Doctorat en co-accréditation
Télécom SudParis et Université d'Évry-Val-d'Essonne

Spécialité : Informatique

École doctorale : Science et Ingénierie

Thèse présentée par

Zied ABID

Pour obtenir le grade de
Docteur de Télécom SudParis

Gestion de la qualité de contexte pour l'intelligence ambiante

Soutenue le 21 Décembre 2012

Devant le jury composé de :

Rapporteurs :	M. Hervé Martin	Professeur, Université Joseph Fourier, Grenoble 1
	M. Philippe Roose	Maître de conférences (HDR), Université de Pau et des Pays de l'Adour
Examinateurs :	M. Antoine Beugnard	Professeur, Télécom Bretagne
	M. Guy Bernard	Professeur Émérite, Télécom SudParis (Directeur de thèse)
	Mme Sophie Chabridon	Maître de conférences, Télécom SudParis (Encadrante)

Thèse n° 2012TELE0041

Mis en page avec la classe thloria-tsp adaptée pour Télécom SudParis.

Remerciements

Le travail que j'ai réalisé au cours de ma thèse n'aurait certainement pas été possible sans l'aide, les encouragements et les conseils de nombreuses personnes qui y ont été impliquées de près ou de loin. C'est pourquoi je tiens à leur exprimer par ces quelques lignes toute ma reconnaissance, en particulier :

Je suis très reconnaissant envers M. Hervé Martin, Professeur à l'Université Joseph Fourier, Grenoble 1 et M. Philippe Roose, Maître de conférences (HDR) à l'Université de Pau et des Pays de l'Adour, pour avoir accepté de rapporter mes travaux et de figurer dans mon jury.

Je tiens à exprimer ma gratitude à M. Antoine Beugnard, Professeur à Télécom Bretagne, pour avoir accepté d'examiner mes travaux et faire partie du jury.

Je remercie M. Bruno Defude pour m'avoir accueilli dans le département Informatique et m'avoir permis d'effectuer cette thèse dans de bonnes conditions.

Je remercie mon directeur de thèse M. Guy Bernard. J'ai particulièrement apprécié sa disponibilité, son ouverture d'esprit et les nombreuses discussions qui ont animé ces années de thèse.

Je remercie chaleureusement mon encadrante Sophie Chabridon pour m'avoir encadré tout au long de cette thèse. Je la remercie pour ses conseils, sa disponibilité, sa patience et son dynamisme qui m'ont été très profitables et qui m'ont permis d'avancer tout au long de cette thèse.

Je tiens à remercier spécialement Denis Conan et Chantal Taconet pour leurs conseils, leurs encouragements et leur gentillesse ; j'ai appris beaucoup durant nos échanges et nos débats constructifs.

Je tiens à remercier les membres de l'équipe MARGE, en particulier Djamel Belaïd, Michel Simatic et Sébastien Leriche pour leur accueil et leur collaboration durant tout mon parcours de thèse.

Je tiens également à remercier les membres du département Informatique et de la direction de la Recherche pour leur aide et collaboration, en particulier notre chère assistante de gestion Mme Brigitte Houassine et à la direction de la Recherche, Mme Xayplathi Lyfoung. Je remercie M. Samir Tata, Walid Gaaloul, Olivier Berger et Amel Bouzeghoub pour leurs conseils et leurs encouragements.

Merci également à mes collègues et amis Mohamed Sellami, Léon Lim et Mohammed El-Amine Matougui, Dorsaf Zekri, Imen Lahmar, Djallel Bouneffouf et Jerome Sicard qui m'ont toujours soutenu.

Je dédie cette thèse
à mes chers parents,
ma chère femme,
et mon cher fils.

Résumé

Gestion de la qualité de contexte pour l'intelligence ambiante

L'informatique sensible au contexte vise à réduire la quantité d'informations explicites qu'un utilisateur doit fournir pour que le système accomplisse la tâche souhaitée. Ceci est particulièrement vrai dans le domaine récent de l'intelligence ambiante où les objets de la vie courante deviennent capables de déclencher une action ou un échange spontané d'informations, sans interaction avec l'utilisateur. Les progrès techniques en matière de réseaux de communication sans fil, d'équipements mobiles individuels, de capteurs et de logiciels embarqués, rendent aujourd'hui possibles des services aux usagers dépendants du contexte, mais les applications concrètes demeurent encore très limitées. Les travaux existants dans la littérature décomposent la gestion de contexte en quatre fonctionnalités : la collecte, l'interprétation, la détection de situations et l'utilisation pour l'adaptation. L'élément discriminant des solutions existantes est la qualité des informations abstraites obtenues par inférence et devant caractériser les situations de l'utilisateur. Les limites de ces solutions sont le manque de composition aisée des informations de contexte, le passage à l'échelle, tant en termes de quantité d'informations de contexte que de nombre d'applications clientes, l'absence de garantie sur la cohérence et la qualité des informations de contexte, et le manque de solutions intergicielles permettant de libérer le concepteur d'applications des aspects liés à la gestion de contexte.

Nous nous intéressons dans cette thèse à la gestion de la qualité de contexte (QoC) dans un environnement ambiant. Les problématiques de gestion de la qualité de contexte sont multiples : choisir la méthode adéquate pour la gestion du contexte, extraire la qualité associée au contexte, interpréter et analyser la qualité de contexte pour les applicationsensibles au contexte. Nous proposons de répondre à ces problématiques en intégrant la qualité de contexte au sein de la plateforme de gestion de contexte COSMOS [1] de l'équipe MARGE [2] de Télécom SudParis. Afin d'effectuer cette intégration, nous avons conçu des éléments spécifiques à la qualité de contexte et avons mis en place une gestion fine et efficiente de cette qualité en limitant le surcoût associé. Nous proposons également un processus de conception basé sur l'ingénierie dirigée par les modèles afin de générer les éléments requis à la gestion de la qualité de contexte. Nous avons validé nos contributions à l'aide de deux applications fonctionnant sur téléphone mobile : une application de "vente flash" dans un centre commercial et une application de détection de localisation sur un campus. Les tests de performances que nous avons effectués permettent de comparer les résultats avec et sans la prise en compte de la QoC et montrent le faible coût de la gestion de la qualité par rapport aux améliorations apportées aux applications sensibles au contexte.

Mots clés : intelligence ambiante, contexte, qualité de contexte, intergiciel.

1. http ://picoforge.int-evry.fr/projects/svn/cosmos
2. http ://www-inf.it-sudparis.eu/MARGE

Abstract

Management of the quality of context for ambient intelligence

Context-aware computing aims to reduce the amount of explicit information required from a user for a system to perform a task. This is particularly true in the recent domain of ambient intelligence where everyday life objects are able to trigger an action or a spontaneous information exchange, without any interaction with the user. Technical advances in wireless communication, personal mobile devices, sensors and embedded software make context-aware services possible, but concrete applications are still very limited. The solutions proposed in the literature decompose context management into four functions : acquisition, interpretation, situation detection and application adaptation. The differentiating element in these proposals is the quality of the high-level context information obtained by inference and characterising the situation of the user. The limits of these solutions are the difficulty for composing context information, scalability in terms of the quantity of context information and of the number of client applications, the absence of guarantee on the consistency of context information and the lack of middleware solutions able to free the designer of context-aware applications from the management of context data.

In this thesis, we are interested in the management of the quality of context information (QoC) in an ambient environment. There are several key issues in QoC management : choosing the adequate method for context management, extracting the quality associated to the context, analysing and interpreting the quality of the context with regard to the requirements of context-aware applications. We propose to answer these questions by integrating QoC management into the COSMOS context management framework [3] developed by the MARGE team [4] of Télécom SudParis. For this purpose, we have designed the necessary components dedicated to QoC management and we have implemented the mechanisms allowing a fine-grain manipulation of the QoC together with a limitation of the associated overhead. We also propose a design process based on model-driven engineering in order to automatically generate the elements responsible of QoC management. We validate our contributions through the development of two prototype applications running on mobile phones : a Flash sale offer application to be used in malls and a location detection application proposed to the students of a campus. The performance tests we have conducted allow to compare the results obtained with and without taking into account the QoC and show the low overhead associated to QoC management with regard to the benefits brought to context-aware applications and services.

Keywords : ambient intelligence, context, quality of context, middleware.

3. http ://picoforge.int-evry.fr/projects/svn/cosmos
4. http ://www-inf.it-sudparis.eu/MARGE

Publications dans le cadre de la thèse

Revues internationales

[Chabridon et al., 2012a] CHABRIDON Sophie, CONAN Denis, ABID Zied, TACONET Chantal, Building Ubiquitous QoC-Aware Applications through Model-Driven Software Engineering, Science of Computer Programming, Elsevier, Online publication complete : 1-SEP-2012, http ://dx.doi.org/10.1016/j.scico.2012.07.019

Conférences internationales

[Chabridon et al., 2011a] CHABRIDON Sophie, ABID Zied, TACONET Chantal, CONAN Denis, A model driven approach for the QoC-awareness of ubiquitous applications. UCAmI '11 : 5th International Symposium on Ubiquitous Computing and Ambient Intelligence, Ambient Intelligence Association (AIAm), 05-09 december 2011, Riviera Maya, Mexico, 2011, ISBN 978-84-694-9677-0

[Chabridon et al., 2011b] CHABRIDON Sophie, NGO Cao Cuong, ABID Zied, CONAN Denis, TACONET Chantal, OZANNE Alain, Towards QoC-aware location-based services. DAIS '11 : 11th IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 06-09 june 2011, Reykjavik, Iceland, 2011, vol. 6723, ISBN 978-3-642-21386-1

Conférences nationales

[Chabridon et al., 2012b] CHABRIDON Sophie, CONAN Denis, ABID Zied, TACONET Chantal, Ingénierie dirigée par les modèles pour la construction d'applications ubiquitaires sensibles à la qualité du contexte. Ubimob '12 : 8èmes journées francophones Mobilité et Ubiquité, Toulouse : Cépaduès, 04-06 juin 2012, Anglet, France, 2012, ISBN 978-2-36493-018-6

