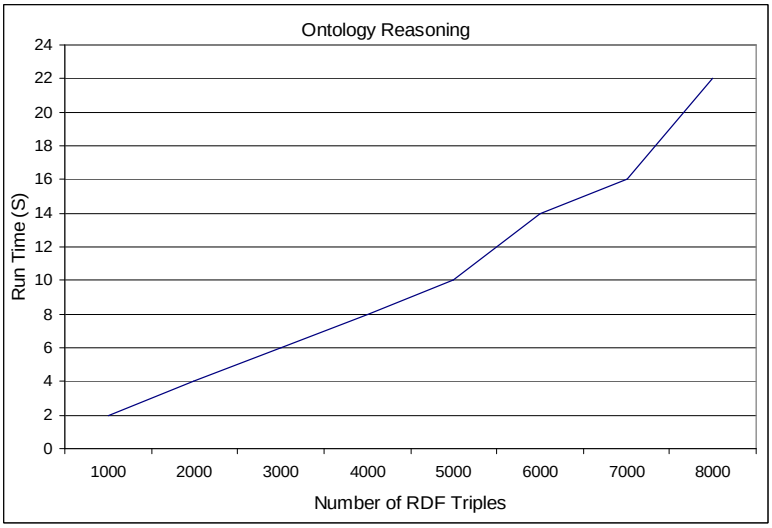


Figure 6-16: Reasoning performance for CONON (2.4 GHz CPU)

Similar experiment were made by Wang et al on their CONON context ontology, Figure 6.16, shows that the time complexity of context reasoning based on ontology tends to grow fast like that of our GCoM model. Both graphs have the same trend but have different values due to the difference of the CPU speeds used during the experiment.

As a conclusion, our experiment shows that HCoM improves the performance of CoCA service and makes it a scalable and extensible platform. HCoM model, its basic constructor, the EHRAM model and the CoCA platform, therefore, are promising context management and reasoning tools in pervasive context-aware environments where most of the devices have limited capacity.

Chapter 7 CONCLUSIONS AND FUTURE WORKS

7.1 Summary of contributions

We have developed novel specifications for context representation, context modeling and context management. We have also presented specification for a domain-independent context-aware platform as a middleware that enables application developers to utilize easily context information. Our major contributions in this work can be summarized as follows:

A semantic context representation model: We have proposed EHRAM, a conceptual context representation metamodel based on hierarchy of descriptors of context entities. It is represented using layered and directed graph. Hierarchies in EHRAM are important structures to organize and classify context entities and relations. Layered organization also helps to classify and tag context data as generic domain independent or as domain dependent. EHRAM can be easily serialized to a standard markup languages for storage, retrieval, transmission, processing. In our case, we use RDF triples and its derivatives to represent basic structures in the model. Context metadata and axioms in EHRAM are represented using ontology and RDF reification principles.

A hybrid context management model: We have proposed HCoM, a generic context management model based on a hybrid approach. HCoM is an upgrade from our initial model named GCoM. Our rational behind the need for a hybrid context model is to distinguish the works of context data management and context semantic management, process them separately and put the results together for better reasoning and decision support in a context-aware environment. We use ontology approach to manage context semantics and relational approach to manage context data. HCoM model aims to combine the bests from the two worlds. HCoM provides a means to select and load only part of the large static context data that is accumulated over a period of time depending on who and where the user is, the intended activity to which the user is going to be engaged, devices available for use, institutional policies etc. It uses matching patterns gained through experience to identify relevant context data. Loading only relevant data for reasoning minimizes the size of the

reasoning space and reduces the unnecessary overloading of the reasoner so as to improve the overall performance of the context-aware service. It helps to overcome limitations of lack of scalability of most of the reasoning systems to the ever-increasing volume of reasoning resources in the pervasive environment.

A collaborative context-aware service platform: We have developed a CoCA platform. CoCA is a neighborhood-based system in pervasive computing that aims at acquiring and utilizing context information to provide appropriate services. CoCA is domain independent middleware support for application developers that enable them to use context information without the overheads of caring on how to manage it. CoCA platform is independent of the data elements from application domain. We have successfully implemented a demonstration version of the CoCA platform. Demonstration of the function of CoCA platform and the HCoM model is given using different application scenarios. Among these is a Pervasive Campus-Aware Smart Onlooker (PiCASO) scenario in a university campus where research students and professors are involved. We have developed a test procedure for PiCASO using CoCA.

7.2 Conclusions

Most of the current context management systems and context-aware services are based on ad-hoc models and domain dependent applications that may cause lack of desired formality, genericity and expressiveness. Hence, management and use of context information and development of context-aware services in pervasive environments is still a challenge. Computing devices in pervasive computing environment on the other hand are tiny and have limited hardware resources. In this work, we have shown the importance of generic context modeling for efficient context reasoning in a context-aware computing environment. Our approach is based on context entities, hierarchy of entities, relations between entities, axioms and metadata. We have proposed the EHRAM conceptual context representation model based on these components.

For a lightweight context reasoning service, we have proposed a generic context management model named as GCoM. GCoM is ontology based context management model. From the performance evaluation, we made, we have found that GCoM has scalability problem for large volume of context data. This is because GCoM uses ontology data model that is stored and processed using text based markup languages. Therefore, we have further

investigated and proposed HCoM, a hybrid model that satisfy both the need of semantic processing and ease of dynamic context data transaction.

HCoM is our innovative context management model for aggregating and presenting context data from the EHRAM context representation model. It is an upgrade to GCoM. HCoM, like GCoM, is generic. HCoM is hybrid because it uses ontology for management of context semantics and database principles for management of context data. In HCoM, we use a hybrid approach where ontology schema, context data and rules are stored and processed separately before they are combined and presented for reasoning purpose. Only relevant data is selected and loaded from the RCDB repository into the context-ontology schema. We have shown the theoretical aspect of how the selection/pruning algorithm is important to determine the quality and response time of the reasoning process.

We have also presented CoCA, a data independent context-aware service middleware that is based on the EHRAM and the HCoM models. CoCA is a collaborative platform that supports multiple-domains of applications for the development of context-aware applications in a pervasive environment. It has the module called RAID action engine that performs reasoning, interpretation, aggregation and decisions. CoCA has interfacing feature with neighboring peers to communicate and interchange context information on the fly. It uses Jxta protocol based peer-to-peer collaboration to ensure neighborhood communication between the pervasive devices of any capacity.

A demonstration version of CoCA is implemented based on the EHRAM and HCoM models. Validation of the models and the platform is made using a demonstration example on the smart campus scenario, PiCASO. We have also tested CoCA with data from a hospital scenario on patient monitoring and follow up service, and data from application adaptation on various device-properties like screen size, memory size, display capability, processor speed, and connection speed.

Results from our experiment show that EHRAM/HCoM model are scalable and extensible context management models, and CoCA is a generic, collaborative and data independent middleware.

7.3 Future works

In recent years, there has been considerable research into the development of pervasive context-aware applications. The trend shows that context-awareness is becoming natural part of the future of computing. With an increasing diversity of computer systems integrated in our surroundings and increasing mobility of both users and hardware this will be one of the main computing challenges in the years to come. The followings are some of the envisaged future works that can be considered as a continuation to this work.

Handling uncertainty: Human reasoning power is based on plenty of uncertainties in the environment. In most cases, sensor error (inherent granularity and/or false readings), out of date data and poor predictions will give rise to some uncertainty about sensed context. We need some means of handling this uncertainty problem before using the context data. A work on enhancing EHRAM and HCoM model and incorporate uncertainty factor to improve the quality of reasoning and decision support is an open research area. The use of metadata component of the EHRAM model can be a starting point towards achieving a more robust probabilistic context management model.

Security and privacy: Relationship between context data and security is bidirectional. On one hand, context data by itself needs security measures from capturing up to utilization. On the other hand, context data can be used to provide secured computing environment. According to [Hong04], due to the inherent need of collaboration, pervasive context-aware systems face security challenges in the form of privacy, integrity and trust. Privacy of context information focuses on protecting context resources from unauthorized entities. For example, a user should be able to protect personal information such as his/her health status, or medical history. Integrity of context information focuses on guaranteeing that the provided context information has not been corrupted by a third party. Trust is a respect for common security policy and goal.

There are very recent works that deal with limited aspects of this problem [Saad07] and [Bouna07]. Saad et al deals with trust based authentication and access control. The principle that, “friends of my friends are also my friends”. Input of context data and its role is very limited. In Bouma et al, the authors have created a proprietary context model based on limited number of tuples suited for multimedia context representation called MCC (Multi media Context Condition). They have investigated certainty issues related to multimedia based context data.

However, we need to have generic services that handle security problems in a heterogeneous pervasive environment without a restriction of the type of data or domain of application. Incorporating such security management component into the CoCA platform is another open problem area considered as a continuation of this work.

Other remaining challenges include the development of more intelligent proactive services that can be achieved by enhancing the platform through rule-mining using data from knowledge repository of the decision engine. The rule-mining module needs to be enhanced in line with the state of the art datamining tools in order to provide a new set of useful rules that are then converted to knowledge for decision and action. Therefore, the question of rule-mining in a pervasive context-aware computing environment is one of the central challenges in order to provide autonomic, proactive and intelligent decision and action trigger supports and services to the user.

The collaborative aspect of CoCA platform naturally exposes the system to faulty and malicious peers. One research area is therefore to incorporate fault tolerating and self-healing mechanisms. The work may involve identification of malicious peers and tagging them for consideration in their future participation, preparation of multiple alternatives so that failure of one particular communication will not block the system from functioning, etc.

GLOSSARY OF ACRONYMS

ACAI	Agent-based Context-aware Infrastructure
AP	WiFi Access Point (WiFi hot spot)
CAMidO	Context-Aware Middleware Based on Ontology Meta-Model
CC/PP	Composite Capabilities/Preference Profile *Description of device capabilities and user preferences.*
CCML	Centaurus Capability Markup Language
CDF	Context Description Framework
CFNs	Context Fusion Networks
CMF	Context Mediated Framework: *Application. programming interface for managing context information.*
CoBrA	Context Broker Architecture
CoCA	Collaborative Context-Aware service platform
Confab	Context Fabric *Architecture for privacy-sensitive systems.*
CONON	CONtext Ontology : *OWL based context Ontology for reasoning and representation of contexts in pervasive environments.*
CSCP	Comprehensive Structured Context Profiles
EHRAM	Entity, Hierarchy, Relation, Axiom and Metadata based context representation model
GAS	Gadget-ware Architectural Style ontology
GCoM	Ontology based Generic Context Management model
HCoM	Hybrid Context Management model
HTML	Hyper Text Mark-up Language
Jena	Java API based framework for building Semantic Web applications *It allows users to read, write, and manipulate RDF(S) and OWL models.*
JXTA	Set of open, generalized peer-to-peer protocols. *It allows any connected device (cell phone to PDA, PC to server) on the network to communicate and collaborate. JXTA protocols standardize the manner in which peers self-organize into peer groups, discover each other, advertise network services, communicate with each other, and monitor each other.*

MobiLife	A Project that aims at developing and validating a new generation of mobile applications and services for everyday users *The project emphasizes development of multi-modal interfaces, context awareness functionalities with privacy and trust support for the emerging 3G/WLan landscape and beyond.*
MySQL	Multithreaded, Multi-user SQL database management system
OWL	Web Ontology Language
PACE	Pervasive Autonomic Context-aware Environments project
PDA	Personal Data Assistant
PerSE	Pervasive Service Environment *A middleware that supports the interaction of independent and collaborating services to perform an intended action.*
PiCASO	Pervasive Campus-Aware Smart Onlooker *A use case scenario implemented to demonstrate the CoCA platform.*
PMML	Predictive Model Mark-up Language
Protégé	An ontology editor that provides tools to construct domain models and knowledge-based applications with ontology *It has a graphical user interface, with separate tabs for displaying ontology classes, properties and instances. Classes and properties are organized in to tree structures.*
RCDB	Relational Context Database
RCSM	Reconfigurable Context-Sensitive Middleware
RDF	Resource Description Framework
RDFS	Schema for RDF
SGML	Standard Generalized Markup Language
SOCAM	Service-oriented Context-Aware Middleware
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SPARQL	A query language and data access protocol for the Semantic Web *It is defined in terms of the W3C's RDF data model and will work for any data source that can be mapped into RDF.*
W3C	World Wide Web Consortium *Develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. It is a forum for information, commerce, communication, and collective understanding.*
XML	eXtensible Markup Language

BIBLIOGRAPHY

[Abowd97] Abowd G. D., Atkeson C. G., Hong J., et al. Cyberguide: a mobile context-aware tour guide. Journal of Wireless Networks: special issue on mobile computing and networking, 1997, Vol. 3(5), pp. 421-433.

[Ackoff89] Ackoff R. L. From Data to Wisdom. Journal of Applies Systems Analysis, 1989, Vol. 16, pp. 3-9.

[Akman96] Akman V., Surav M. Steps toward formalizing context. AI Magazine, 1996, Vol. 17(3), pp. 55-72.