Workshops internationaux

[Abid and Chabridon, 2011] ABID Zied, CHABRIDON Sophie, A fine-grain approach for evaluating the quality of context. CoMoRea '11 : 8th IEEE Workshop on Context Modeling and Reasoning, IEEE, 21-25 march 2011, Seattle, United States, 2011, ISBN 978-1-61284-938-6

[Chabridon et al., 2010] CHABRIDON Sophie, CONAN Denis, TACONET Chantal, NGUYEN Kinh, NGO Cao Cuong, LIM Léon, ABID Zied, MDE, DSL and Tooling for Effective Context Management in Ubiquitous Computing, Proc. MobiCASE Workshop on Mobile Software Engineering, Santa Clara, CA, USA. 2010

[Abid et al., 2009b] ABID Zied, CHABRIDON Sophie, CONAN Denis, A framework for quality of context management. QuaCon '09 : First International Workshop on Quality of Context, Berlin, Heidelberg, New York : Springer-Verlag, 25-26 june 2009, Stuttgart, Germany, 2009, vol. 5786/2009, ISBN 978-3-642-04558-5

Workshops nationaux

[Abid et al., 2009a] ABID Zied, CHABRIDON Sophie, CONAN Denis, Cohérence et qualité des informations de contexte en environnement pervasif. 3ème Workshop CDUR '09 : Cohérence des Données en Univers Réparti, 11 septembre 2009, Toulouse, France, 2009

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Sommaire

Nous introduisons dans ce chapitre le domaine de recherche abordé dans cette thèse. Nous commençons par délimiter le cadre de notre travail au sein de l'informatique ambiante, puis nous évoquons les motivations qui sont à l'origine de notre travail. Nous donnons enfin un résumé des contributions proposées et présentons l'organisation de ce document.

1.1 Cadre de la thèse

La vision d'une informatique ubiquitaire invisible, disparaissant de la vue des utilisateurs, et présente en tout lieu et en toute chose telle que proposée par Mark Weiser au début des années 1990 [Weiser, 1991] reste aujourd'hui encore, plus de vingt après, une référence pour les nombreux travaux qui ont suivi dans le but de développer cette idée d'une intelligence ambiante [Sadri, 2011]. Weiser oppose l'informatique invisible à la réalité virtuelle consistant à recréer le monde dans une machine et non pas à intégrer des machines dans la réalité. Les formidables avancées en termes de dispositifs mobiles et de communication sans fil ont rendu possible cette intelligence ambiante. L'espace ambiant devient capable de percevoir des changements en son sein au travers d'une multiplicité de dispositifs intelligents et mobiles dont l'accumulation fournit une puissance de calcul substantielle. Les interactions ne se font plus via un utilisateur humain mais directement entre les objets connectés. L'informatique ubiquitaire a alors pour objectif de gérer ces objets et leurs interconnexions afin de rendre des services de manière transparente pour l'utilisateur. Cela passe par la perception du contexte de l'utilisateur pour déterminer quelle est sa situation et quel comportement doit avoir le système informatique ; on parle alors également d'informatique sensible au contexte. L'importance de ce concept de

contexte pour l'informatique ubiquitaire a été analysée notamment par [Coutaz et al., 2005] en démontrant que les informations de contexte permettent d'enrichir les activités humaines avec de nouveaux services capables de s'adapter aux circonstances dans lesquelles ils sont utilisés. La figure 1.1 donne une vision de l'évolution de l'informatique ubiquitaire depuis l'informatique mobile vers l'intelligence ambiante. On peut remarquer que la sensibilité au contexte est une préoccupation récurrente. Cependant, la qualité des informations de contexte influe sur les décisions d'adaptation. Par exemple, l'information de localisation « Marie est dans un rayon de 3 km autour du centre commercial » ne permettra pas la même adaptation de la présentation des commerces de proximité que l'information « Marie passe devant le magasin de chaussures ». Au sein de l'intelligence ambiante, ce travail de thèse s'attache à proposer les mécanismes permettant de **caractériser la qualité des informations de contexte** afin de garantir la **pertinence des décisions d'adaptation** qui en découlent.

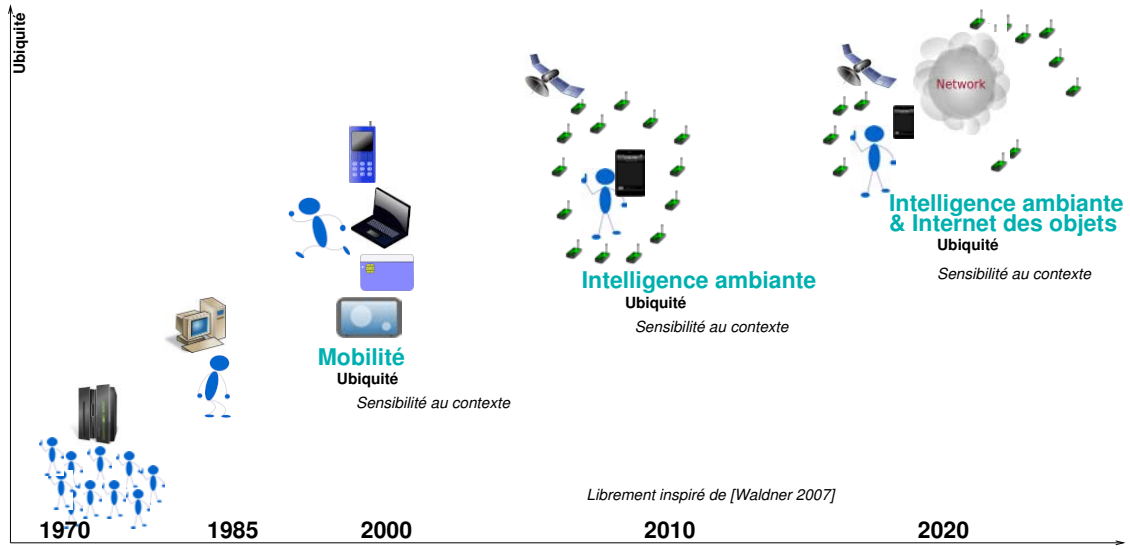


FIGURE 1.1 – Évolution de l'informatique ubiquitaire

1.2 Motivations

Les progrès techniques en matière de réseaux de communication sans fil, d'équipements mobiles personnels, de capteurs et de logiciels embarqués, rendent aujourd'hui possibles des services aux usagers dépendants du contexte. Les opportunités d'applications sont nombreuses : par exemple, la téléphonie mobile où le téléphone sait comment réagir aux appels selon la situation de l'utilisateur, les services en entreprise tenant compte en temps en réel de paramètres physiques extérieurs, les jouets interactifs réagissant en fonction de l'enfant et de son comportement, ou encore les parkings indiquant aux conducteurs où trouver une place libre. La figure 1.2

illustre quelques exemples d'applications ubiquitaires. Cependant, les applications disponibles aujourd'hui demeurent encore très limitées dans les fonctionnalités qu'elles offrent.

FIGURE 1.2 – Informatique sensible au contexte en environnement ubiquitaire
Source : `http://www.uidcenter.org`

En regardant de plus près les verrous auxquels sont confrontés les scientifiques, l'élément discriminant des solutions proposées jusque-là est la prise en compte ou non de la qualité des informations de contexte. Dans le mesure où ces informations de contexte ont un impact direct sur les décisions qui peuvent être prises par le système, il est primordial de maîtriser leur qualité et d'utiliser des mécanismes efficaces pour ce faire. L'un des freins à la concrétisation de l'intelligence ambiante n'est donc plus dans «qu'est-ce que le contexte» mais dans comment créer des applications intelligentes et proactives en les rendant sensibles à un contexte qu'elles peuvent définir, vérifier et valider, à travers la maîtrise de la qualité de ce contexte. Disposer d'une connaissance de la qualité du contexte aide à prendre les décisions appropriées et permet d'identifier les informations de contexte incertaines, économisant ainsi du temps de calcul pour dériver une description pertinente du phénomène observé. La QoC permet de gérer des priorités entre toutes les informations de contexte collectées. [Manzoor et al., 2008] présente un exemple d'application pour la gestion de situations d'urgence. Des sauveteurs sont présents sur le terrain et utilisent des appareils sans fil pour communiquer, collecter des informations et prendre des photos. Par ailleurs, les responsables disposent d'images satellite prises sur le lieu du sinistre.

Un responsable pourra ainsi privilégier les photos envoyées par un sauveteur ayant un niveau de confiance élevé indiquant qu'il est proche de la zone du sinistre et dont l'appareil de prise de photo possède une grande précision, par rapport à des images satellites.

1.3 Contributions

Un certain nombre de travaux récents dans le domaine du logiciel participent à l'essor de l'intelligence ambiante, notamment pour identifier des situations à partir d'informations de contexte et diriger des adaptations dans le système en fonction des situations. Nous regroupons ces travaux en deux volets : *(1)* la modélisation et la gestion du contexte et *(2)* l'adaptation des systèmes. Cette thèse se situe dans le premier volet.

L'ambition de cette thèse est de **faciliter le développement des applications sensibles au contexte** en proposant des solutions intergicielles selon deux aspects : la définition de méta-données permettant la prise en compte de la qualité de contexte (QoC) et les mécanismes efficaces et performants qui doivent être intégrés au sein d'un gestionnaire de contexte.

Dans notre première contribution, nous identifions les critères de qualité pertinents selon les familles d'applications, nous proposons un **modèle extensible de méta-données** à associer aux informations de contexte et nous concevons des opérateurs pour calculer ces méta-données et pour évaluer la qualité de contexte.

Notre deuxième contribution s'inscrit dans une démarche d'ingénierie dirigée par les modèles afin de **fournir des outils logiciels** pour faciliter le développement des applications sensibles au contexte mais aussi à la qualité de ce contexte.

Nos contributions ont été intégrées à la plateforme libre COSMOS [5] [Conan et al., 2007a] conçue principalement par l'équipe MARGE [6] de Télécom SudParis, équipe d'accueil de cette thèse. Nos contributions ont ainsi pu être mises en œuvre et validées dans le cadre du projet FUI Cappucino [7] et ont permis la réalisation de deux prototypes : une application de détection de localisation en situation de mobilité et une application de vente Flash dans un centre commercial.

1.4 Organisation du manuscrit

Ce manuscrit se compose de trois parties. Dans la première partie, nous présentons l'état de l'art de la qualité de contexte pour l'intelligence ambiante. Nous commençons par définir, dans la section 2.2, les concepts manipulés et la notion de qualité de contexte puis nous faisons une étude détaillée de ses différents aspects dans la section 2.3. Dans la section 2.4, nous présentons les différentes méthodes de gestion de l'incertitude et de l'inférence du contexte. La première partie se termine par l'étude de la modélisation de la qualité de contexte dans les travaux de différentes équipes de recherche.

5. http ://picoforge.int-evry.fr/projects/svn/cosmos
6. http ://www-inf.it-sudparis.eu/MARGE
7. `http://www.cappucino.fr`

La deuxième partie est consacrée à notre contribution dans le domaine de la gestion de la qualité de contexte. Cette partie se compose de deux chapitres. Dans le chapitre 3, nous présentons nos travaux sur l'intégration de la gestion de la qualité de contexte dans l'intergiciel COSMOS dont la section 3.1 présente l'architecture. Les sections 3.2, 3.3 et 3.4 sont consacrées à la présentation de la contribution. La section 3.5, quant à elle, est dédiée à une nouvelle notion que nous proposons : la gestion fine de la qualité de contexte. Le chapitre 4 présente un processus d'ingénierie dirigée par les modèles pour la construction d'applications ubiquitaires sensibles à la qualité de contexte.

La dernière partie du manuscrit est consacrée à la validation de nos contributions. Nous présentons les résultats obtenus à partir d'une série de tests effectués sur des prototypes applicatifs utilisant notre intergiciel. Dans la section 5.1, nous présentons les résultats d'une application de détection de localisation. Dans la section 5.2, nous décrivons les résultats obtenus lors de la simulation d'un scénario de Vente Flash dans un centre commercial.

Nous finissons ce manuscrit par le chapitre 6 avec une synthèse des contributions et la discussion de perspectives à notre travail.

Première partie

Étude de l'état de l'art

Chapitre 2

Analyse de l'état de l'art sur la qualité de contexte pour l'intelligence ambiante

Sommaire

Ce chapitre dresse un panorama des différents travaux de recherche qui ont été proposés ces dernières années autour de la qualité de contexte. Afin de discuter des différents paradigmes et propositions qui apparaissent dans ces travaux, nous avons divisé ce chapitre en trois parties principales. Après une introduction des concepts manipulés et une définition des termes liés au domaine d'études, nous traitons en premier lieu de l'utilisation de méta-données associées

au contexte pour représenter la qualité de ce contexte. Nous considérons ensuite les différentes méthodes de gestion de l'incertitude et de l'inférence de contexte à partir de données incertaines et nous décrivons enfin les approches proposées pour la modélisation de la qualité de contexte. Nous terminons ce chapitre par une analyse et une synthèse des différents points présentés.

2.1 Introduction

L'informatique sensible au contexte vise à réduire la quantité d'informations explicites qu'un utilisateur doit fournir pour que le système accomplisse la tâche souhaitée. Ceci est particulièrement vrai dans le domaine de l'intelligence ambiante où les objets de la vie courante deviennent capables de déclencher une action ou un échange spontané d'informations, sans interaction avec l'utilisateur.

Les informations de contexte sont par nature imparfaites puisque d'une part elles résultent d'un modèle représentant une abstraction de la réalité ne pouvant être exhaustive et d'autre part elles dépendent de dispositifs tels que des capteurs ayant des limitations physiques intrinsèques. Les informations de contexte ont donc tendance à être *i*ncorrectes car elles ne reflètent pas exactement l'état réel de l'entité observée, *i*ncomplètes lorsque certains aspects du contexte sont manquants, ou encore *a*mbiguës si plusieurs valeurs sont collectées et ne correspondent pas entièrement entre elles [Henricksen, 2003]. Afin de ne pas mettre en péril les décisions qui en découlent, la qualité des informations de contexte doit être explicite et faire l'objet d'accords négociés entre les fournisseurs et les utilisateurs de contexte. En l'absence de tels accords, des décisions critiques concernant par exemple l'adaptation dynamique de l'application avec le changement de l'interface utilisateur pourraient être basées sur des informations non fiables et ne répondraient alors plus aux besoins des utilisateurs.

Il est donc important de concevoir les mécanismes nécessaires pour limiter l'impact que peuvent avoir ces imperfections, notamment en permettant au système de prendre en compte leur existence même. C'est ainsi que la qualité des informations de contexte devient primordiale pour les applications sensibles au contexte. Les systèmes sensibles au contexte existants ne traitent que succinctement des aspects qualité. [Baldauf et al., 2007] mentionne la nécessité pour un serveur de contexte de préciser les paramètres de qualité de service réseau, ce qui diffère de la QoC, et [Kristian Ellebæk Kjær, 2007] indique que parmi tous les intergiciels passés en revue, seul MiddleWhere [Ranganathan et al., 2004b] traite explicitement de la qualité de contexte, mais limitée aux informations de localisation. Des travaux de recherche prenant en compte la qualité de contexte dans le cadre de l'intelligence ambiante ont été menés ces dernières années ([Dargie and Hamann, 2006a, Kim and Lee, 2006b, Preuveneers and Berbers, 2007a, Ranganathan et al., 2004a, Razzaque et al., 2005, Tang et al., 2007b, Zimmermann, 2007, Manzoor, 2010, Becker et al., 2010, Yasar et al., 2011]) mais la proposition de solutions intergicielles flexibles et performantes reste indispensable comme indiqué par le récent article [Conti et al., 2012].

2.2 Définitions

Nous présentons un ensemble de définitions permettant de délimiter le domaine de recherche traité dans cette thèse. Nous commençons par l'informatique ubiquitaire, et continuons par la notion de contexte qui est au cœur de ces nouvelles formes d'informatique pour ensuite introduire le principe de sensibilité au contexte. Nous présentons enfin la dimension de la qualité liée à l'information de contexte, thème que nous développons plus en détail dans la suite de cette section et qui est central pour cette thèse.

Informatique Ubiquitaire Le mot *ubiquité* a pour origine *ubique* en Latin, qui signifie présent en plusieurs lieux simultanément. L'informatique ubiquitaire (ou omniprésente) est une informatique accessible de tout endroit et depuis toute sorte de dispositif. Elle représente un univers dans lequel il est possible de regrouper différents services fournis par les objets informatisés situés autour des utilisateurs, quelle que soit la localisation des utilisateurs.

Exemples : le téléphone portable peut interagir avec le réfrigérateur pour proposer des recettes avec les ingrédients disponibles. Le système peut même passer automatiquement commande des provisions nécessaires en se basant sur les produits manquants ou disponibles en faible quantité et les préférences de l'utilisateur.

Plusieurs variantes de l'informatique sont à rapprocher de l'informatique ubiquitaire. L'informatique mobile, incluant la mobilité utilisateur, la mobilité du terminal et la mobilité réseau, est caractérisée par un changement de contexte permanent. L'informatique diffuse (pervasive) correspond à un environnement saturé de capacités de calcul et de communication suffisamment bien intégrées pour devenir invisibles des utilisateurs [Satyanarayanan, 2001]. Finalement, l'informatique ambiante est présente dans un milieu ayant la faculté de percevoir, de raisonner, d'agir, et d'interagir afin de fournir des services améliorant la qualité de vie des êtres vivants et notamment des personnes [Coutaz and Crowley, 2008].

Contexte Il existe autant de définitions de la notion de contexte que d'équipes de recherche dans le domaine de l'informatique ubiquitaire.

La définition du *contexte* dans les premiers travaux où le mot *contexte* apparaît, se réfère à un ensemble décrit d'éléments : la *localisation*, les *identités* des personnes autour de l'utilisateur, des objets dans l'environnement physique et les différents changements ayant lieu dans cet environnement.

Schilit et al. [Schilit and Theimer, 1994] introduisent la notion de sensibilité au contexte (context-awareness) pour désigner un système doté d'un modèle du contexte. Étudier le contexte revient à repondre aux questions « Où suis-je ? », « Avec qui suis-je ? », « Quelles sont les ressources disponibles aux environs ? ». Il définit ainsi le contexte comme les changements de l'environnement physique, de l'utilisateur et des ressources de calcul. Pascoe [Pascoe, 1998] prolonge ce concept et définit le contexte comme un sous-ensemble d'états d'intérêts physiques et conceptuels à une entité particulière. Dey [Dey, 2001] propose une définition généralisée du contexte : « *Le contexte est toute information qui peut être utilisée afin de caractériser la situation d'une entité. Une entité peut être une personne, une place ou un objet que l'on considère pertinent à*

l'interaction d'un utilisateur et d'une application, incluant même l'utilisateur et cette application.»

En intelligence artificielle, [Brézillon, 2002] définit le contexte par comme étant « ce qui n'intervient pas directement dans la résolution d'un problème mais contraint sa résolution ».

Dans le cadre de l'informatique ubiquitaire, nous pouvons citer quelques exemples d'entités et de contextes associés. Le terminal utilisé auquel sont associés la bande passante de la connexion réseau, la mémoire disponible, la charge du processeur, la taille d'écran et les modes d'interaction avec l'utilisateur. L'utilisateur avec les informations sur l'identité de l'utilisateur, sa langue maternelle et sa localisation géographique. Une salle et le niveau de lumière ou de bruit et la température qui y règnent.

Sensibilité au contexte Un système est sensible au contexte s'il peut utiliser et interpréter les informations issues du contexte et adapter sa réponse en fonction du contexte d'utilisation [Schilit and Theimer, 1994].

[Brown, 1995] dit d'une application sensible au contexte qu'elle doit automatiquement extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par des capteurs.

[Salber et al., 1998] définit la sensibilité au contexte comme étant la meilleure capacité d'un système à agir avec des données provenant du contexte.