[Amanuddin04] Amanuddin R., Ronchi D., Nguyen J., et al. Service-Oriented Architecture in a Pervasive Environment. IBM, 22 Sep 2004.

[Andrew99] Andrew C. H., Benjamin C. L., Shankar P. et al. Pervasive Computing: What is it good for? In: Proceedings of the Workshop on Mobile Data Management (MobiDE) in conjunction with ACM MobiCom '99,September 1999, Seattle, USA.

[Asthana94] Asthana A., Cravatts M., Krzyzanouski P. An indoor wireless system for personalized shopping assistance. In: Workshop on Mobile Computing Systems and Applications, December 1994, Santa Cruz, CA, USA, IEEE Computer Society Press, pp. 69-74.

[Baldauf07] Baldauf M., Dustdar S., Rosenberg F. A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2007, Vol. 2(4), pp. 263-277.

[Bardram04] Bardram J. The Java Context-Awareness Framework (JCAF) - A service infrastructure and programming framework for context-aware applications. In:Pervasive 2004, 2004, pp. 98-115.

[Behlouli06] Behlouli B. N., Taconet C., Bernard G. An architecture for Supporting Development and Execution of Context-Aware Componentapplications. In: ICPS'06 : IEEE International Conference on PervasiveServices 2006, June 2006, Lyon, France, pp. 57–66.

[Bellinger04] Bellinger G., Castro D., Mills A. Data, Information, Knowledge, and Wisdom. Published 2004, http://www.systems-thinking.org/dikw/ (last checked in May 2007).

[Berhe05] Berhe G., Brunie L., Pierson J. Distributed Content Adaptation for Pervasive Systems. In: Proceedings of the international Conference on information Technology: Coding and Computing (Itcc'05), April 2005, Washington DC, pp. 234-241 (DOI= http://dx.doi.org/10.1109/ITCC.2005.133).

[Bihler06] Bihler P., Scuturici M., Brunie L. Expressing and Interpreting User Intention in Pervasive Service Environments. Journal of Digital Information Management, 2006, Vol. 4(2), pp.102-106.

[Black04] Black J. P., Segmuller W., Cohen N., et al. Pervasive Computing in Health Care: Smart Spaces and Enterprise Information Systems. In: MobiSys Workshop on Context Awareness, June 2004, Boston Massachusetts, USA.

[Bolliger98] Bolliger J., Gross T. A Framework-Based Approach to the Development of Network-Aware Applications. IEEE Transactions on Software Engineering, MAY 1998, Vol. 24(5), pp. 376-390.

[Bouna07] Bouna B. A., Chbeir R., Miteran J. MCA2CM: Multimedia Context Aware Access Control Model. In: IEEE International Conference on Intelligence and Security Informatics (ISI'07), May 2007, New Jersey, USA, pp. 115-123.

[Bouzeghoub04] Bouzeghoub A., Defude B., Ammour S., et al. A RDF Description Model for Manipulating Learning Objects. In: Fourth IEEE International Conference on Advanced Learning Technologies (ICALT'04), 2004, pp. 81-85.

[Britannica07] Markup Language. Encyclopedia Britannica, Online http://www.britannica.com/ebc/article-9371358, last checked in May 2007.

[Broadbent97] Broadbent J., Marti P. Location-Aware Mobile Interactive Guides: Usability Issues. In: Proc. Inter. Cultural Heritage Informatics Meeting, 1997, Paris, France.

[Brumitt00] Brumitt B., Meyers B., Krumm J., et al. EasyLiving: Technologies for Intelligent Environments, Handheld and Ubiquitous Computing, Bristol , Royaume Uni, September 2000.

[Buchholz04] Buchholz S., Hamann T., Hubsch G. Comprehensive structured context profiles (CSCP): design and experience. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, March 2004. , pp. 43- 47.

[Burke05] Burke E. K., Kendall G. (Eds): Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer, 2005, New York.

[Burkhardt02] Burkhardt J., Henn H., Hepper S. et al: Pervasive Computing: Technology and Architecture of Mobile Internet Applications. Addison Wesley, 2002, London.

[Capra03] Capra L., Emmerich W., Mascolo, C. CARISMA: Context-Aware Reflective Middleware System for Mobile Applications. IEEE Transactions on Software Engineering, 2003, vol. 29(10), 17 p.

[CC/PP04] Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies. W3C Recommendation, 15 January 2004, Online http://www.w3.org/TR/CCPP-struct-vocab/. (last checked July 2007)

[Cerqueira01] Cerqueira R., Hess C. K., Roman M., et al. Gaia: A Development Infrastructure for Active Spaces. In: Workshop on Application Models and Programming Tools for Ubiquitous Computing with UBICOMP'01, Sept 2001, Atlanta Georgia, USA.

[Chaari06] Chaari T. , Ejigu D., Laforest F. et al. Modeling and Using Context in Adapting Applications to Pervasive Environments. In: The Proceedings of the IEEE International Conference on Pervasive Services (ICPS'06), 2006, Lyon, France, pp. 111-120.

[Chaari07] Chaari T., Ejigu D., Laforest F. et al. A Comprehensive Approach to Model and Use Context for Adapting Applications in Pervasive Environments. International Journal of Systems and Software (JSS), 2007, Vol. 80(12), pp. 1973-1992.

[Chen03a] Chen H., Finin T., Joshi A. An ontology for context-aware pervasive computing environments. Knowledge Engineering Review, 2003 Vol. 18, Special Issue on Ontologies for Distributed Systems, pp. 197-207.

[Chen03b] Chen H. An Intelligent Broker Architecture for Context-Aware Systems. A Ph. D. Thesis, University of Maryland Baltimore County, January 2003.

[Chen04a] Chen G., Li M., Kotz D. Design and implementation of a large-scale context fusion network. In: 1st Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous), 2004, Boston, Massachussets, IEEE Computer Society, 246-255.

[Chen04b] Chen H., Perich, F., Chakraborty, D. et al Intelligent Agents Meet Semantic Web in a Smart Meeting Room. In: The Third International Joint Conference on Autonomous Agents and Mutli-Agent Systems, 2004.

[Chen04c] Chen H., Finin T., Joshi A. et al. Intelligent Agents Meet the Semantic Web in Smart Spaces. IEEE Internet Computing, November 2004, Vol. 8(6), pp. 69-79.

[Chen04d] Chen H., Finin, T., Joshi A. Semantic Web in the Context Broker Architecture. In: Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications, PerCom'04, 2004, Washington DC, USA.

[Chen04e] Chen H., Perich F., Finin T. et al. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In: International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004, Boston, USA.

[Chen04f] Chen H., An Intelligent Broker Architecture for Pervasive Context-Aware Systems. PhD Thesis. University of Maryland, December 2004, Baltimore County, USA.

[Cheverst00] Cheverst K., Davies N., Mitchell K. et al. Developing a Context-Aware Electronic Tourist Guide: some issues and experiences. In: The proceedings of CHI'00, March 2000, the Hague, The Netherlands.

[Christopoulou04] Christopoulou E., Kameas A. GAS Ontology: an ontology for collaboration among ubiquitous computing devices. International Journal of Human-Computer Studies, Mai 2005, Vol. 62(5), pp. 664-685.

[Daby01] Daby M. S., Guruduth B., John S. D. et al. Preparing the Edge of the Network for Pervasive Content Delivery. In: Workshop on Middleware for Mobile Computing, November 16, 2001.

[Davis99] Davies N., Cheverst K., Mitchell K., et al. Caches in the Air: Disseminating Tourist Information in the Guide System. In: Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA'99), New Orleans, LA, USA, 1999.

[Dertouzos99] Dertouzos M. The Oxygen Project. Scientific American, Aug 1999, Vol. 281(2), pp. 52-63.

[Dey00] Dey A. K., Abowd G. D. Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the CHI Workshop on the What, Who, Where, and How of Context-Awareness, April 2000, The Hague, The Netherlands.

[Dey01] Dey A. K., Salber D., Abowd, G. D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 2001, Vol 16, pp. 97-166.

[Ejigu07a] Ejigu D., Scuturici M., Brunie L. An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing. In: Fifth IEEE International Conference on Pervasive Computing and Communications, PerComW'07, 2007, New York, USA, pp. 14-19.

[Ejigu07b] Ejigu D., Scuturici, M., Brunie L. CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing. In: Fourth IEEE International Conference on Information Technology - New Generations, ITNG'07, April 2007, Las Vegas, USA, pp. 297-302.

[Ejigu07c] Ejigu D., Scuturici M., Brunie L. Semantic Approach to Context Management and Reasoning in Ubiquitous Context-Aware Systems. In: the proceedings of the Second IEEE International Conference on Digital Information Management (ICDIM'07), October 2007, Lyon, France, pp. 500-5005.

[Ejigu08] Ejigu D., Scuturici M., Brunie L. Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. E Dejene, V. Scuturici, L. Brunie. Journal of Computers (JCP) 3(1):40-50, Academy Publisher, ISSN 1796-203X. 2008.

[Eriksson04] Hans-Erik E., Magnus P., Brian L. et al. UML 2 Toolkit, Wiley publishers, October 2004, USA.

[Espinoza01] Espinoza F., Persson P., Sandin A., et al. GeoNotes: social and navigational aspects of location-based information systems. In: the Proceedings of International Conf. On UbiComp, Sept 2001, Atlanta, USA, pp. 2-17.

[Feiner02] Feiner, S. Augmented Reality: A New Way of Seeing. Scientific American, April 2002.

[FreeNet06] Project FreeNet, online http://freenet.sourceforge.net/, last checked in December 2006.

[Garlan02] Garlan D., Siewiorek, D., Smailagic, A., et al. Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, April 2002, pp. 22-31.

[Gellersen00] Gellersen W., Schmidt H. A., Beigl M. Adding some smartness to devices and everyday things. In: the Proceedings of the third IEEE Workshop on Mobile Computing Systems and Applications, Monterey, California, 2000.

[Gnutella06] Project Gnutella, online http://www.gnutella2.com/, last checked in December 2006.

[GPS07] Global Positioning System, online http://www.gps.gov/, last checked in August 2007.

[Gripay06] Gripay, Y., Pierson J., Pigeot C.E. Une architecture pervasive sécurisée : PerSE, Dans UbiMob'06, ACM ed., 2006, Paris. pp. 147-150.

[Groff02] Groff J.R., Weinberg P. N. SQL: The Complete Reference, McGraw-Hill, 2nd edition, August 2002.

[Gruber93] Gruber T. R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, November 1993, Vol. 43, Issues 4-5, pp. 907-928.

[Gu05] Gu T., Pung H. K., Zhang D. Q. A service-oriented middleware for building context-aware services. Journal of Network Computing, Jan 2005, pp. 1-18.

[Halpin07] Halpin T. Object Role Modeling (ORM), online http://www.orm.net/, last checked in July 2007.

[Hand01] Hand D., Mannila H., Smyth P. Principles of Data Mining. MIT Press, 2001, Cambridge, USA.

[Hariri06] Hariri S., Khargharia B., Chen H. et al. The Autonomic Computing Paradigm, Cluster Computing, 2006, Vol. 9(1), pp. 5-17.

[Held02] Held A., Buchholz S., Schill A. Modeling of Context Information for Pervasive Computing Applications. In: the Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), July 2002, Orlando, USA.

[Henricksen02] Henricksen K., Indulska J., Rakotonirainy A. Modeling Context Information in Pervasive Computing Systems. In: Proceedings of Pervasive'02, 2002, Zurich.

[Henricksen04a] Henricksen K., Indulska J. Modelling and using imperfect context information. In: CoMoRea at 2nd IEEE Conference on Pervasive Computing and Communications (PerCom'04), March 2004, Orlando, USA, pages 33- 37.

[Henricksen04b] Henricksen K., Indulska J. A software engineering framework for context-aware pervasive computing. In: 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04), IEEE Computer Society, 2004, pp 77-86.

[Henricksen05a] Henricksen K., Indulska J. Developing context-aware pervasive computing applications: Models and approach, Journal of Pervasive and Mobile Computing, 2005, Elsevier.

[Henricksen05b] Henricksen K., Indulska J., McFadden T., et al. Middleware for distributed context-aware systems. International Symposium on Distributed Objects and Applications (DOA), Agia Napa, Cyprus, 2005.

[Hong01] Hong J.I., Landay J. A. An infrastructure approach to context-aware computing. HumanComputer Interaction, 2001, 16(2).

[Hong04] Hong J.I., Landay J.A. An architecture for privacy-sensitive ubiquitous com puting. In: 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys), Boston, 2004.

[IBM01] Autonomic Computing: IBM's Perspective on the State of Information Technology, online http://www.research.ibm.com/autonomic/manifesto, published 2001, last checked September 2007.

[Indulska03] Indulska J., Robinsona R., Rakotonirainy A. Experiences in using cc/pp in context-aware systems. In: Proceedings of the 4th International Conference on Mobile Data Management (MDM'03), Lecture Notes in Computer Science (LNCS), Springer, January 2003, Melbourne, Australia, pp. 247–261.

[Intel07] Moore's Law, Intel® Research, online http://www.intel.com/technology/mooreslaw/index.htm, last checked in June 2007.

[Jacobson99] Jacobson I., Booch G., Rumbaugh J. The unified software development process, Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1999.

[Jena07] Jena from SourceForge.Net, A Semantic Web Framework for Java, online http://jena.sourceforge.net/, last checked in February 2007.

[JXME07] JXME, JXTA Java Micro Edition Project, online https://jxta-jxme.dev.java.net/, last checked June 2007.

[JXTA07] Project JXTA, online http://www.jxta.org/, last checked in February 2007.

[Kagal02] Kagal L. , Korolev V. , Avancha S. et al. Centaurus: an infrastructure for service management in ubiquitous computing environments. Wireless Networks, November 2002, Vol. 8(6), p.619-635.

[Kheder05] Khedr M., Karmouch A. ACAI: agent-based context-aware infrastructure for spontaneous applications. Journal of Network Computing Applications, 2005, Vol. 28(1), pp. 19-44.

[Khriyenko05] Khriyenko05] Khriyenko O., Terziyan V. Context Description Framework for the Semantic Web. In: Context Representation and Reasoning Workshop, Paris, France, Jul 2005.

[Kindberg00] Kindberg T, Barton J, Morgan J. et al. People, Places, Things: Web Presence for the Real World. Proc. 3rd IEEE Workshop on Mobile Computing Systems and Applications, 2000, California, USA, pp. 19-28.

[Kindberg01] Kindberg T., Barton J. A Web-based nomadic computing system, Computer Networks, 2001, Vol 35(4), pp. 443-456.

[Korkea00] Korkea A. M. Context-Aware Applications Survey. Department of Computer Science, Helsinki University of Technology, 2000.

[Korpipaa03] Korpipaa P., Mantyjarvi J., Kela J., et al. Managing Context Information in Mobile Devices. Pervasive Computing, July-Sept. 2003, Vol. 2(3), pp. 42- 51.

[Lai02] Lai J., Levas A., Chou P. et al. BlueSpace: personalizing workspace through awareness and daptability. International Journal of Human-Computer Study, November 2002, Vol. 57(5), pp. 415-428.

[Lee03] Lee C. H., Na J. C., Khoo, C. Ontology Learning for Medical Digital Libraries. In: the Proceedings of ICADL'03 (International Conference on Asian Digital Libraries), December 2003, Malaysia, Kuala Lumpur, pp. 302-305, 2003.

[Lins04] Lins E. P., Shultz U.P. Compatibility vs. Evolution in Pervasive Computing, OOPSLA '04 Workshop Workshop on Building Software for Pervasive Computing, Vancouver, Canada, August 20, 2004.

[Luger05] Luger G. F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Fifth Edition, Addison-Wesley, Harlow, England, 2005.

[Maibaum02] Maibaum N., Mundt T. JXTA: A Technology Facilitating Mobile Peer-To-Peer Networks. In: International Mobility and Wireless Access Workshop (MobiWac'02), Fort Worth, TX, 2002.

[Marco06] Marco D. A Meta data repository is a key to knowledge management. The data administration newsletter, online URL:http://www.tdan.com/view-articles/5064, last checked July 2006.

[Mattern03] Mattern F., Sturn P. From Distributed Systems to Ubiquitous Computing - State of the Art. Trends and Prospects of Future Networked systemspp. 3-25. Fachtagung "Kommunikation in Verteilten Systemen" (KiVS), Leipzig, , 2003. Springer-Verlag, Berlin, 2003.

[McCarthy98] McCarthy S. B. Formalizing context (expanded notes)on Computing Natural Language. In: Aliseda, Atocha, Van Glabbeek, Rob J., Westerståhl, Dag, Computing natural language. Stanford, Californie : Center for the Study of Language and Information (CSLI Publications), , 1998, pp. 13-50. (CSLI Lecture Notes, Vol. 81)

[McFadden04] McFadden T., Henricksen K., Indulska J. Automating context-aware application development. In: UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, 2004, Nottingham, pp. 90-95.

[McFadden05] McFadden T., Henricksen K., Indulska J. et al. Applying a disciplined approach to the development of a context-aware communication application. In: 3rd IEEE Int. Conference on Pervasive Computing and Communications, PerCom05,2005, pp. 300-306.

[MCubiX07] MCubiX, online http://www.diagnos.ca/, last checked in February 2007.

[Meyer03] Meyer S., Rakotonirainy A. A Survey of Research on Context-Aware Homes. In: Conferences in Research and Practice in Information Technology Series, 2003, Vol. 34, Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, Vol. 21, Adelaide, Australia, pp. 159-168.

[MobiLife07] MobiLife Project, online http://www.ist-mobilife.org/, last checked in June 2007.

[Mrohs06] Mrohs B., Steglich S., Klemettinen M. MobiLife Service Infrastructure and SPICE Architecture Principles. In: Vehicular Technology Conference (VTC), September 2006, Montreal, Canada.

[MySQL07] Development with MySQL, online http://dev.mysql.com/, last checked in February 2007.

[Napster06] Project Napster, online http://opennap.sourceforge.net/#status/, last checked in December 2006.

[Nelson99] Nelson R. A.: The Global Positioning System. Online http://www.aticourses.com/global_positioning_system.htm, publishe November 1999, Last checked in July 2007.

[Nielson95] Nielson H. R., Nielson F. Semantics with Applications: A Formal Introduction (1st ed.), 1995, Chicester, England: John Wiley & Sons, ISBN 0-471-92980-8.

[ODBC07] ODBC and MySQL, online http://dev.mysql.com/downloads/connector/odbc/3.51.html, last checked in March 2007.

[OpenGALEN07] OpenGALEN Manifesto, jointly by the Victoria University of Manchester in the United Kingdom and the University of Nijmegen in The Netherlands, online http://www.opengalen.org/index.html, last checked in October 2007.

[Ouksel03] Ouksel A. M. In-Context Information Filtering On the Web: An Emergent Semantics P2P Approach. SIGMOD Record, September 2003, Vol 32(3), pp. 65-70.

[OWL07] McGuinness D. L., Van-Harmelen F. OWL Web Ontology Language Overview, online at http://www.w3.org/TR/owl-features/, last checked in February 2007.

[Paritosh06] Paritosh, P.K. The Heuristic Reasoning Manifesto. In: the Proceedings of the 20th International Workshop on Qualitative Reasoning, Hanover, 2006.

[Pascoe97] Pascoe J. The Stick-e Note Architecture: Extending the Interface Beyond the User. In: Proceedings of the International Conference on Intelligent User Interfaces, Orlando Florida USA, 1997, pp. 261-264.

[Pigeot07] Pigeot C.E, Gripay Y., Scuturici M., et al. Context-Sensitive Security Framework for Pervasive Environments, In: Fourth European Conference on Universal Multiservice Networks (ECUMN'07), Toulouse, 2007, pp. 391-400.

[PMML05] Predictive Model Markup Language (PMML): Data Mining Group, online http://www.dmg.org/, last checked 23/12/2005.

[Priyantha01] Priyantha N., Miu A., Balakrishnan H., Teller S. The Cricket Compass for Context-Aware Applications, Proc. of 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001), Rome, Italy, July 2001.

[Protégé07] Protégé from stanford.edu, Ontology editor and knowledge-base frame, online http://protege.stanford.edu/, last checked in February 2007.

[Ranganathan04] Ranganathan A., Al-Muhtadi J., Campbell R. H. Reasoning about Uncertain Contexts in Pervasive Computing Environments. IEEE CS and IEEE ComSoc, 2004, University of Illinois, USA.

[Rarau05] Rarau A., Pusztai K., Salomie L. MultiFacet Item Based Context-Aware Applications, International Journal of Computing & Information Sciences, 2005, Vol. 3(2), pp.10-18.

[RDF06] RDF W3C Recommendations, online http://www.w3.org/2001/sw/RDFCore/#documents, last checked in December 2006.

[Rekimoto95] Rekimoto J., Nagao K. The World through the Computer: Computer Augmented Interaction with Real World Environments. In: Proceedings of the 8th ACM

Symposium of User Interface Software and Technology, Pittsbugh, PA, November 1995, 29-38.

[Roman02] Roman M., Hess C., Cerqueira R., et al. Gaia: A middleware infrastructure for active spaces. In IEEE Pervasive Computing, Special Issue on Wearable Computing,2002, 74-83.

[Roussos05] Roussos Y., Stavrakas Y., Pavlaki V. Towards a Context-Aware Relatioal Model. In: Proceedings of the CRR'05 Workshop, 2005, Paris, France, pp. 5-8.

[Rudolph01] Rudolph L. Project Oxygen: Pervasive, Human-Centric Computing - An Initial Experience. In: Proceedings of the 13th international Conference on Advanced information Systems Engineering, Interlaken, Switzerland, June 4-8, 2001.

[Saadi07] Saadi R., Pierson J., Brunie L.Authentication and Access Control Using Trust Collaboration in Pervasive Grid Environment.In: Grid and Pervasive Compuing (GPC'07), Lecture Notes in Computer Science (LNCS), Springer Verlag, 2007, Paris, France. pp. 348-361.

[Sadeh02] Sadeh N., Chan E., Shimazaki Y. et al. Mycampus: An agent-based environment for context-aware mobile services. In: the proceedings of the AAMAS02 Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, July 2002, Bologna, Italy.

[Salber99] Salber D., Day A. K., Abowd G. D. The context toolkit: Aiding the development of context-enabled applications. In: Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI'99), Pittsburg, PA, May 1999, pp. 434-441.

[Satyanarayanan01] Satyanarayanan M. Pervasive Computing: Vision and Challenges. IEEE Personal Communications, August 200, pp. 10-17.

[Schilit93] Schilit B., Adams N., Gold, R. et al. The PARCTAB mobile computing system. In: Proceedings of the Fourth IEEE Workshop on Workstation Operating Systems (WWOS-IV), Napa, CA, October 1993, pages 34-39.

[Schilit94] Schilit B.N., Adams N., Want R. Context-Aware Computing Applications. In: the Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, 1994, p. 85-90.

[Scuturici06] Scuturici M., Ejigu D. Positioning Support in Pervasive Environments. In: IEEE International Conference on Pervasive Services, ICPS'06, 2006, pp. 19-26, Lyon, France.

[Sharmin06] Sharmin M., Ahmed S., Ahamed, S. I. MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments. In: Proceedings of the Third international Conference on information Technology: New Generations (ITNG'06), IEEE Computer Society, April 2006, Washington DC, USA, pp. 306-313.

[Silva94] Silva J. P. M., Sakallah K. A. Dynamic Search-Space Pruning Techniques in Path Sensitization. In: Proceedings Of IEEE/ACM Design Automation Conference (DAC), June 1994, San Diego, California, pp. 705-711.

[Singh06] Singh A., Conway M. Survey of Context aware Frameworks: Analysis and Criticism. UNC Information Technology Services, The University Of North Carolina, Chapel Hill, 2006.

[Skiena90] Skiena S. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Addison-Wesley, 1990, Reading, MA.

[SPARQL07] SPARQL Query Language for RDF, Edited by Prud'hommeaux E.and Seaborne A., W3C Candidate Recommendation, online http://www.w3.org/TR/rdf-sparql-query/, last checked 14 June 2007.

[Staab00] Staab S., Erdmann M., Maedche A. et al. An Extensible Approach for Modeling Ontologies in RDF(S)", In: the Proceedings of ECDL 2000 Workshop on the Semantic Web, 2000, Lisbon, Portugal.

[Strang04] Strang T., Linnhoff-Popien C. A Context Modeling Survey. In: the Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management, Sixth International Conference on UbiComp'04. 2004, Nottingham, England.

[TechTarget07] Pervasive Computing, online http://searchnetworking.techtarget.com/ sDefinition/, last checked in July 2007.

[Wang04] Wang X., Zhang D. Q., Gu T., Pung H. K.: Ontology Based Context Modeling and Reasoning using OWL, workshop on context modeling and reasoning. In: IEEE International Conference on Pervasive Computing and Communication , March 2004, Orlando, Florida.

[Want92] Want R., Hopper A. Falcao The Active Badge Location System. In: ACM Transactions on Information Systems, 1992, Vol. 10(1). pp. 91-102.

[Weiser91] Weiser M. The Computer for the Twenty-First Century, Scientific American, Sept. 1991, Vol.265(3), pp. 94-104.

[WiFi07] WiFi, online http://compnetworking.about.com/cs/wireless80211/a/ aa80211standard.htm, last checked in July 2007.

[Winograd01] Winograd T. Architectures for Context. Human-Computer Interaction, 2001, Vol. 16(2,3,4), pp. 401-419.

[Woodruff01] Woodruff A., Aoki P.M., Hurst A. et al. Electronic Guidebooks and Visitor Attention. In: Proc. 6th Int. Cultural Heritage Informatics Meeting, Sep. 2001, Milan, Italy, pp. 437-454.

[Yau02] Yau S. S., Karim F., Wang Y.: Reconfigurable Context-Sensitive Middleware for Pervasive Computing, IEEE Pervasive Computing, July-September 2002, Vol 1(3), pp.33-40.