D'après [Abowd et al., 1998], un système est sensible au contexte s'il utilise ce contexte pour mettre à disposition de l'utilisateur des informations ou des services qui lui sont utiles. L'utilité dépend de la tâche de l'utilisateur.

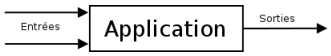


FIGURE 2.1 – Application appartenant à un système standard

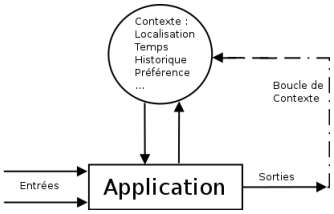


FIGURE 2.2 – Application sensible au contexte

[Lieberman and Selker, 2000] complète le schéma de la boîte noire d'un système informatique (cf. figure 2.1) en ajoutant des éléments de contexte qui vont pouvoir agir sur l'application. De même, il est possible pour l'application d'agir et de modifier le contexte. C'est ce que les auteurs appellent la *boucle de contrôle du contexte* où la sortie contextuelle de l'application va modifier les valeurs des variables contextuelles. (figure 2.2)

Qualité de Contexte La notion de qualité de contexte (ou QoC) a été abordée pour la première fois par [Buchholz et al., 2003] en tant que concept à part entière dans le domaine des applications sensibles au contexte. La QoC est définie comme étant *Toute information qui décrit la qualité d'une information utilisée dans le contexte*, et se différencie de la qualité de service (ou QoS) ou encore de la qualité des équipements (ou QoD). Ces travaux traitent la QoC au travers de méta-données attachées aux informations de contexte, qui sont l'objet de la section suivante.

2.3 Méta-données de qualité de contexte

Nous traitons dans cette section des travaux associant des méta-données représentant la qualité de contexte aux informations de contexte en elles-mêmes. Cette association est effectuée le plus souvent par le cadriciel de gestion de contexte au moment de la réception des données de contexte qui va alors calculer les valeurs des méta-données, mais certaines sources de contexte peuvent également fournir directement des méta-données en même temps que les données de contexte.

L'un des premiers articles évoquant la **Qualité de Contexte** est de [Buchholz et al., 2003]. Les auteurs définissent le concept de *qualité de contexte* (voir ci-dessus à la section 2.2) avec un ensemble de dimensions telles que *précision, probabilité de justesse, résolution et fraîcheur*. En effet, deux informations de contexte de même type reliées à une entité identique et obtenues de la même source de contexte peuvent être différentes en terme de leur précision, fiabilité, fraîcheur, etc.

Kim et al. [Kim and Lee, 2006a] étendent la liste des dimensions de la qualité de l'information avec ls paramètres de la QoC en incluant *l'exactitude, la complétude, la cohérence de la représentation, et la sécurité d'accès*. Krause [Krause and Hochstatter, 2005] tiennent également compte des caractéristiques de capteurs utilisés, la situation de mesure et la granularité de la représentation comme source de détermination de la QoC, et ajoute la notion de valeur dans l'information apportée par la qualité de contexte : « *Quality of Context is any inherent information that describes context information and can be used to determine the worth of the information for a specific application.* »

[Neisse et al., 2008] adopte comme référence pour le concept de qualité le vocabulaire proposé par l'*International Organization for Standardization* (**ISO**), un standard utilisé par les ingénieurs de métrologie et de mesure [ISO, 2004]. La comparaison effectuée entre les différentes définitions de la qualité de contexte dans la littérature est compatible avec les concepts proposés par l'**ISO**. Afin d'étudier les différents aspects de la qualité de contexte, cette partie est consacrée à explorer les études effectuées autour de la qualité de contexte. Nous présentons dans la section 2.3.1, les classifications des différents critères de qualité qui été proposées. Nous distinguons dans la section 2.3.2 la qualité des sources d'informations, telles que les capteurs fournissant des données brutes provenant de mesures faites sur l'environnement, des différents paramètres de qualité qui peuvent être obtenus à un plus haut niveau après transformation des données brutes et que nous présentons dans la section 2.3.3.

2.3.1 Critères de qualité et classification

Dans leurs travaux, Manzoor et al. [Manzoor et al., 2008, Manzoor, 2010, Manzoor et al., 2012] classent les entités de la qualité de contexte en deux groupes (Figure 2.3) : les *sources de la QoC* et les *paramètres la QoC*. Les **sources de QoC** comportent les informations de qualité collectées à partir de l'environnement ou extraites de la configuration du système. Les éléments appartenant à ce groupe représentent des informations de contexte brutes. Ce type d'information ne peut pas être utilisé directement par les applications sensibles au contexte, il faut donc que ces informations soient transformées en un niveau plus haut niveau d'abstraction. Ce sont les **paramètres de QoC**.

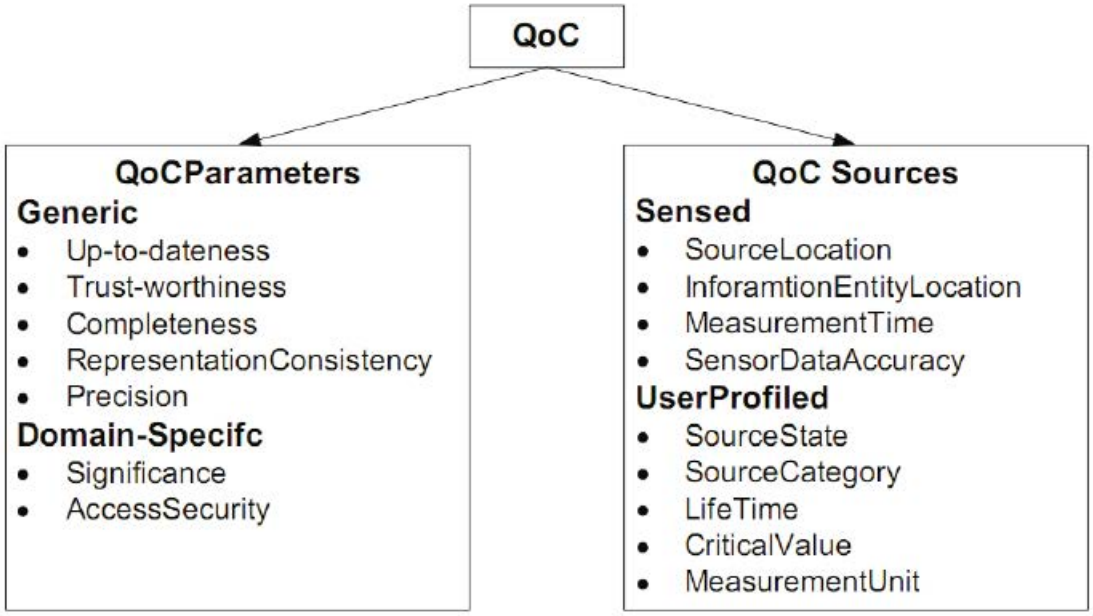


FIGURE 2.3 – Classification des critères de QoC en sources et paramètres, [Manzoor et al., 2008]

[Kim and Lee, 2006a] présente la relation entre les paramètres de qualité de contexte et les informations de contexte au travers des *dimensions de l'information de qualité*. Une classification est effectuée avec une approche similaire à celle de Manzoor. Les informations de qualité brutes sont appelées les *dimensions de l'information de qualité*, les paramètres du niveau supérieur sont appelés *paramètres de qualité de contexte*. La figure 2.4 montre que les auteurs classent certains critères de qualité dans les deux ensembles, c'est-à-dire, les critères qui appartiennent à l'intersection, peuvent jouer à la fois le rôle de *dimension de l'information* mais également celui de *paramètre de qualité*. Dans le cas de l'*incertitude*, un paramètre de qualité, défini par les auteurs comme la probabilité de correction, peut être une dimension pour générer la *confiance*.

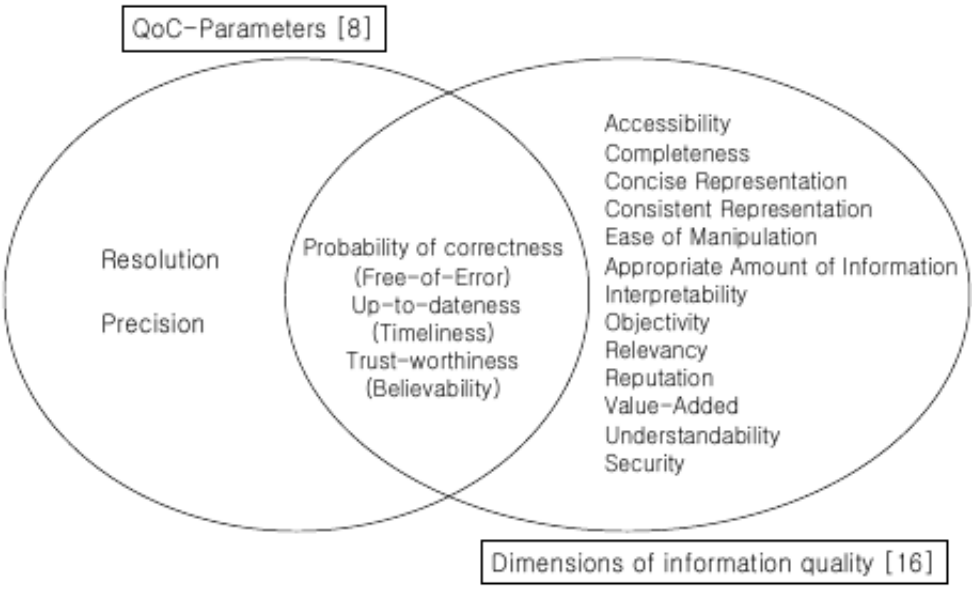


FIGURE 2.4 – Dimensions de la qualité de l'information et paramètres de qualité de contexte [Kim and Lee, 2006a]

2.3.2 Sources de qualité de contexte

SourceLocation Cela concerne le lieu où est située la source de l'information du contexte. Cette localisation peut être absolue (des coordonnées GPS, une adresse, etc.) ou relative (une distance, coordonnées par rapport à un objet, etc.).