[Zuo06] Zuo M., Haarslev V. High Performance Absorption Algorithms for Terminological Reasoning. In: Proceedings of the International Workshop on Description Logics (DL), UK, May 2006, Lake District, pp. 159-166.

ANNEXES

I. OWL vocabularies for semantic reasoning

OWL features related to RDF schema

- Class: A class defines a group of individuals that belong together because they share some properties. For example, Deborah and Frank are both members of the class Person. Classes can be organized in a specialization hierarchy using subClassOf. There is a built-in most general class named Thing that is the class of all individuals and is a super class of all OWL classes. There is also a built-in most specific class named Nothing that is the class that has no instances and a subclass of all OWL classes.
- rdfs:subClassOf: Class hierarchies may be created by making one or more statements that a class is a subclass of another class. For example, the class Person could be stated to be a subclass of the class Mammal. From this a reasoner can deduce that if an individual is a Person, then it is also a Mammal.
- rdf:Property: Properties can be used to state relationships between individuals or from individuals to data values. Examples of properties include hasChild, hasRelative, hasSibling, and hasAge. The first three can be used to relate an instance of a class Person to another instance of the class Person (and are thus occurrences of ObjectProperty), and the last (hasAge) can be used to relate an instance of the class Person to an instance of the datatype Integer (and is thus an occurrence of DatatypeProperty). Both owl:ObjectProperty and owl:DatatypeProperty are subclasses of the RDF class rdf:Property.
- rdfs:subPropertyOf: Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties. For example, hasSibling may be stated to be a subproperty of hasRelative. From this a reasoner can deduce that if an individual is related to another by the hasSibling property, then it is also related to the other by the hasRelative property.
- rdfs:domain: A domain of a property limits the individuals to which the property can be applied. If a property relates an individual to another individual, and the property has a class as one of its domains, then the individual must belong to the class. For example, the property hasChild may be stated to have the domain of Mammal. From this a reasoner can deduce that if Frank hasChild Anna, then Frank must be a Mammal. Note that rdfs:domain is called a global restriction since the restriction is stated on the property and not just on the property when it is associated with a particular class. See the discussion below on property restrictions for more information.
- rdfs:range: The range of a property limits the individuals that the property may have as its value. If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. For example, the property hasChild may be stated to have the range of Mammal. From this a reasoner can deduce that if Louise is related to Deborah by the hasChild property, (i.e., Deborah

is the child of Louise), then Deborah is a Mammal. Range is also a global restriction as is domain above. Again, see the discussion below on local restrictions (e.g. AllValuesFrom) for more information.

- Individual : Individuals are instances of classes, and properties may be used to relate one individual to another. For example, an individual named Deborah may be described as an instance of the class Person and the property hasEmployer may be used to relate the individual Deborah to the individual StanfordUniversity.

OWL equality and inequality

- equivalentClass : Two classes may be stated to be equivalent. Equivalent classes have the same instances. Equality can be used to create synonymous classes. For example, Car can be stated to be equivalentClass to Automobile. From this a reasoner can deduce that any individual that is an instance of Car is also an instance of Automobile and vice versa.
- equivalentProperty: Two properties may be stated to be equivalent. Equivalent properties relate one individual to the same set of other individuals. Equality may be used to create synonymous properties. For example, hasLeader may be stated to be the equivalentProperty to hasHead. From this a reasoner can deduce that if X is related to Y by the property hasLeader, X is also related to Y by the property hasHead and vice versa. A reasoner can also deduce that hasLeader is a subproperty of hasHead and hasHead is a subProperty of hasLeader.
- sameAs: Two individuals may be stated to be the same. Can be used to create a number of different names that refer to the same individual. For example, the individual Deborah may be stated to be the same individual as DeborahMcGuinness.
- differentFrom: An individual may be stated to be different from other individuals. For example, the individual Frank may be stated to be different from the individuals Deborah and Jim. Thus, if the individuals Frank and Deborah are both values for a property that is stated to be functional (thus the property has at most one value), then there is a contradiction. Explicitly stating that individuals are different can be important in when using languages such as OWL (and RDF) that do not assume that individuals have one and only one name. For example, with no additional information, a reasoner will not deduce that Frank and Deborah refer to distinct individuals.
- AllDifferent: A number of individuals may be stated to be mutually distinct in one AllDifferent statement. For example, Frank, Deborah, and Jim could be stated to be mutually distinct using the AllDifferent construct. Unlike the differentFrom statement above, this would also enforce that Jim and Deborah are distinct (not just that Frank is distinct from Deborah and Frank is distinct from Jim). The AllDifferent construct is particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects. It is

used in conjunction with distinctMembers to state that all members of a list are distinct and pairwise disjoint.

OWL properties

- inverseOf: One property may be stated to be the inverse of another property. If the property P1 is stated to be the inverse of the property P2, then if X is related to Y by the P2 property, then Y is related to X by the P1 property. For example, if hasChild is the inverse of hasParent and Deborah hasParent Louise, then a reasoner can deduce that Louise hasChild Deborah.
- TransitiveProperty: If a property is transitive, then if the pair (x,y) is an instance of the transitive property P, and the pair (y,z) is an instance of P, then the pair (x,z) is also an instance of P. For example, if ancestor is stated to be transitive, and if Sara is an ancestor of Louise (i.e., (Sara,Louise) is an instance of the property ancestor) and Louise is an ancestor of Deborah (i.e., (Louise,Deborah) is an instance of the property ancestor), then a reasoner can deduce that Sara is an ancestor of Deborah (i.e., (Sara,Deborah) is an instance of the property ancestor).
- SymmetricProperty: Properties may be stated to be symmetric. If a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P. For example, friend may be stated to be a symmetric property. Then a reasoner that is given that Frank is a friend of Deborah can deduce that Deborah is a friend of Frank.
- FunctionalProperty : Properties may be stated to have a unique value. If a property is a FunctionalProperty, then it has no more than one value for each individual (it may have no values for an individual). This characteristic has been referred to as having a unique property. FunctionalProperty is shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1. For example, hasPrimaryEmployer may be stated to be a FunctionalProperty. From this a reasoner may deduce that no individual may have more than one primary employer. This does not imply that every Person must have at least one primary employer however.
- InverseFunctionalProperty: Properties may be stated to be inverse functional. If a property is inverse functional then the inverse of the property is functional. Thus the inverse of the property has at most one value for each individual. This characteristic has also been referred to as an unambiguous property. For example, hasUSSocialSecurityNumber (a unique identifier for United States residents) may be stated to be inverse functional (or unambiguous). The inverse of this property (which may be referred to as isTheSocialSecurityNumberFor) has at most one value for any individual in the class of social security numbers. Thus any one person's social security number is the only value for their isTheSocialSecurityNumberFor property. From this a reasoner can deduce that no two different individual instances of Person have the identical US Social Security Number. Also, a reasoner can deduce that if two instances

of Person have the same social security number, then those two instances refer to the same individual.

OWL restrictions

- owl:Restriction: OWL allows restrictions to be placed on how properties can be used by instances of a class. These type are used within the context of an owl:Restriction.
- owl:onProperty: The element indicates the restricted property. The following restrictions limit which values can be used while the cardinality restrictions limit how many values can be used.
- allValuesFrom: The restriction allValuesFrom is stated on a property with respect to a class. It means that this property on this particular class has a local range restriction associated with it. Thus if an instance of the class is related by the property to a second individual, then the second individual can be inferred to be an instance of the local range restriction class. For example, the class Person may have a property called hasDaughter restricted to have allValuesFrom the class Woman. This means that if an individual person Louise is related by the property hasDaughter to the individual Deborah, then from this a reasoner can deduce that Deborah is an instance of the class Woman. This restriction allows the property hasDaughter to be used with other classes, such as the class Cat, and have an appropriate value restriction associated with the use of the property on that class. In this case, hasDaughter would have the local range restriction of Cat when associated with the class Cat and would have the local range restriction Person when associated with the class Person. Note that a reasoner can not deduce from an allValuesFrom restriction alone that there actually is at least one value for the property.
- someValuesFrom: The restriction someValuesFrom is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type. For example, the class SemanticWebPaper may have a someValuesFrom restriction on the hasKeyword property that states that some value for the hasKeyword property should be an instance of the class SemanticWebTopic. This allows for the option of having multiple keywords and as long as one or more is an instance of the class SemanticWebTopic, then the paper would be consistent with the someValuesFrom restriction. Unlike allValuesFrom, someValuesFrom does not restrict all the values of the property to be instances of the same class. If myPaper is an instance of the SemanticWebPaper class, then myPaper is related by the hasKeyword property to at least one instance of the SemanticWebTopic class. Note that a reasoner can not deduce (as it could with allValuesFrom restrictions) that all values of hasKeyword are instances of the SemanticWebTopic class

OWL cardinalities

- minCardinality: Cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is stated on a property with respect to a class, then any instance of that class will be related to at least one individual by that property. This restriction is

another way of saying that the property is required to have a value for all instances of the class. For example, the class Person would not have any minimum cardinality restrictions stated on a hasOffspring property since not all persons have offspring. The class Parent however would have a minimum cardinality of 1 on the hasOffspring property. If a reasoner knows that Louise is a Person, then nothing can be deduced about a minimum cardinality for her hasOffspring property. Once it is discovered that Louise is an instance of Parent, then a reasoner can deduce that Louise is related to at least one individual by the hasOffspring property. From this information alone, a reasoner can not deduce any maximum number of offspring for individual instances of the class parent.

- maxCardinality: Cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is stated on a property with respect to a class, then any instance of that class will be related to at most one individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique property. For example, the property hasRegisteredVotingState on the class UnitedStatesCitizens may have a maximum cardinality of one (because people are only allowed to vote in only one state). From this a reasoner can deduce that individual instances of the class USCitizens may not be related to two or more distinct individuals through the hasRegisteredVotingState property. From a maximum cardinality one restriction alone, a reasoner can not deduce a minimum cardinality of 1. It may be useful to state that certain classes have no values for a particular property. For example, instances of the class UnmarriedPerson should not be related to any individuals by the property hasSpouse. This situation is represented by a maximum cardinality of zero on the hasSpouse property on the class UnmarriedPerson.

- cardinality: Cardinality is provided as a convenience when it is useful to state that a property on a class has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1. For example, the class Person has exactly one value for the property hasBirthMother. From this a reasoner can deduce that no two distinct individual instances of the class Mother may be values for the hasBirthMother property of the same person.

Other OWL vocabularies

- oneOf: (enumerated classes): Classes can be described by enumeration of the individuals that make up the class. The members of the class are exactly the set of enumerated individuals; no more, no less. For example, the class of daysOfTheWeek can be described by simply enumerating the individuals Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. From this a reasoner can deduce the maximum cardinality (7) of any property that has daysOfTheWeek as its allValuesFrom restriction.

- hasValue: (property values): A property can be required to have a certain individual as a value (also sometimes referred to as property values). For example, instances of the class of dutchCitizens can be characterized as those people that have theNetherlands as a

value of their nationality. (The nationality value, theNetherlands, is an instance of the class of Nationalities).

- disjointWith: Classes may be stated to be disjoint from each other. For example, Man and Woman can be stated to be disjoint classes. From this disjointWith statement, a reasoner can deduce an inconsistency when an individual is stated to be an instance of both and similarly a reasoner can deduce that if A is an instance of Man, then A is not an instance of Woman.
- unionOf, complementOf, intersectionOf (Boolean combinations): OWL allows arbitrary Boolean combinations of classes and restrictions: unionOf, complementOf, and intersectionOf. For example, using unionOf, we can state that a class contains things that are either USCitizens or DutchCitizens. Using complementOf, we could state that children are not SeniorCitizens. (i.e. the class Children is a subclass of the complement of SeniorCitizens). Citizenship of the European Union could be described as the union of the citizenship of all member states.
- minCardinality, maxCardinality, cardinality (full cardinality): OWL allows cardinality statements for arbitrary non-negative integers. For example the class of DINKs ("Dual Income, No Kids") would restrict the cardinality of the property hasIncome to a minimum cardinality of two (while the property hasChild would have to be restricted to cardinality 0).

II. Major CoCA implementation classes

Source URL: http://liris.cnrs.fr/~edejene/CoCASys/index.html

```
// Interface and data/event listeners
Class coca.hcom.CoCAframe
Methods
      public javax.swing.Box collabBox()
      public javax.swing.Box ontologyBox()
      public javax.swing.Box contextBox()
      public javax.swing.Box actionBox()
      public javax.swing.Box consoleBox()
      public java.lang.String trimResult(java.lang.String longResult)
      public static void toConsole(java.lang.String stt)

      //Nested classes for action listening/trigger
      class CoCAframe.DataListener
      class CoCAframe.ShowFilteredListener
      class CoCAframe.TakeActionListener
      class CoCAframe.ValidityListener
      class CoCAframe.WinCleanListener
      class CoCAframe.WinCleanListener2
      class CoCAframe.WinCleanListener3

//Create and initialize memory based reasoning models
Class coca.hcom.CoCAModelFactory
Methods
      public InfModel getInitialModel()
      public InfModel getCompleteModel()

//Query, aggregation and decision
Class coca.hcom.CoCAquery
Methods
      public java.lang.String[]
      public getFilteredTriples(java.lang.String subject,
            java.lang.String relation,java.lang.String value)
      public java.lang.String[] getAllInstances()
      public java.lang.String[] getAllRelations()
      public java.lang.String[] getActionsToGo()
      public java.lang.String trimResult(java.lang.String longResult)

// Context and ontology merging
Class coca.hcom.CoCArcdb
Methods
      public void initialContextToCDB()
      public void staticContextToRDF()

// Context filtering storage
Class coca.hcom.CoCAstore
Methods
      public void post(java.lang.String[] textString, int size)
      public void postStaticContext(java.lang.String[]
                                    textString,int size)
```

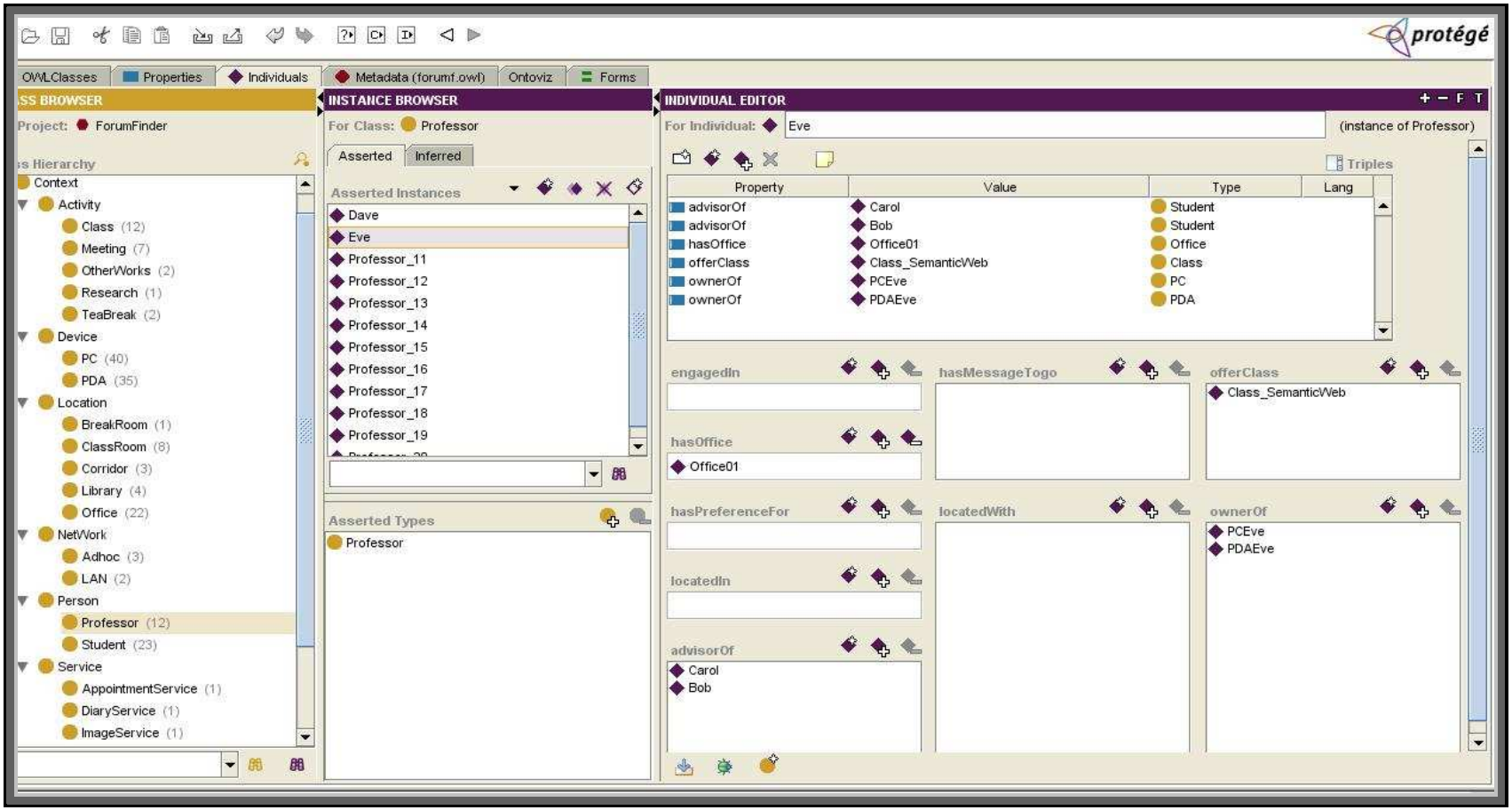
```
//CaCA congigure and initialize
Class coca.hcom.CoCAsystem
Methods
      public static void main(java.lang.String[] args)

// Context capturing, filter, …
Class coca.hcom.ContextHandler
Methods
      public static boolean validContext(java.lang.String subject,
                                    java.lang.String relation,
                                    java.lang.String value)
      public void addContextToModel(java.lang.String Sub,
                                    java.lang.String Rel,
                                    java.lang.String Val)
// Connect, get and or receive data
class cova.hcom.CoCAcollaboration
Methods
      public void startJxta()
      public void run()//wait for msgs
      public void sendAndReceiveData(JxtaSocket socket)
      public java.lang.String getContext(java.lang.String msg)
      public java.lang.String getSource(java.lang.String msg)
      protected static java.io.InputStream
      getResourceInputStream(java.lang.String resource)
      protected static boolean configured(java.io.File home)
      protected static void createConfig(java.io.File home,
                                         java.lang.String name,
                                         boolean server)
```

III. PiCASO demonstration sample ontology

Ontology Source URL: http://liris.cnrs.fr/~edejene/PiCASO/index.html

Sample protégé screenshot showing PiCASO ontology definitions



An excerpt of PiCASO OWL ontology from Protégé

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.cocasp.fr/forumf.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://www.cocasp.fr/forumf.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="VideoService">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Service"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Office">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Service">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Context"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Professor">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DiaryService">
    <rdfs:subClassOf rdf:resource="#Service"/>
  </owl:Class>
  <owl:Class rdf:ID="Meeting">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>
  <owl:Class rdf:ID="Research">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>
  <owl:Class rdf:ID="VoiceService">
    <rdfs:subClassOf rdf:resource="#Service"/>
  </owl:Class>
  <owl:Class rdf:ID="Corridor">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </owl:Class>
  <owl:Class rdf:about="#Device">
    <rdfs:subClassOf rdf:resource="#Context"/>
  </owl:Class>
  <owl:Class rdf:ID="PC">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </owl:Class>
  <owl:Class rdf:ID="BreakRoom">
    <rdfs:subClassOf rdf:resource="#Location"/>
```

```
  </owl:Class>
  <owl:Class rdf:ID="ImageService">
    <rdfs:subClassOf rdf:resource="#Service"/>
  </owl:Class>
  <owl:Class rdf:about="#NetWork">
    <rdfs:subClassOf rdf:resource="#Context"/>
  </owl:Class>
  <owl:Class rdf:ID="ClassRoom">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </owl:Class>
  <owl:Class rdf:ID="LAN">
    <rdfs:subClassOf rdf:resource="#NetWork"/>
  </owl:Class>
  <owl:Class rdf:ID="OtherWorks">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>
  <owl:Class rdf:ID="TeaBreak">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="tobeSetOn">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="setNotificationMode"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Service"/>
    <rdfs:range rdf:resource="#Device"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#setNotificationMode">
    <owl:inverseOf rdf:resource="#tobeSetOn"/>
    <rdfs:domain rdf:resource="#Device"/>
    <rdfs:range rdf:resource="#Service"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="canBeUsedOn">
    <rdfs:domain rdf:resource="#Service"/>
    <rdfs:range rdf:resource="#Device"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="capableToProvide"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="attendClass">
    <rdfs:range rdf:resource="#Class"/>
    <rdfs:domain rdf:resource="#Student"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="attendedBy"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:TransitiveProperty rdf:ID="partOf">
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:ID="hasPart"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#Location"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#Location"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#hasPart">
    <rdfs:domain rdf:resource="#Location"/>
```

```
    <owl:inverseOf rdf:resource="#partOf"/>
    <rdfs:range rdf:resource="#Location"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:ID="locatedWith">
    <rdfs:domain rdf:resource="#Person"/>
    <owl:inverseOf rdf:resource="#locatedWith"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Person"/>
  </owl:TransitiveProperty>
  <owl:SymmetricProperty rdf:ID="locatedNear">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Person"/>
    <rdfs:domain rdf:resource="#Device"/>
    <owl:inverseOf rdf:resource="#locatedNear"/>
  </owl:SymmetricProperty>
  <owl:FunctionalProperty rdf:ID="engagedIn">
    <rdfs:range rdf:resource="#Activity"/>
    <rdfs:domain rdf:resource="#Person"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="scheduledIn">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Location"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasPreferenceFor">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#MessageService"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasScheduleType">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >strict</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >relaxed</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
          </rdf:rest>
```

```
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
  </owl:FunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#holds">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#locatedIn"/>
    <rdfs:range rdf:resource="#Person"/>
    <rdfs:domain rdf:resource="#Location"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#officeOf">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#hasOffice"/>
    <rdfs:domain rdf:resource="#Office"/>
    <rdfs:range rdf:resource="#Person"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#contains">
    <owl:inverseOf rdf:resource="#containedIn"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Device"/>
    <rdfs:domain rdf:resource="#Location"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#ownerOf">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Device"/>
    <owl:inverseOf rdf:resource="#ownedBy"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  <owl:DataRange>
    <owl:oneOf rdf:parseType="Resource">
      <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >immobile</rdf:first>
      <rdf:rest rdf:parseType="Resource">
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
        <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >mobile</rdf:first>
      </rdf:rest>
    </owl:oneOf>
  </owl:DataRange>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355)
http://protege.stanford.edu -->
```

IV. PiCASO demonstration sample context data

```
<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.cocasp.fr/forumf.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
xml:base="http://www.cocasp.fr/forumf.owl">

<TeaBreak rdf:ID="AfternoonTea">
     <hasScheduleType rdf:resource="#relaxed"/>
     <scheduledIn rdf:resource="#Room306"/>
</TeaBreak>
<Student rdf:ID="Alice">
     <attendClass rdf:resource="#Class_5"/>
     <attendClass rdf:resource="#Class_ContextManagement"/>
     <hasOffice rdf:resource="#OfficeWithAlice"/>
     <ownerOf rdf:resource="#PCAlice"/>
     <ownerOf rdf:resource="#PDAAlice"/>
     <studentOf rdf:resource="#Dave"/>
</Student>
<ClassRoom rdf:ID="AmphiRoom1">
     <reservedFor rdf:resource="#Class_ContextManagement"/>
</ClassRoom>
<Student rdf:ID="Bob">
     <attendClass rdf:resource="#Class_ContextManagement"/>
     <attendClass rdf:resource="#Class_SemanticWeb"/>
     <hasOffice rdf:resource="#OfficeWithBob"/>
     <ownerOf rdf:resource="#PCBob"/>
     <ownerOf rdf:resource="#PDABob"/>
     <studentOf rdf:resource="#Eve"/>
     <studentOf rdf:resource="#Professor_11"/>
</Student>
<Meeting rdf:ID="BrainStorming1">
     <hasScheduleType rdf:resource="#relaxed"/>
     <scheduledIn rdf:resource="#MeetingRoom_1"/>
</Meeting>
<Research rdf:ID="Browsing">
     <hasEndTime rdf:resource="#2007-06-02T00:00:00"/>
     <hasScheduleType rdf:resource="#relaxed"/>
</Research>
<Student rdf:ID="Carol">
     <attendClass rdf:resource="#Class_SemanticWeb"/>
     <hasOffice rdf:resource="#OfficeWithCarol"/>
     <ownerOf rdf:resource="#PCCarol"/>
     <ownerOf rdf:resource="#PDACarol"/>
     <studentOf rdf:resource="#Eve"/>
</Student>
<Library rdf:ID="CatalogueRoom">
     <partOf rdf:resource="#DocINSA"/>
</Library>
```