InformationEntityLocation Cela indique la localisation de l'entité observée. Avec la *SourceLocation*, cette information construit *l'espace de résolution* de l'information de contexte, regroupant les informations relatives à l'entité observée et à la source d'observation. Par exemple, cela permet de calculer la *distance* entre la source de contexte et l'entité observée, qui est ensuite utilisée notamment pour calculer le paramètre de *confiance* (voir 2.3.3.6).

MesurementTime C'est la date à laquelle l'information de contexte est mesurée. Cette information est utilisée comme estampille pour calculer le paramètre de *fraîcheur*.

SensorDataAccuracy Cela correspond à la capacité du capteur à mesurer une valeur du contexte proche de la valeur réelle. Cette information est déterminante pour le calcul de la *précision* du contexte.

SourceState Indique si la source de l'information est dynamique (par exemple un GPS dans un téléphone mobile) ou statique (un capteur de température dans une pièce).

SourceCategory Représente la catégorie de la source du contexte qui peut être capturée, dérivée ou provenant d'un profil utilisateur.

Paramètres de qualité	Sources de QoC
Fraîcheur	Date de la mesure, Date actuelle
Complétude	Nombre d'attributs collectés, Nombre d'attributs total.
Exactitude	Étalonnage ou spécifications des capteurs
Précision	Étalonnage ou spécifications des capteurs
Résolution	Unité de mesure ou spécifications des capteurs
Confiance	Localisation de la source, Exactitude, Localisation de l'entité observée

TABLE 2.1 – Relation entre sources de qualité et paramètres de qualité [Manzoor et al., 2008]

LifeTime C'est l'intervalle de temps à partir duquel l'information de contexte est considérée comme obsolète. Après l'écoulement de cette durée, il est nécessaire de prendre une nouvelle mesure pour obtenir une valeur valide.

MesurementUnit C'est l'unité de la valeur de l'information dans laquelle est mesurée la valeur de l'information. Cela influence la *précision* des informations de contexte.

2.3.3 Paramètres de qualité de contexte

Dans la littérature, la limite est étroite entre les sources de qualité et les paramètres de qualité. Tout d'abord, il n'y a pas un consensus préalable sur le vocabulaire. La définition de la précision concorde avec la gestion et la correction d'erreur, l'exactitude a un sens proche de rigueur, la fraîcheur est liée à la vivacité et de même pour la fidélité et la confiance.

Les travaux initiaux [Buchholz et al., 2003] et [Kim and Lee, 2006a] ont défini plusieurs paramètres dans le cadre des services sensibles au contexte. Ce sont encore aujourd'hui les paramètres les plus utilisés et ils sont présents dans différents domaines d'applications de l'information ubiquitaire. Parmi ces paramètres, nous pouvons compter la *complétude*, la *précision*, l'*exactitude* et la *confiance*.

La possibilité d'ajouter des paramètres spécifiques au domaine d'application n'est pas exclue. Parmi ces paramètres, nous pouvons citer *la sécurité d'accès*, la *valeur ajoutée*, l'*objectivité*, la *sensibilité*, la *résolution*, l'*incertitude*, etc.

Dans cette section, différentes définitions de paramètres sont présentées ainsi que les méthodes utilisées pour générer les valeurs des paramètres dans les différents domaines de gestion de contexte. En général, les valeurs des paramètres calculés appartiennent à l'intervalle $[0, 1]$.

La table 2.1 montre la relation entre les paramètres de qualité et les informations de qualité liées aux sources d'informations que nous venons de présenter à la section 2.3.2. Nous décrivons plus en détail chacun des paramètres de QoC mentionnés ci-après.

2.3.3.1 Fraîcheur

La fraîcheur correspond au «*temps qui s'écoule entre la détermination de l'information de contexte et sa livraison à la destination*». C'est une qualité qui caractérise l'importance des

informations de contexte par rapport à création ou la mesure des informations de contexte. La *fraîcheur* est définie par des termes différents : les travaux de [Preuveneers and Berbers, 2007a] utilisent le terme *timeliness*, mais [Manzoor et al., 2008] traite plutôt de *Up-To-Dateness*.

En général, la fraîcheur est générée en attribuant une estampille à l'information de contexte indiquant la date de génération. Un mécanisme de synchronisation d'horloges entre la source de l'information de contexte et son utilisateur doit alors être utilisé.

Calcul de la fraîcheur [Manzoor et al., 2008] propose de calculer la fraîcheur en fonction de l'âge d'une information de contexte et de sa durée de vie.

Soit $\mathcal{O}$ un objet à l'étude. L'âge de $\mathcal{O}$ est la différence entre la date courante t_{curr} et la date d'estampillage de la mesure $t_{measure}$.

$$Age(\mathcal{O}) = t_{curr} - t_{measure} \tag{2.1}$$

La fraîcheur $\mathcal{F}$ est proportionnelle au rapport de l'âge du contexte et la valeur de sa durée de vie (*lifetime*) qui représente l'âge maximal accepté pour valider le contexte. L'équation 2.2 indique comment calculer la fraîcheur :

$$\mathcal{F}(\mathcal{O}) = \begin{cases} 1 - \dfrac{Age(\mathcal{O})}{Lifetime(\mathcal{O})} & \text{Si} Age(\mathcal{O}) < Lifetime(\mathcal{O}) \\ 0 & \text{Sinon} \end{cases} \tag{2.2}$$

La valeur de la fraîcheur est maximale si la mesure vient juste d'être effectuée et la valeur directement acheminée au demandeur de l'information, l'âge du contexte est donc proche de zéro. La valeur diminue si l'âge augmente, l'écart entre la date de mesure et la date de réception augmente jusqu'à ce qu'il devienne supérieur à la durée de vie. La valeur de la fraîcheur à ce moment-là est nulle.

2.3.3.2 Complétude

La *complétude* est le rapport entre la quantité d'informations présentes dans le contexte et la quantité totale qui peut être fournie dans le contexte. [Kim and Lee, 2006a] définit également la *complétude* comme « *Le degré avec lequel l'information de contexte est présente*». [Manzoor et al., 2008] définit ce paramètre de qualité comme «*la quantité d'informations fournie par l'objet du contexte*».

La complétude peut indiquer si certaines informations sont manquantes dans le contexte. Le manque d'information est un élément important dans la fiabilité de l'objet de contexte. Ce paramètre peut être déterminant pour le choix d'un objet de contexte parmi d'autres contextes de sources différentes.

Calcul de la complétude La complétude est calculée dans [Kim and Lee, 2006a] avec l'équation 2.3 où AD est le nombre d'attributs fournis et TD est le nombre total d'attributs.

$$\mathcal{C} = \frac{AD}{TD} \tag{2.3}$$

[Manzoor et al., 2008] raffine le calcul de la complétude en utilisant des poids pour traduire la valeur relative des attributs les uns par rapport aux autres. La complétude est donc le résultat d'un rapport entre la somme pondérée des attributs fournis et la somme pondérée totale des attributs comme le montre l'équation 2.4.

$$\mathcal{C} = \frac{\sum_{j=0}^{m} w_j(\mathcal{C})}{\sum_{i=0}^{n} w_i(\mathcal{C})} \tag{2.4}$$

m représente le nombre des attributs pondérés fournis par le contexte, $w_j(\mathcal{C})$ est le poids associé à l'attribut m et n est le nombre total d'attributs de contexte. La valeur de la complétude sera maximale si $n = m$, c'est-à-dire, si tous les attributs de contexte sont fournis.

2.3.3.3 Exactitude

[Kim and Lee, 2006a] définit l'*exactitude* comme «*le degré de représentation des informations de contexte par rapport à la situation réelle*». Elle décrit la façon dont la valeur de l'information de contexte décrit la réalité de la source de contexte au moment de la capture.

D'après l'**ISO**) [ISO, 2004], l'exactitude ne peut pas être exprimée en valeur numérique proprement dit. C'est plutôt la mesure de l'inexactitude ou le pourcentage d'erreur qui détermine la notion d'exactitude.

Calcul de l'exactitude Il est difficile de percevoir la valeur réelle d'une information sans un mécanisme de vérification comme une validation effectuée par l'Homme : par exemple un étalonnage ou une comparaison avec un appareil de référence, etc.

[Kim and Lee, 2006a] estiment au départ l'*intervalle de confiance* d'une information de contexte générée par un capteur en utilisant une méthode d'estimation statistique. Puis si la valeur d'information de contexte appartient à cet intervalle de confiance, ils considèrent qu'elle est bien correcte, donc ayant une *exactitude* maximale. Dans le cas de données continues, ils utilisent un intervalle confiance qui délimite la valeur minimale et la valeur maximale considérées comme correctes.

2.3.3.4 Précision

La *précision* est le degré de différence des valeurs capturées au cours de plusieurs mesures successives d'une information de contexte décrivant un objet de contexte.

La *précision* peut être exprimée à l'aide d'un intervalle $[-\epsilon, \epsilon]$, $\pm\epsilon$ ou bien elle peut être implicitement dépendante de l'unité de mesure. Si la valeur d'une distance est exprimée en

`mètres`, implicitement la précision est de $\pm 1m$ à moins que le fournisseur exprime explicitement la précision, correspondant généralement à la précision du *capteur de mesure*.

[Preuveneers and Berbers, 2007a] indique que la précision influence le paramètre de l'*exactitude* car la précision indique la granularité de l'information mesurée.

Relation et différence entre précision et exactitude Les deux qualités *précision* et *exactitude* sont souvent liées même si l'application sensible au contexte a plus d'intérêt pour l'une que pour l'autre. Afin d'étudier la relation entre *précision* et *exactitude*, l'étude de la température ambiante d'une pièce sera utilisée.

La figure 2.5 montre des représentations graphiques de mesures de la température ambiante effectuées dans une pièce par plusieurs capteurs. La *température réelle* de la pièce est de $25^{\circ}C$. Elle correspond au cœur des quatre cibles de la figure. On considère qu'une erreur de mesure est acceptable si elle ne dépasse pas $1^{\circ}C$, les valeurs tolérées sont donc dans l'intervalle $[24^{\circ}, 26^{\circ}]$.