```
<ClassRoom rdf:ID="ClassRoom_1">
     <reservedFor rdf:resource="#Class_1"/>
</ClassRoom>
<ClassRoom rdf:ID="ClassRoom_2">
     <reservedFor rdf:resource="#Class_2"/>
</ClassRoom>
<Class rdf:ID="Class_ContextManagement">
     <attendedBy rdf:resource="#Alice"/>
     <attendedBy rdf:resource="#Bob"/>
     <attendedBy rdf:resource="#Student_22"/>
     <hasScheduleType rdf:resource="#strict"/>
     <offeredBy rdf:resource="#Dave"/>
     <scheduledIn rdf:resource="#AmphiRoom1"/>
</Class>
<Class rdf:ID="Class_SemanticWeb">
     <attendedBy rdf:resource="#Bob"/>
     <attendedBy rdf:resource="#Carol"/>
     <attendedBy rdf:resource="#Student_21"/>
     <hasScheduleType rdf:resource="#strict"/>
     <offeredBy rdf:resource="#Eve"/>
</Class>
<MessageService rdf:ID="ColleagueForTea">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     <canBeUsedOn rdf:resource="#PCDave"/>
     <canBeUsedOn rdf:resource="#PCEve"/>
     <canBeUsedOn rdf:resource="#PDA_42"/>
     <canBeUsedOn rdf:resource="#PDA_43"/>
     <canBeUsedOn rdf:resource="#PDA_44"/>
     <canBeUsedOn rdf:resource="#PDA_45"/>
     …….
     <comment rdf:resource="#Your colleague invites you for tea!"/>
     <hasContent rdf:resource="#Your colleague invites you for
tea!"/>
</MessageService>
<MessageService rdf:ID="ColleagueHere">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     <canBeUsedOn rdf:resource="#PCDave"/>
     <canBeUsedOn rdf:resource="#PCEve"/>
     <canBeUsedOn rdf:resource="#PDAEve"/>
     <canBeUsedOn rdf:resource="#PDA_31"/>
     <canBeUsedOn rdf:resource="#PDA_32"/>
     <canBeUsedOn rdf:resource="#PDA_41"/>
     <canBeUsedOn rdf:resource="#PDA_42"/>
     <canBeUsedOn rdf:resource="#PDA_43"/>
     <canBeUsedOn rdf:resource="#PDA_44"/>
     <canBeUsedOn rdf:resource="#PDA_45"/>
     ……
     <comment rdf:resource="#Your Colleague is here!"/>
     <hasContent rdf:resource="#Your Colleague is here!"/>
</MessageService>
<Library rdf:ID="Corridor_Library">
     <partOf rdf:resource="#DocINSA"/>
```

```
</Library>
<PC rdf:ID="PCAlice">
     <capableToProvide rdf:resource="#ColleagueForTea"/>
     <capableToProvide rdf:resource="#ColleagueHere"/>
     <capableToProvide rdf:resource="#Dairy1"/>
     <capableToProvide rdf:resource="#ProfessorForTea"/>
     <capableToProvide rdf:resource="#ProfessorHere"/>
     <capableToProvide rdf:resource="#RingTone"/>
     <capableToProvide rdf:resource="#SilentMode"/>
     <capableToProvide rdf:resource="#StudentForTea"/>
     <capableToProvide rdf:resource="#StudentHere"/>
     <connectedTo rdf:resource="#PC2PC"/>
     <connectedTo rdf:resource="#PDA2PC"/>
     <connectedTo rdf:resource="#Wired"/>
     <connectedTo rdf:resource="#Wireless"/>
     <containedIn rdf:resource="#OfficeWithAlice"/>
     <hasProcessorSpeed rdf:resource="#High"/>
     <hasScreenSize rdf:resource="#Large"/>
     <hasStorageCapacity rdf:resource="#Large"/>
     <ownedBy rdf:resource="#Alice"/>
</PC>
<MessageService rdf:ID="ProfessorForTea">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     <canBeUsedOn rdf:resource="#PDA_43"/>
     <canBeUsedOn rdf:resource="#PDA_44"/>
     <canBeUsedOn rdf:resource="#PDA_45"/>
     ……...
     <comment rdf:resource="#Your professor  invites you for tea!"/>
     <hasContent rdf:resource="#Your professor  invites you for
tea!"/>
</MessageService>
<MessageService rdf:ID="ProfessorHere">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     <canBeUsedOn rdf:resource="#PCDave"/>
     <canBeUsedOn rdf:resource="#PCEve"/>
     <canBeUsedOn rdf:resource="#PDA_44"/>
     <canBeUsedOn rdf:resource="#PDA_45"/>
     …. …….
     <comment rdf:resource="#Your professor is here!"/>
     <hasContent rdf:resource="#Your professor is here!"/>
</MessageService>
<Professor rdf:ID="Professor_11">
     <advisorOf rdf:resource="#Bob"/>
     <advisorOf rdf:resource="#Student_21"/>
     <advisorOf rdf:resource="#Student_23"/>
     <hasOffice rdf:resource="#Office_1"/>
     <offerClass rdf:resource="#Class_1"/>
     <ownerOf rdf:resource="#PC_11"/>
     <ownerOf rdf:resource="#PDA_31"/>
</Professor>
<Professor rdf:ID="Professor_12">
     <advisorOf rdf:resource="#Student_22"/>
```

```
     <advisorOf rdf:resource="#Student_24"/>
     <hasOffice rdf:resource="#Office_2"/>
     <offerClass rdf:resource="#Class_2"/>
     <ownerOf rdf:resource="#PC_12"/>
     <ownerOf rdf:resource="#PDA_32"/>
</Professor>
<Library rdf:ID="ReadingRoom">
     <partOf rdf:resource="#DocINSA"/>
</Library>
<MessageService rdf:ID="RingTone">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     <canBeUsedOn rdf:resource="#PCDave"/>
     <canBeUsedOn rdf:resource="#PDA_44"/>
     <canBeUsedOn rdf:resource="#PDA_45"/>
     ……..
     <comment rdf:resource="#Ring normal tone on arrival of call."/>
     <hasContent rdf:resource="#Ring normal tone on arrival of call
!"/>
</MessageService>
<BreakRoom rdf:ID="Room306">
     <reservedFor rdf:resource="#AfternoonTea"/>
     <reservedFor rdf:resource="#MorningTea"/>
</BreakRoom>
<MessageService rdf:ID="StudentForTea">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     … …..
     <comment rdf:resource="#Your student invites you for tea!"/>
     <hasContent rdf:resource="#Your student invites you for tea!"/>
</MessageService>
<MessageService rdf:ID="StudentHere">
     <canBeUsedOn rdf:resource="#PCAlice"/>
     <canBeUsedOn rdf:resource="#PCBob"/>
     <canBeUsedOn rdf:resource="#PCCarol"/>
     … …..
     <comment rdf:resource="#Your student is here!"/>
     <hasContent rdf:resource="#Your student is here!"/>
</MessageService>
</rdf:RDF>
```

V. PiCASO demonstration sample rules

```
@prefix pre: <http://www.cocasp.fr/forumf.owl#>
@prefix sc: <http://www.w3.org/2000/01/rdf-schema#>
@prefix tp: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix owl: <http://www.w3.org/2002/07/owl#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

# Location based rules=======================================
[locatedNearRule:
      (?p pre:ownerOf ?d)
      (?d rdf:type pre:PDA)
  -> (?p pre:locatedNear ?d)
]
[locatedWithrRule:
      (?p1 pre:locatedIn ?l)
      (?p2 pre:locatedIn ?l)
  -> (?p1 pre:locatedWith ?p2)
]
#MobilePhone in Library Rule=================================
[MobilePhoneRule:
      (?d pre:ownedBy ?p)
      (?d rdf:type pre:PDA)
      (?p pre:locatedIn ?l)
      (?l rdf:type pre:Library)
  ->  (?d pre:setNotificationMode pre:SilentMode)
]
[MobilePhoneRule:
      (?d pre:ownedBy ?p)
      (?d rdf:type pre:PDA)
      (?p pre:locatedIn ?S)
      (?s pre:componentOf ?l)
      (?l rdf:type pre:Library)
  ->       (?d pre:setNotificationMode pre:SilentMode)
]
#MobilePhone Rule============================================
[MobilePhoneRule:
      (?d pre:ownedBy ?p)
      (?d rdf:type pre:PDA)
      (?p pre:locatedIn ?l)
      (?l rdf:type pre:ClassRoom)
  ->       (?d pre:setNotificationMode pre:VibratingMode)
]
[MobilePhoneRule:
      (?d pre:ownedBy ?p)
      (?d rdf:type pre:PDA)
      (?p pre:locatedIn ?l)
      (?l rdf:type pre:BreakRoom)
  ->       (?d pre:setNotificationMode pre:MusicTone)
]
 [MobilePhoneRule:
      (?d pre:ownedBy ?p)
      (?d rdf:type pre:PDA)
      (?p pre:locatedIn ?l)
```

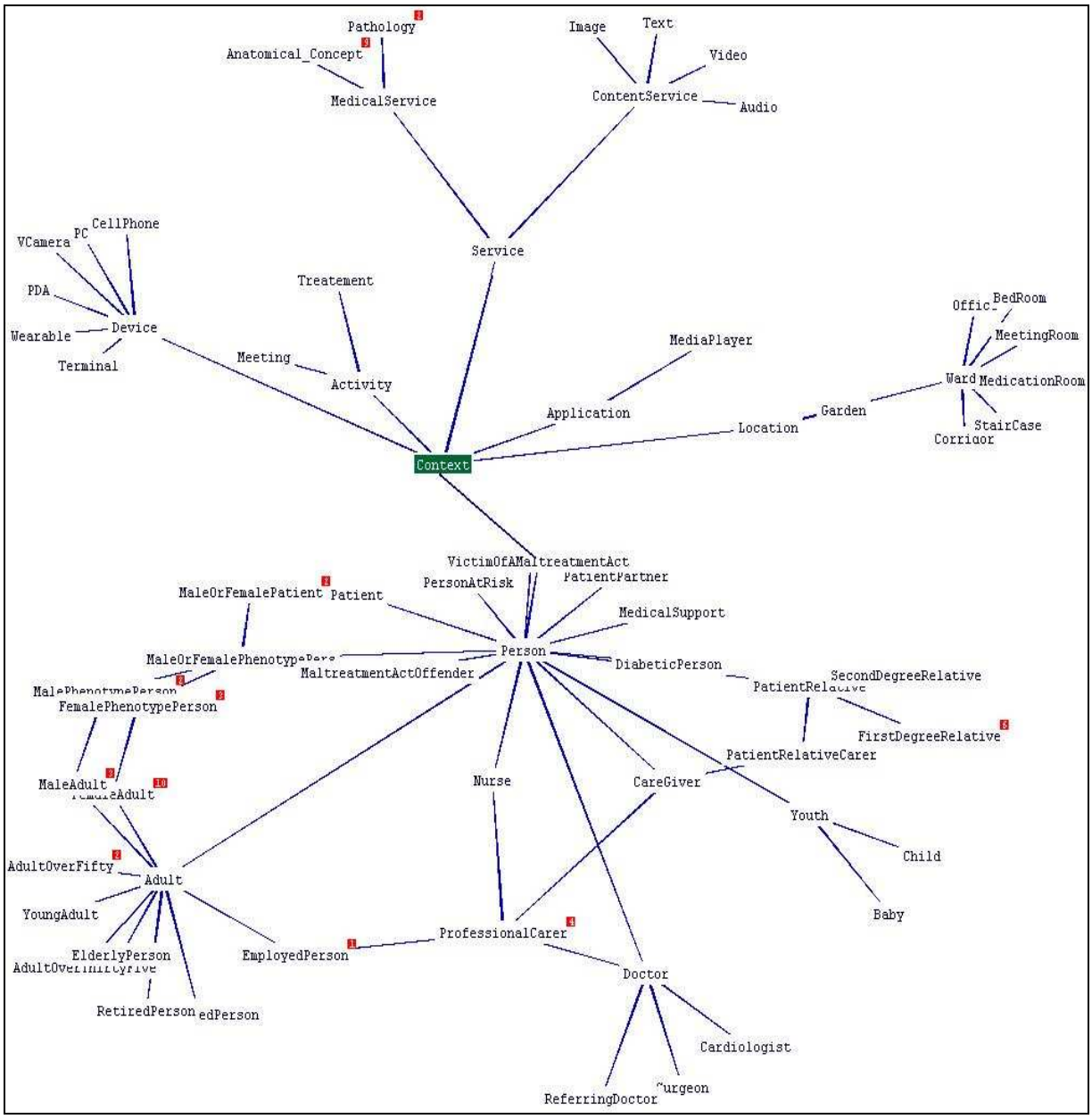
```
      (?l rdf:type pre:Office)
  ->       (?d pre:setNotificationMode pre:RingTone)
]
#hasMessage To go Rule ======================================
[InformRule1:
      (?S rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?S pre:studentOf ?P)
      (?S pre:locatedWith ?P)
  ->       (?S pre:hasMessageTogo pre:ProfessorHere)
]
[InformRule1:
      (?S rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?P pre:advisorOf ?S)
      (?S pre:locatedWith ?P)
  ->       (?S pre:hasMessageTogo pre:ProfessorHere)
]
[InformRule2:
      (?S rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?S pre:studentOf ?P)
      (?S pre:locatedWith ?P)
  ->       (?P pre:hasMessageTogo pre:StudentHere)
]
[InformRule2:
      (?S rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?P pre:AdvisorOf ?S)
      (?S pre:locatedWith ?P)
  ->       (?P pre:hasMessageTogo pre:StudentHere)
]
[InformRule3:
      (?S1 rdf:type pre:Student)
      (?S2 rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?S1 pre:studentOf ?P)
      (?S2 pre:studentOf ?P)
      (?S1 pre:locatedWith ?S2)
      notEqual(?S1,?S2)
  ->       (?S1 pre:hasMessageTogo pre:ColleagueHere)
]
[InformRule3:
      (?S1 rdf:type pre:Student)
      (?S2 rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?P pre:advisorOf ?S1)
      (?P pre:advisorOf ?S2)
      (?S1 pre:locatedWith ?S2)
      notEqual(?S1,?S2)
  ->       (?S1 pre:hasMessageTogo pre:ColleagueHere)
]
[InformRule4:
      (?P1 rdf:type pre:Professor)
      (?P2 rdf:type pre:Professor)
      (?S rdf:type pre:Student)
```