Les résultats sont représentés sur la figure 2.5 de gauche à droite. Pour le premier capteur, les mesures fournies sont à $25^{\circ}C, \pm 0.2^{\circ}$, les mesures sont donc considérées comme exactes car elles sont proches de la valeur réelle de la température, mais également précises car la différence entre les différentes mesures est petite $\pm 0.2^{\circ}$. Les mesures du deuxième capteur avoisinent $27^{\circ}C, \pm 0.3^{\circ}$. Elles sont donc précisent vu l'écart entre les valeurs. Mais les prises restent éloignées de la valeur réelle de température, donc, elles ne sont pas exactes. De même, les mesures du troisième capteur $(25^{\circ}C, \pm 1^{\circ})$ sont tolérées en raison du non dépassement la valeur maximale, mais la différence entre les prises est considérable, les mesures ne sont donc pas précises. Le dernier capteur fournit des mesures éparpillées qui ne sont ni précises ni exactes.

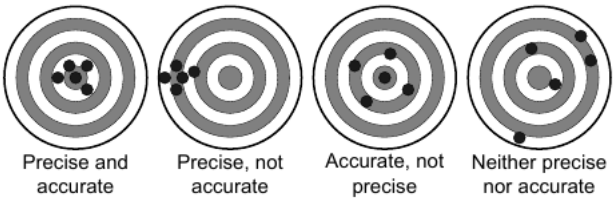


FIGURE 2.5 – Différence entre précision et exactitude[Neisse et al., 2008]

2.3.3.5 Résolution

[Neisse et al., 2008] définit le concept de la *résolution* comme étant la granularité que peut avoir la caractéristique du contexte concerné. Elle peut être définie également comme une limite supérieure (ou inférieure) à partir de laquelle la source de l'information ne peut plus donner un niveau supérieur (ou inférieur) de détails. Les auteurs distinguent deux types de résolution : la *résolution temporelle* et la *résolution spatiale*. La résolution temporelle est définie comme « *la période de temps pendant laquelle une seule instance de contexte est applicable*», la résolution temporelle peut être définie également comme *la meilleure approximation de l'intervalle de temps*

pendant lequel le contexte est déterminé. La résolution est mesurée par une unité significative (année, mois, jour, heure, minute). La *résolution spatiale* est définie comme «*l'expression de la précision de l'environnement physique par rapport à l'information de contexte*». C'est à dire la résolution du lieu dans lequel se trouve l'entité étudiée.

Exemples :

GPS La résolution spatiale minimale d'un GPS est de `20 mètres` c'est-à-dire que l'appareil ne distingue plus les déplacements de l'entité en-dessous du seuil de `20 mètres`.

Rendez-vous Un rendez-vous noté dans le profil de l'utilisateur d'un téléphone mobile est de la forme `heure de début - heure de fin`. La résolution du calendrier diffère d'un logiciel à un autre. Le seuil minimal ne dépasse généralement pas les `5min` ; c'est-à-dire que l'utilisateur ne peut pas choisir un début de rendez-vous à `15h02`, il aura le choix entre `15h` ou `15h05`.

2.3.3.6 Confiance - Fidélité

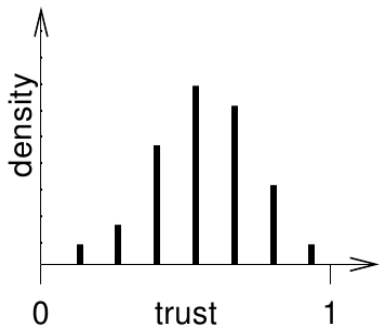
Définition [Neisse et al., 2008] définissent la confiance (*trustworthiness*) comme « *une mesure subjective de la croyance d'une entité à propos du comportement d'un aspect particulier d'une seconde entité*». La confiance est définie également par la mesure de l'exactitude de l'information de contexte, c'est-à-dire de la fidélité entre les mesures et la valeur réelle du contexte.

Différentes représentations de la confiance La forme de modélisation la plus simple pour la confiance est l'ensemble booléen dans lequel la confiance ne peut avoir que deux valeurs possibles : $T = \{Vrai, Faux\}$. La confiance peut être modélisée par une échelle *discrète* plus détaillée comme dans le projet PGP/GnuPG [Ashley et al., 1999] (implémentant le modèle du *web de confiance* [Khare and Rifkin, 1998]) avec plusieurs valeurs possibles de confiance {`non fiable, confiance limitée, confiance totale` } ({`untrustworthy, marginal trust, full trust`}) ou par une fonction de distribution discrète comme dans le cas de la figure 2.6a.

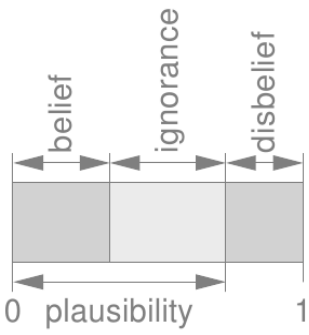
La confiance peut être modélisée par une échelle continue comme proposée dans [Marsh and Science, 1994] sur un intervalle $[-1, 1]$. Elle peut être représentée également par une résolution de la mesure de la distance entre le capteur et l'entité observée ou par le ratio entre différents acteurs : la résolution spatiale par rapport à distance maximale tolérée, la précision de la mesure et distance maximale. Pour tout le contexte, la confiance est donc liée aux différentes sources d'information utilisées par le fournisseur de l'information lui-même. Certains travaux représentent la confiance en introduisant la notion d'*incertitude*. Dempster et Shafer [Shafer, 1976] présentent la confiance comme étant des limites :

$$0 \leq belief < plausibility \leq 1$$

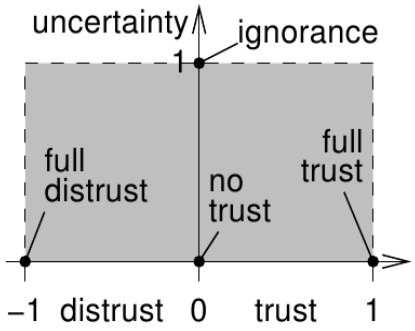
Jøsang [Jøsang, 1997] présente une approche similaire avec trois notions : *belief* (b), *disbelief* (d), et *ignorance* (i) avec $b, d, i \in [0, 1], b + d + i = 1$. La Figure 2.6b, présente la similitude entre l'approche de Jøsan et de Dempster-Shafer.



(a) Représentation de la confiance à échelle discrète



(b) Représentation de la confiance dans les travaux de [Jøsang, 1997] et de [Shafer, 1976]



(c) Représentation de confiance à partir d'autres paramètres de qualité

FIGURE 2.6 – Différentes approches pour représenter la confiance [Becker et al., 2010]

[Becker et al., 2010] montre que la confiance peut être générée au travers d'une ou plusieurs notions : la relation de *réputation*, est une liaison d'une personne avec ses connaissances par relation directe, mais par propagation, elle permet également de générer la réputation résultante avec d'autre personnes qui n'appartiennent pas à l'entourage de la personne de départ. On en déduit donc la valeur de confiance résultante. La figure 2.7 représente un exemple de cas.

Une autre approche consiste à exprimer la confiance à l'aide d'autres paramètres de contexte et notamment d'autres paramètres de qualité de contexte. Par exemple, la confiance 2.5c peut résulter de la combinaison de la relation de la fidélité (`trust`) des informations de contexte définie par l'équation 2.5a et l'incertitude (`uncertainty`) définie par l'équation 2.5b.

(2.5a) $$\forall x \in \mathbb{R}; F(x) \in [-1, 1]$$

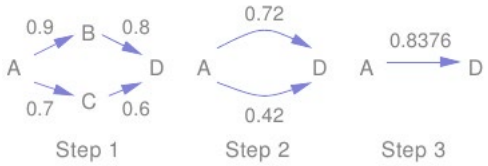


FIGURE 2.7 – Calcul de la confiance résultante de deux personnes sans relation préalable [Becker et al., 2010]

(2.5b) $$\forall y \in \mathbb{R}; U(y) \in [0,1]$$

(2.5c) $$Trust : \left| \begin{array}{rcl} [-1,1] \times [0,1] & \longrightarrow & [-1,1] \\ t,u & \longmapsto & T \end{array} \right.$$

La figure 2.6c est une représentation spatiale de l'équation 2.5c. On peut distinguer sur cette figure quelques points particuliers. Les points de l'axe des abscisses appartiennent à des informations de contexte sûres dans le sens où la certitude de la correction des informations est maximale. Donc les points de coordonnées $x_1(-1,0)$; $x_2(0,0)$; $x_3(1,0)$ représentent respectivement les notions de {`méfiance maximale`, `confiance neutre`, `confiance maximale`}.

Pour les points sur l'axe $U_{max} = \forall A(x_A, y_A) \in T / y_A = 1$ on distingue que les deux points particuliers sur l'axe U_{max} sont de première apparence ambigus, mais ils ne sont pas pour autant contradictoires. Le point $A_1(1,1)$ représente le cas d'une source de `confiance` mais avec des informations non correctes. Quant au point $A_2(-1,1)$, il représente une source de `méfiance` avec des informations de contexte non correctes. Le point $A_0(0,1)$ sur l'axe U_{max} s'interprète donc par une `confiance` nulle et `incertitude` maximale. Ce cas s'applique sur les informations de sources inconnues avec une qualité de correction nulle.

2.3.4 Synthèse

Dans cette section, nous avons exposé différents aspects de la qualité de contexte. Le premier aspect concerne la définition de la qualité de contexte. Les définitions des différentes notions de qualité sont orientées par rapport à la source de contexte selon qu'elle soit physique, telle qu'un capteur fournissant des mesures brutes, ou bien qu'elle soit de plus haut niveau impliquant une transformation de ces données brutes.

Le deuxième aspect est la distinction de plusieurs notions de qualité les unes par rapport aux autres : la *précision*, l'*exactitude*, la *complétude* et la *confiance* se retrouvent dans la plupart des travaux mais sous différentes appellations et avec des méthodes de calcul différentes.