```
      (?S pre:studentOf ?P1)
      (?S pre:studentOf ?P2)
      (?P1 pre:locatedWith ?P2)
      notEqual(?P1,?P2)
  ->       (?P1 pre:hasMessageTogo pre:ColleagueHere)
]
[InformRule4:
      (?P1 rdf:type pre:Professor)
      (?P2 rdf:type pre:Professor)
      (?S rdf:type pre:Student)
      (?P1 pre:advisorOf ?S)
      (?P2 pre:advisorOf ?S)
      (?P1 pre:locatedWith ?P2)
      notEqual(?P1,?P2)
  ->       (?P1 pre:hasMessageTogo pre:ColleagueHere)
]
#=========================================================
[InviteRule1:
      (?S1 rdf:type pre:Student)
      (?S2 rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?S1 pre:studentOf ?P)
      (?S2 pre:studentOf ?P)
      (?S1 pre:engagedIn ?A1)
      (?A1 rdf:type pre:TeaBreak)
      (?S2 pre:engagedIn ?A2)
      (?A2 pre:hasScheduleType ?t)
      equal(?t,pre:relaxed)
      notEqual(?S1,?S2)
  ->       (?S2 pre:hasMessageTogo pre:ColleagueForTea)
]
[InviteRule1:
      (?S1 rdf:type pre:Student)
      (?S2 rdf:type pre:Student)
      (?P rdf:type pre:Professor)
      (?P pre:advisorOf ?S1)
      (?P pre:advisorOf ?S2)
      (?S1 pre:engagedIn ?A1)
      (?A1 rdf:type pre:TeaBreak)
      (?S2 pre:engagedIn ?A2)
      (?A2 pre:hasScheduleType ?t)
      equal(?t,pre:relaxed)
      notEqual(?S1,?S2)
  ->       (?S2 pre:hasMessageTogo pre:ColleagueForTea)
]
#=========================================================
[PostedForRule:
      (?p pre:hasMessageTogo ?m)
  -> (?m pre:postedFor ?p)
]
[PostedForRule:
      (?d pre:setNotificationMode ?m)
 ->  (?m pre:tobeSetOn ?d)
]
```

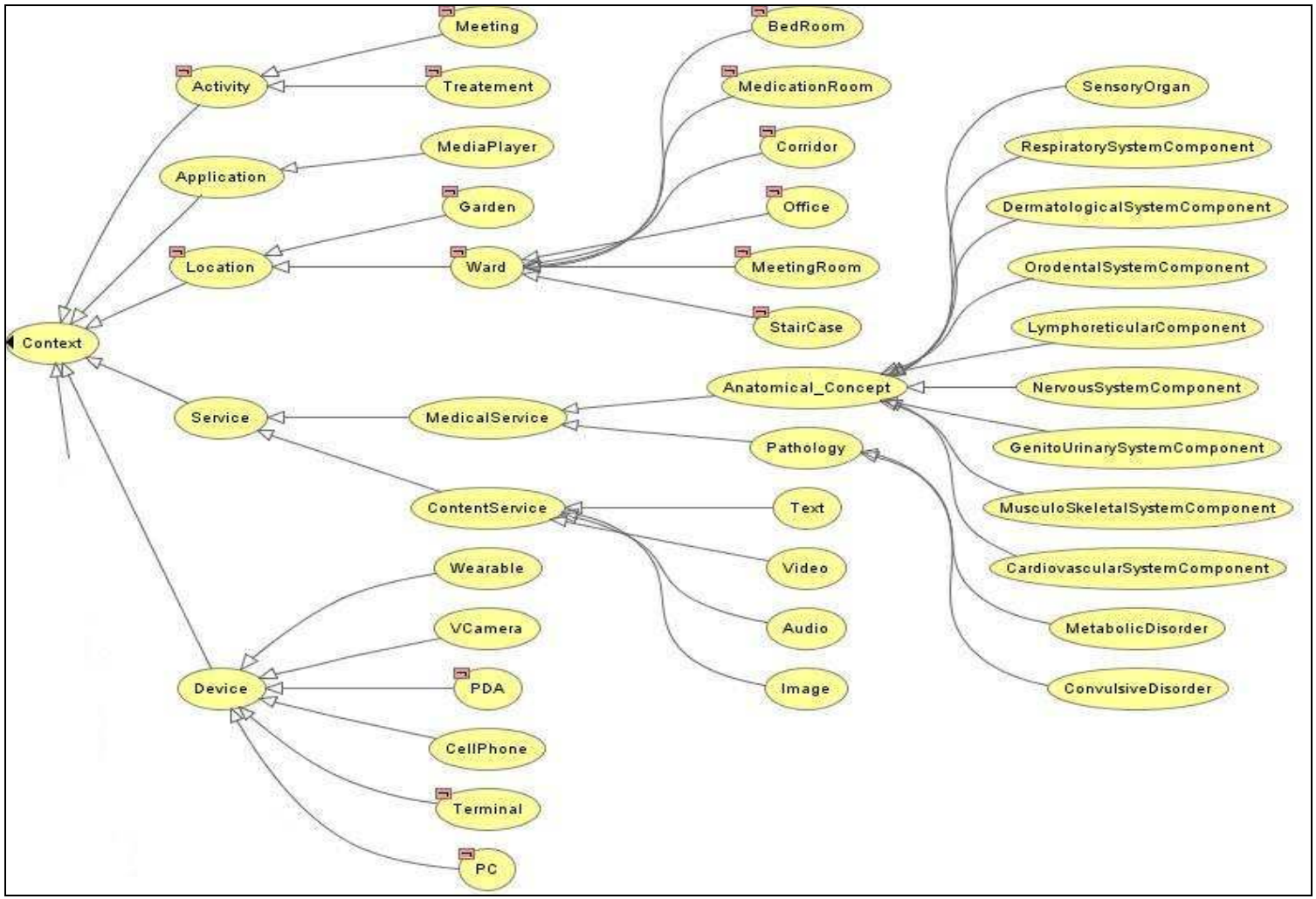
VI. Sample ontology for the smart hospital scenario

Ontology Source URL: http://liris.cnrs.fr/~edejene/PatientCareOnto/index.html

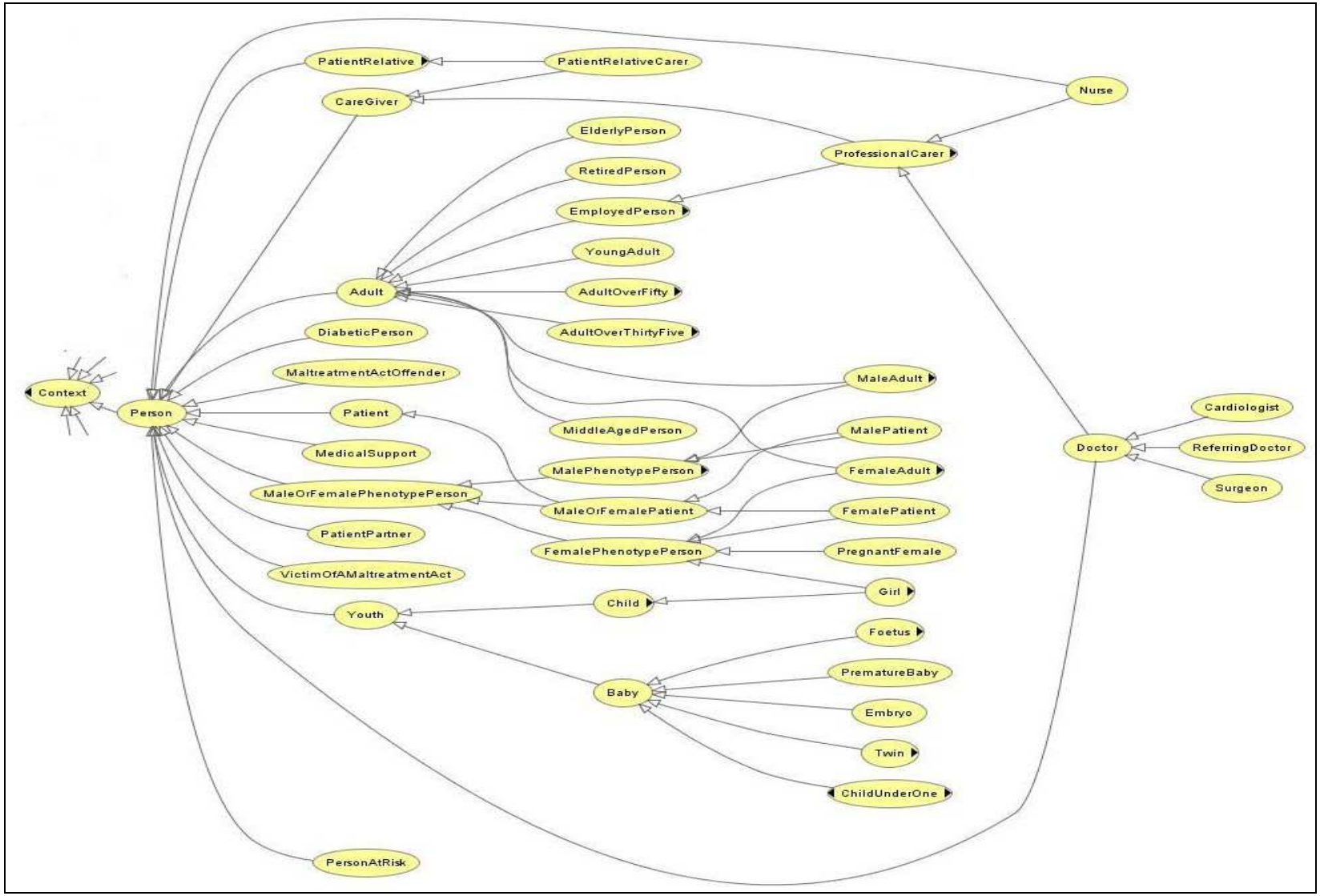
Branched graph showing the overall structure of ontology for the hospital scenario – Graph depth only up to 4th level



Ontology graph for the hospital Scenario (part 1) – Graph depth only up to 4th level



Ontology graph for the hospital Scenario (part 2) - Graph depth only up to 4th level



An excerpt of OWL ontologyfor the hospital scenario (Protégé)

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.cocasp.fr/patientcare.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.cocasp.fr/patientcare.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Ward">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Location"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Garden"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="PatientRelativeCarer">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CareGiver"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#PatientRelative"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#ProfessionalCarer">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CareGiver"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="EmployedPerson"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Patient">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Nurse"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Doctor"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="ReferringDoctor">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Doctor"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Dietician">
    <rdfs:subClassOf rdf:resource="#ProfessionalCarer"/>
  </owl:Class>
```

```
<owl:Class rdf:ID="BedRoom">
  <rdfs:subClassOf rdf:resource="#Ward"/>
</owl:Class>
<owl:Class rdf:about="#Child">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Youth"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Person">
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>
<owl:Class rdf:ID="MedicationRoom">
  <rdfs:subClassOf rdf:resource="#Ward"/>
</owl:Class>
<owl:Class rdf:ID="MaleOrFemalePatient">
  <rdfs:subClassOf rdf:resource="#Patient"/>
  <rdfs:subClassOf rdf:resource="#MaleOrFemalePhenotypePerson"/>
</owl:Class>
<owl:Class rdf:ID="Meeting">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Activity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="MetabolicDisorder">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Pathology"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#FemaleAdult">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#FemalePhenotypePerson"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Adult"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Treatement">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Activity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Son">
  <rdfs:subClassOf rdf:resource="#FirstDegreeRelative"/>
</owl:Class>
<owl:Class rdf:ID="RetiredPerson">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Adult"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChildUnderOne">
  <rdfs:subClassOf rdf:resource="#ChildUnderTwo"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Baby"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="FemaleOverFifty">
  <rdfs:subClassOf rdf:resource="#FemaleAdult"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#AdultOverFifty"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Image">
  <owl:disjointWith>
    <owl:Class rdf:about="#Audio"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Text"/>
  <owl:disjointWith rdf:resource="#Video"/>
  <rdfs:subClassOf rdf:resource="#ContentService"/>
</owl:Class>
<owl:Class rdf:ID="Daughter">
  <rdfs:subClassOf rdf:resource="#FirstDegreeRelative"/>
</owl:Class>
<owl:Class rdf:ID="OrodentalSystemComponent">
  <rdfs:subClassOf rdf:resource="#Anatomical_Concept"/>
</owl:Class>
<owl:Class rdf:ID="Infant">
  <rdfs:subClassOf rdf:resource="#ChildUnderOne"/>
</owl:Class>
<owl:Class rdf:ID="Physiotherapist">
  <rdfs:subClassOf rdf:resource="#ProfessionalCarer"/>
</owl:Class>
<owl:Class rdf:ID="VCamera">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Device"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Wife">
  <rdfs:subClassOf rdf:resource="#FemaleAdult"/>
</owl:Class>
<owl:Class rdf:about="#Pathology">
  <rdfs:subClassOf rdf:resource="#MedicalService"/>
</owl:Class>
<owl:Class rdf:ID="Wearable">
  <rdfs:subClassOf rdf:resource="#Device"/>
</owl:Class>
<owl:Class rdf:ID="StairCase">
  <rdfs:subClassOf rdf:resource="#Ward"/>
</owl:Class>
<owl:Class rdf:ID="Chiropodist">
  <rdfs:subClassOf rdf:resource="#ProfessionalCarer"/>
</owl:Class>
<owl:Class rdf:ID="ConjoinedTwin">
  <rdfs:subClassOf rdf:resource="#Twin"/>
</owl:Class>
<owl:Class rdf:ID="Neonate">
  <rdfs:subClassOf rdf:resource="#ChildUnderOne"/>
</owl:Class>
<owl:Class rdf:ID="NervousSystemComponent">
  <rdfs:subClassOf rdf:resource="#Anatomical_Concept"/>
</owl:Class>
```

```
<owl:Class rdf:ID="MalePatient">
  <rdfs:subClassOf rdf:resource="#MalePhenotypePerson"/>
  <rdfs:subClassOf rdf:resource="#MaleOrFemalePatient"/>
</owl:Class>
<owl:Class rdf:ID="LaboratoryTechnician">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#EmployedPerson"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="MaltreatmentActOffender">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="MusculoSkeletalSystemComponent">
  <rdfs:subClassOf rdf:resource="#Anatomical_Concept"/>
</owl:Class>
<owl:Class rdf:about="#CareGiver">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Father">
  <rdfs:subClassOf rdf:resource="#FirstDegreeRelative"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="capture">
  <rdfs:range rdf:resource="#Video"/>
  <rdfs:domain rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="broadcastTo">
  <rdfs:domain rdf:resource="#Video"/>
  <rdfs:range rdf:resource="#Terminal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="canProvide">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="availableOn"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Device"/>
  <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSchedule">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNurse">
  <rdfs:domain rdf:resource="#Patient"/>
  <rdfs:range rdf:resource="#Nurse"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#availableOn">
  <rdfs:range rdf:resource="#Device"/>
  <rdfs:domain rdf:resource="#Service"/>
  <owl:inverseOf rdf:resource="#canProvide"/>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="http://www.w3.or..#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/..owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Location"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
```

```
          <owl:Class rdf:about="#Device"/>
          <owl:Class rdf:about="#Person"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:TransitiveProperty>
  <owl:FunctionalProperty rdf:ID="hasBP1">
    <rdf:type rdf:resource="http://www.w3.or..owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Patient"/>
  </owl:FunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#hasPatient">
    <rdfs:domain rdf:resource="#Doctor"/>
    <owl:inverseOf rdf:resource="#hasDr"/>
    <rdfs:range rdf:resource="#Patient"/>
    <rdf:type rdf:resource="http://www.w3.org..owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  <StairCase rdf:ID="StairCase_Floor1"/>
  <MedicationRoom rdf:ID="ImergencyRoom01"/>
  <owl:DataRange/>
  <PC rdf:ID="PC_Garden02"/>
  <owl:DataRange/>
  <VCamera rdf:ID="VCamera_Garden_03"/>
  <Garden rdf:ID="Garden_Front"/>
  <Meeting rdf:ID="EvaluationMeeting"/>
  <Terminal rdf:ID="Terminal_Corridor_01"/>
  <BedRoom rdf:ID="BedRoom_10_02"/>
  <MeetingRoom rdf:ID="MeetingRoom_Floor2"/>
  <CellPhone rdf:ID="CellPhone_Pascal"/>
  <Patient rdf:ID="Michel"/>
  <CellPhone rdf:ID="CellPhone_Ada"/>
  <Treatement rdf:ID="EmergencyTreatement"/>
  <Terminal rdf:ID="Terminal_Corridor_02"/>
  <Context rdf:ID="Context_1"/>
  <MediaPlayer rdf:ID="iTune"/>
  <MediaPlayer rdf:ID="WindowsMplayer"/>
  <Office rdf:ID="Office_Secretary"/>
  <VCamera rdf:ID="VCamera_Garden_02"/>
  <owl:DataRange/>
  <StairCase rdf:ID="StairCase_Floor2"/>
  <Garden rdf:ID="Garden_BackDoor"/>
  <MeetingRoom rdf:ID="MeetingRoom_Floor1"/>
  <CellPhone rdf:ID="CellPhone_Michel"/>
  <Treatement rdf:ID="HeavySurgery"/>
  <Treatement rdf:ID="RegularTreatement"/>
  <BedRoom rdf:ID="BedRoom_10_01"/>
  <PC rdf:ID="PC_Central_02"/>
  <MedicalSupport rdf:ID="EmergencyGroup_01"/>
  <Doctor rdf:ID="Pascal"/>
  <Treatement rdf:ID="MiniSurgery"/>
  <PDA rdf:ID="PDA_Pascal"/>
  <Nurse rdf:ID="Ana"/>
  <PDA rdf:ID="PDA_Ada"/>
   . . . .
</rdf:RDF>
```