Nous distinguons également l'aspect d'orientation des critères de qualité vers les sources ou fournisseurs de contexte, ou bien vers les demandeurs ou consommateurs des informations de

contexte ce qui engendre des dépendances entre les critères de qualité de contexte. Par exemple, la *confiance* peut être calculée à partir de la fraîcheur ou de l'incertitude. La *confiance* peut être calculée comme étant une qualité abstraite, elle n'est pas liée directement aux sources des informations utilisées pour la générer. Elle répond plus au besoin de l'application sensible au contexte, le consommateur. Cette orientation amène à penser à la gestion des flux de la qualité depuis les sources et à son acheminement jusqu'aux consommateurs, mais également à la manière avec laquelle on peut analyser la qualité et traiter différentes informations venant de sources différentes, en gérant l'incertitude inhérente à ces informations. Nous essayons de répondre à ces questions dans la section suivante.

2.4 Gestion de l'incertitude et de l'inférence de contexte

Dans la section précédente, nous avons étudié la qualité de contexte sous plusieurs formes : les sources de qualité et les paramètres de qualité qui se révèlent de bas ou de haut niveau. Afin de bénéficier de ces méta-données de qualité dans la gestion du contexte il faut en maîtriser l'analyse, la fusion, la sélection des unes par rapport aux autres et l'acheminement vers les demandeurs de ces informations, à savoir, les applications sensibles au contexte.

L'incertitude et l'imperfection inhérentes aux informations de contexte peuvent être traitées par différentes méthodes mathématiques, que nous présentons à la section 2.4.1. Une autre approche consiste à exploiter la synergie pouvant exister entre des informations contextuelles issues de sources multiples de telle façon que l'information combinée qui en résulte se révèle en quelque sorte meilleure que les différentes informations prises individuellement [Nakamura et al., 2007]. Nous discutons ainsi à la section 2.4.2 des méthodes les plus répandues pour la fusion et l'inférence de contexte.

2.4.1 Méthodes de gestion de l'incertitude

Dans la littérature, des méthodes de gestion de l'incertitude sont utilisées lorsque les informations de contexte sont incomplètes ou imprécises [Roy et al., 2009],[Frank et al., 2010]. Nous présentons dans cette section les méthodes mathématiques le plus souvent mises en œuvre à savoir les réseaux bayésiens, les chaînes de Markov cachées, la logique floue et les systèmes logiques.

Réseaux Bayésiens Les *réseaux Bayésiens* ont l'avantage d'inférer des résultats à partir d'un schéma complexe et de rassembler un groupe de contextes primitifs en un contexte abstrait de niveau supérieur. Cependant, la conception probabiliste de cette méthode dépend de certaines contraintes : elle doit obéir au théorème de Bayes [Bayes, 1763], respecter la propriété locale de Markov, élaborer une initialisation des probabilités des états, mais surtout le coût d'étude des sources multiples est élevé. Cette méthode est utilisée essentiellement pour les analyses et la manipulation des données de contexte des couches hautes. Les réseaux Bayésiens sont utiles pour faire face aux systèmes complexes, par contre ils engendrent un coût important.

Chaînes de Markov cachées [Preuveneers and Berbers, 2008] utilise les *Chaînes de Markov Cachées* (ou HMM) pour leur capacité à prévoir les états des différentes parties d'un système à partir des états précédents. Elles permettent la prédiction de contextes simples avec des états définis mais leur efficacité dépend de la complexité du système étudié.

Logique floue La *Logique floue* est l'évolution de la logique classique. Elle repose essentiellement sur *une fonction d'appartenance*. Elle définit l'appartenance d'un élément à un ou plusieurs ensembles, même si c'est pour un ensemble et son complément en même temps. [Haghighi et al., 2008] et [Korpipaa et al., 2003] utilisent la logique floue pour son efficacité sur les données primitives des sources ; elle atténue la difficulté d'interprétation des valeurs intermédiaires et permet d'enrichir les états du système sans un grand coût. Cependant, elle est moins efficace pour les manipulations des couches hautes comme l'agrégation de différents contextes.

Systèmes logiques Les systèmes logiques à base de règles s'appuient sur des *équations logiques* qui forment les règles. L'ensemble de ces règles forme à son tour une base de connaissances. Un moteur d'inférence, composé d'un ensemble d'actions, permet de dériver un contexte de haut niveau à partir de données issues de capteurs. Cette méthode est utilisée pour permettre l'association de composants à des objets de l'environnement réel ou à l'élaboration de politiques de gestion associées aux situations du monde réel. Les règles sont généralement issues des techniques de génie logiciel et de conception des architectures systèmes réparties : [Conan et al., 2007b, Rouvoy et al., 2008], [Dey and Abowd, 2000], [Chaari et al., 2007]. Ces techniques sont idéales pour les petits systèmes ou de taille moyenne.

Le tableau 2.2 donne une vision générale des différentes techniques étudiées dans cette section selon les aspects suivants :

- gestion de contexte brut : données primitives
- agrégation, extraction (filtrage) de contexte
- analyse et manipulation du contexte : contexte de haut niveau, données incomplètes
- système complexe : capacité à traiter un système complexe, comportant une quantité importante d'informations à traiter
- évolution de la méthode : facilité de configuration et d'initialisation
- complexité des algorithmes

La notation par + et − traduit la plus ou moins bonne prise en compte de ces aspects par chacune des méthodes.

2.4.2 Méthodes de fusion et d'inférence

Dans un environnement sensible au contexte, il peut exister plusieurs sources qui diffusent des informations de contexte ainsi que la qualité correspondante. Afin d'avoir une vue globale de la situation de contexte, il devient donc nécessaire de fusionner ces différentes informations [Nakamura et al., 2007]. En exploitant la synergie et les liens pouvant exister entre

Aspect / Technique	Logique Floue	Réseaux Bayésiens	HMM	Règles logiques
Données primitives	+++	-	- - -	+++
Agréger des données	-	+	-	++
Analyse et manipulation	- -	+++	+++	+
Système complexe	-	+++	+	++
Évolution	+	+	-	+
Complexité algorithmique	+	-	+	+

TABLE 2.2 – Comparaison des méthodes de gestion d'incertitude

les informations disponibles, les techniques de fusion permettent de faire des déductions par inférence sur le comportement d'une entité observée. L'inférence des informations de contexte devient alors un défi auquel, non seulement la gestion de contexte est confrontée, mais également la gestion de la qualité de contexte elle-même. Il s'agit en effet de déterminer la qualité de l'information de contexte obtenue en sortie et résultant de la fusion de plusieurs informations de contexte en entrée.

Nous considérons dans cette partie tout d'abord les méthodes de fusion non probabilistes qui sont les plus simples puis, nous traitons plus particulièrement de la théorie des croyances souvent utilisée pour la combinaison de données issues de sources multiples.

2.4.2.1 Méthodes de fusion non probabilistes

Parmi les méthodes de fusion non probabilistes, nous pouvons citer l'agrégation d'un ensemble de données par calcul de la *moyenne*. Elle peut être appliquée si les sources de chaque information de départ sont cohérentes et homogènes afin de permettre à l'information résultante d'hériter des mêmes caractéristiques (qualité). Une forme plus souple consiste à utiliser une moyenne pondérée tenant compte d'un poids relatif associé à chacune des sources.

Une autre forme simple de fusion met en œuvre un mécanisme de *vote*. Dans le cas où les sources proposent au système de gestion de contexte plusieurs valeurs, le système choisit l'une des valeurs candidates. Le vote peut être *majoritaire* où la valeur est acceptée à partir d'un nombre de votes supérieur ou égal à la moitié du nombre total des sources qui élisent cette valeur. Pour le vote avec *majorité absolue*, le nombre de votes requis est généralement les deux tiers du nombre total. Le vote peut être également pondéré pour donner une plus grande importance à certaines sources ou au contraire minimiser leur rôle dans la procédure d'élection. Les facteurs de pondération sont divers et dépendent de l'environnement et de l'application. Les facteurs peuvent être la distance de la source (le capteur) par rapport à l'entité observée mais également l'une des qualités relatives à la source comme la précision, la confiance, l'exactitude, etc. [Hall and McMullen, 2004], [Atrey et al., 2010].

2.4.2.2 Théorie des croyances

La théorie des croyances de Dempster-Shafer [Shafer, 1976] est utilisée pour combiner des informations différentes afin de calculer la probabilité d'un événement ; c'est une forme de généralisation des règles de Bayes. [Wu, 2004], [Ricquebourg et al., 2008], [Beamon and Kumar, 2010], [Nzekwa et al., 2010], [McKeever et al., 2009b, McKeever, 2011] utilisent la méthode de *Dempster-Shafer* afin de bénéficier de l'agrégation des données provenant de plusieurs sources telles que des capteurs différents. Cette méthode est bien adaptée au traitement de l'information de contexte des couches abstraites de la gestion de contexte mais son coût est important en termes de temps de calcul.

a) De l'hypothèse à la fusion

Soit Θ l'ensemble des hypothèses possibles appelé **cadre de discernement** et contenant N éléments H_i, $i \in \{1, \ldots, N\}$ exclusifs et exhaustifs.

Masse élémentaire Soit $A_i \in 2^\Theta$. Une masse élémentaire **m()** est une application de 2^Θ (l'ensemble des parties de Θ) dans $[0, 1]$ vérifiant les deux propriétés suivantes :

$$\begin{cases} m(\emptyset) = 0 \\ \sum\limits_{A_i \in 2^\Theta} m(A_i) = 1 \end{cases} \tag{2.6}$$

Les éléments $A_i \in 2^\Theta$ tels que $m(A_i) \neq 0$ sont appelés les *éléments focaux*. Dès lors que l'ensemble des éléments focaux se réduit aux seuls singletons H_i du cadre de discernement, la masse élémentaire $m()$ est assimilable à une probabilité $P()$

Fonctions de crédibilité et de plausibilité Les sources d'information attribuent des *crédibilités* aux hypothèses H_i présentées pour le schéma de raisonnement.