List of Publications

I. International Journals

- Ejigu D., Scuturici M., Brunie L., “Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing”, Journal of Computers (JCP), Volume 2 number 8, Academy Publishers, October 2007.
- Chaari, T., Ejigu D., Laforest F., Scuturici V.M., “A Comprehensive Approach to Model and Use Context for Adapting Applications in Pervasive Environments”, International Journal of Systems and Software (JSS), Volume 80/12, Elsevier Science Publishers, Amsterdam, 2007. pp.1973-1992.

II. International Conferences

- Ejigu D., Scuturici M., Brunie L.: “Semantic Approach to Context Management and Reasoning in Ubiquitous Context-Aware Systems”, In the proceedings of the IEEE International Conference on Digital Information Management (ICDIM’07), Lyon, France, October 2007. pp. 500-5005.
- Ejigu D., Scuturici M., Brunie L.: "An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing," *percomw*, Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), 2007. pp. 14-19.
- Ejigu D., Scuturici, M., Brunie L.: “CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing”, Fourth IEEE International Conference on Information Technology: New Generations, ITNG’07, Las Vegas, USA, April 2007. pp. 297-302.
- Scuturici M., Ejigu, D. : “Positioning Support in Pervasive Environments”, IEEE International Conference on Pervasive Services (ICPS'06), Lyon, France, June 2006, pp. 19-26.
- Chaari T., Ejigu D., Laforest F., Scuturici M., “Modeling and Using Context in Adapting Applications to Pervasive Environments”, IEEE International Conference on Pervasive Services (ICPS'06), Lyon, France, June 2006, pp. 111-120.

Curriculum Vitae

Personal Information

- Name: Dejene Ejigu Dedefa
- Sex: Male
- Marital Status: Married (three children)
- Nationality: Ethiopian
- Date of birth: July13, 1965
- Place of birth: Woliso, Ethiopia
- Language: English, French
  Amharic, Oromo (Ethiopian languages)
- Current Address: INSA de Lyon, LIRIS, Bat. Blaise Pascal 501.325
  20 Avenue Albert Einstein, 69621 Villeurbanne, France

  Tel. +33 (0)4 72 43 63 48

  Fax: +33 (0)4 72 43 87 13

Education

- Ph.D. candidate at INSA Lyon, October 2004 to date.
- Professional Certificate in Computer Technology, Kyoto School of Computer Science, Japan, 1997.
- M. Sc. Degree in Computer Science, University of Wales Swansea, UK, 1989.

  (Dissertation Title "*The Grid File: Implementation and Performance Analysis*")
- B. Sc. (*Distinction*) Degree in Statistics, Addis Ababa University, Ethiopia, 1986.
- Diploma in Auto-Mechanics & School Leaving *Very High Distinction* Certificate, 1982.

Work Experience

- Doctoral researcher at LIRIS laboratory, INSA Lyon, October 2004 to date.
- French language training at Alliance Françaises de Lyon and pre-doctoral research practice at INSA de Lyon, October 2003 to July 2004.
- Lecturer, Department of Computer Science, Addis Ababa University, September 2001 to 2003.
- Assistant Faculty Dean and Lecturer, Faculty of Science, Addis Ababa University, September 1999 to 2001.
- Lecturer, Department of Mathematics & Comp. Science, Addis Ababa University, Sep. 1997 to 1999.
- Senior Computer Expert and Software Team Leader, National Computer & Information Center, Ethiopian Science & Technology Commission, January 1995 to September 1997.
- Computer Expert, National Computer Center, Ethiopian Science and Technology Commission, October 1988 to January 1995.
- Junior Expert, Computer Research Unit, Ethiopian Science and Technology Commission, October 1986 to September 1987.

FOLIO ADMINISTRATIF
THESE SOUTENUE DEVANT L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

NOM : DEJENE EJIGU DATE de SOUTENANCE : 12 Décembre 2007
(avec précision du nom de jeune fille, le cas échéant)
Prénoms : DEDEFA
TITRE : Services Pervasifs Contextualisés : Modélisation et Mise en Œuvre
"Context Modeling and Collaborative Context-Aware Services for Pervasive Computing"
NATURE : Doctorat Numéro d'ordre : 2007-ISAL-0108

Ecole doctorale : École doctorale Informatique et Information pour la Société

Spécialité : Informatique

Cote B.I.U. - Lyon : T 50/210/19 / et bis CLASSE :

RESUME : Résumé : Les systèmes pervasifs visent à intégrer des services fournis par des dispositifs répartis communicants. De tels environnements ont comme objectif d'optimiser l'interaction de l'utilisateur avec les dispositifs intégrés, par exemple en permettant à l'utilisateur d'accéder à l'ensemble des informations disponibles et en adaptant celles-ci aux conditions matérielles effectives (qualité de service réseau, caractéristiques du matériel de connexion). Cela impose aux applications d'adapter dynamiquement leur fonctionnement aux caractéristiques de l'environnement (notion de "contexte d'exécution").

Dans cette thèse, nous proposons un modèle sémantiquement riche pour la collaboration, la représentation et la gestion du contexte. Nous utilisons un modèle de représentation du contexte fondé sur une approche hybride utilisant des ontologies et des bases de données relationnelles. Cette richesse de modélisation nous permet de sélectionner de manière efficace les informations contextuelles pertinentes et ainsi d'améliorer les performances du processus de raisonnement mis en oeuvre dans l'analyse du contexte d'exécution.

Nous présentons la plateforme logicielle d'intégration de services pervasifs que nous avons développée. Cette plateforme s'appuie sur la méthodologie et les modèles de représentation et de gestion du contexte proposés dans la thèse. Elle permet une interaction "contextualisée" des services fournis par les dispositifs participants, offrant en particulier des mécanismes d'adaptation au contexte et de déclenchement proactif ou réactif de services en réponse à une évolution du contexte. Cette plate-forme implémente le protocole JXTA dans ses composants de collaboration et utilise la librairie JENA pour le raisonnement (déclaration et interprétation des règles d'analyse du contexte).

Des démonstrateurs ont été développés et testés illustrant l'utilisation de la plate-forme dans trois cas d'utilisation liés à des domaines applicatifs variés : les réseaux sociaux, l'hôpital intelligent, l'adaptation d'IHM au contexte.

Les résultats obtenus illustrent la performance, la robustesse et l'extensibilité de l'approche proposée.

MOTS-CLES : Informatique Pervasif, Réactivité au Contexte, Contexte Modélisation, Raisonnement Sémantique, Ontologie du Contexte, Informatique Collaborative

Laboratoire (s) de recherche : LIRIS - Laboratoire d'InfoRmatique en Image et Systèmes d'information

Directeur de thèse: Lionel Brunie Co-directeur : Vasile-Marian Scuturici

Rapporteurs :Bruno DEFUDE, Professeur, Institut National des Telecommunications, Paris

Manish PARASHAR, Professeur, Rutgers University, New Jersey, USA

Président de jury : Prof. Aris OUKSEL, University d'Illinois à Chicago, USA

Composition du jury :

Prof. Bruno DEFUDE, INT Paris	(Rapporteur)
Prof. Aris OUKSEL, Université d'Illinois à Chicago, USA	(Examinateur)
Prof. Jean-Marc PETIT, INSA de Lyon	(Examinateur)
Dr. Thierry DELOT, Universteé de Valenciennes	(Examinateur)
Dr. Richard CHBEIR, Université de Bourgogne	(Examinateur)
Prof. Lionel BRUNIE, INSA de Lyon	(Directeur de thèse)
Dr. Marian SCUTURICI, INSA de Lyon	(Co-directeur de thèse)