La **crédibilité** $(Cr(B_j))$ représente la *masse minimale* qu'il est possible d'affecter à une partie B de 2^Θ. La **plausibilité** $(Pl(B_j))$ représente à l'inverse la *masse maximale*.

$$Cr(B_j) = \sum_{A_i \subset B_j} m(A_i) \tag{2.7}$$

$$Pl(B_j) = \sum_{A_i \cap B_j \neq 0} m(A_i) \tag{2.8}$$

En outre, quel que soit $A \in 2^\Theta$, la somme de la *plausibilité* de A et de la *crédibilité* de $\bar{A}$ est toujours égale à 1 (2.9).

$$Pl(A) + Cr(\bar{A}) = 1 \tag{2.9}$$

Règle de combinaison Chaque source d'information est l'origine d'un jeu de masses élémentaires noté m_i. La combinaison d'évidences de deux sources indépendantes est accomplie par la *règle de combinaison* de Dempster :

$$m_{12}(A) = \frac{\sum_{\forall X,Y;X\cap Y=A} m_1(X) \cdot m_2(Y)}{1-K} \tag{2.10}$$

Où K est l'*incohérence de fusion* :

$$K = \sum_{\forall X,Y;X\cup Y=\emptyset} m_1(X) \cdot m_2(Y)$$

b) Théorie des croyances et inférence de la confiance

[Becker et al., 2010] présente une méthode de génération de la confiance résultante basée sur la `recommandation` [Gutscher, 2007]. Pour la relation de recommandation de confiance, il existe une limite h pour déterminer la longueur de la chaîne de confiance à construire (les sauts de recommandation). Une relation de recommandation de confiance avec une limite $h = 1$ n'engage le fournisseur de confiance que sur ses `relations de premier ordre` (connaissances directes), alors que pour une limite $h = 2$, la relation de confiance se propage sur deux sauts. Ainsi est dérivée la construction d'une confiance résultante à partir d'une chaîne de confiance (en assumant que le saut conserve toujours le même notion de confiance) :

1. La relation de confiance de A vers B avec un saut $h = 1$ peut être combinée avec la relation de confiance de B vers C en multipliant la valeur de confiance de la première relation par la deuxième (2.11)

$$t_r = t_1 t_2 \tag{2.11}$$

2. La fusion de deux relations parallèles de A vers B avec des valeurs de confiance t_1 et t_2 respectivement se calcule avec l'équation 2.12.

$$t_r = t_1 + t_2 - t_1 t_2 \tag{2.12}$$

Soit un ensemble de personnes `A,B,C,D;` muni de la relation R de confiance. Afin de déterminer la confiance que peut accorder `A` à `D`, les équations d'inférence (2.11, 2.12) sont utilisées de la manière suivante. La première étape consiste à déterminer les relations entre les différentes personnes dans le groupe. Par exemple :

- R_1 la relation entre A et B, $R_1 = 0.9$
- R_2 la relation entre A et C, $R_2 = 0.7$

– R_3 la relation entre B et D, $R_3 = 0.8$
– R_4 la relation entre C et D, $R_4 = 0.6$

La deuxième étape consiste à calculer l'inférence des relations de confiance R_1 et R_2 qui donne la relation résultante $R_{12} = 0.72$. De même, celle de R_3 et R_4 donne la relation $R_{34} = 0.42$. Les relations R_{12} et R_{34} représentent la confiance de A vers D à travers B et C. L'étape finale permet d'obtenir à partir des deux relations la relation de confiance résultante $R = 0.8376$.

Conjonction, disjonction et négation Lorsque la représentation de la confiance est celle de Dempster-Shafer [Shafer, 1976] et de Jøsang [Jøsang, 1997] ($t = (b, i, d)$) alors la *conjonction*, la *disjonction* et la *négation* sont proposées par Jøsang [Jøsang, 1997] et par Baldwin [Baldwin, 1987] comme le montrent les équations 2.13a, 2.13b et 2.13c. Pour rappel, b est le symbole de la *croyance* (`belief`), i le symbole de l'*ignorance* (`ignorance`), et d le symbole de *plausabilité* (`disbelief`).

Les tableaux 2.3a, 2.3b et 2.3c sont les tableaux de vérité respectivement de *conjonction*, de *disjonction* et de *négation* avec les symboles : $+$ pour désigner b, $\varnothing$ pour i et $-$ pour d.

(2.13a)
$$t_x \wedge t_y = \begin{pmatrix} b_x b_y \\ i_x i_y + i_x b_y + b_x i_y \\ d_x + d_y - d_x d_y \end{pmatrix}$$

(2.13b)
$$t_x \vee t_y = \begin{pmatrix} b_x + b_y - b_x b_y \\ i_x i_y + i_x d_y + d_x i_y \\ d_x d_y \end{pmatrix}$$

(2.13c)
$$\neg t_x = \begin{pmatrix} d_x \\ i_x \\ b_x \end{pmatrix}$$

$\wedge$	$+$	$\varnothing$	$-$
$+$	$+$	$\varnothing$	$-$
$\varnothing$	$\varnothing$	$\varnothing$	$-$
$-$	$-$	$-$	$-$

(a) conjonction

$\vee$	$+$	$\varnothing$	$-$
$+$	$+$	$+$	$+$
$\varnothing$	$+$	$\varnothing$	$\varnothing$
$-$	$+$	$\varnothing$	$-$

(b) disjonction

$\neg$	
$+$	$-$
$\varnothing$	$\varnothing$
$-$	$+$

(c) négation

TABLE 2.3 – Tables de vérité des opérateurs *conjonction*, *disjonction* et la *négation*

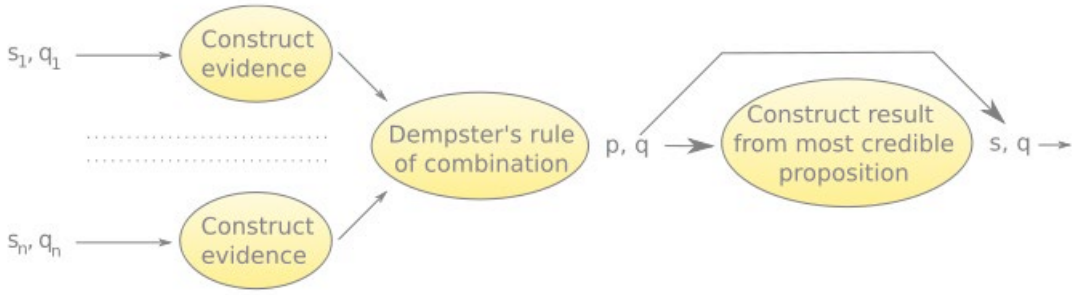


FIGURE 2.8 – Fusion de la confiance à l'aide de la théorie des croyances dans les réseaux de capteurs sans fils (WSN) [Frederik Hermans, 2009]

Consensus L'opérateur consensus ($\oplus$) combine la valeur de confiance de deux relations de confiance qui font référence à la même proposition. Jøsang [Jøsang, 1997], Dempster-Shafer [Shafer, 1976] et Yager [Yager, 1987] ont proposé différentes définitions de l'opérateur de *consensus* respectivement 2.14a, 2.14b et 2.14c et les tableaux de vérité dans la table 2.4. Les valeurs inconnues sont mentionnées avec le symbole $\diamond$.

$$t_x \oplus t_y = \frac{1}{i_x + i_y - i_x i_y} \begin{pmatrix} b_x i_y + i_x b_y \\ i_x i_y \\ d_x i_y + i_x d_y \end{pmatrix} \tag{2.14a}$$

$$t_x \oplus t_y = \frac{1}{1 - b_x d_y - d_x b_y} \begin{pmatrix} b_x b_y + b_x i_y + i_x b_y \\ i_x i_y \\ d_x d_y + d_x i_y + i_x d_y \end{pmatrix} \tag{2.14b}$$

$$t_x \oplus t_y = \begin{pmatrix} b_x b_y + b_x i_y + i_x b_y \\ i_x i_y + b_x d_y + d_x b_y \\ d_x d_y + d_x i_y + i_x d_y \end{pmatrix} \tag{2.14c}$$

Dans le cas des opérateurs de Dempster-Shafer et Jøsang, la différence réside dans la gestion du conflit. L'opérateur de Dempster-Shafer est défini seulement pour $1 - b_x d_y - d_x b_y > 0$, c'est-à-dire qu'il est indéfini pour la combinaison de la croyance (*belief*) avec la plausibilité (*disbelief*). La masse de probabilité des combinaisons est éliminée. b,i et d sont re-normalisés pour que $b + i + d = 1$. L'opérateur de Jøsang est, par contre, défini pour $i_x + i_y - i x_i y > 0$, c'est-à-dire qu'il est indéfini pour deux valeurs de la croyance (*belief*) et deux valeurs de plausibilité

$\oplus$	$+$	$\varnothing$	$-$
$+$	$\diamond$	$+$	$\diamond$
$\varnothing$	$+$	$\varnothing$	$-$
$-$	$\diamond$	$-$	$\diamond$

(a) Jøsang

$\oplus$	$+$	$\varnothing$	$-$
$+$	$+$	$+$	$\diamond$
$\varnothing$	$+$	$\varnothing$	$-$
$-$	$\diamond$	$-$	$-$

(b) Dempster-Shafer

$\oplus$	$+$	$\varnothing$	$-$
$+$	$+$	$+$	$\varnothing$
$\varnothing$	$+$	$\varnothing$	$-$
$-$	$\varnothing$	$-$	$-$

(c) Yager

TABLE 2.4 – Tables de vérité des différentes définitions de l'opérateur *consensus*

(*disbelief*). La normalisation de Jøsang a pour effet d'ignorer le conflit ce qui mène à des effets contre-intuitifs [Zadeh, 1984].

c) Extension de la théorie des croyances pour la gestion de contexte

Susan Mc Keever propose une extension de la théorie des croyances permettant de prendre en compte la qualité de contexte lors de l'inférence de situations de contexte [McKeever et al., 2009b, McKeever, 2011]. Les critères de qualité utilisés sont la précision, définie comme l'intervalle sur lequel une donnée d'un capteur est correcte, pour une exactitude donnée, et le flou (*fuzziness*) d'une information de contexte permettant de mesurer l'imprécision. Si la qualité de contexte est disponible et quantifiable, elle est incorporée dans les masses élémentaires des données de capteurs (voir plus haut) afin de rendre plus réaliste la fonction de croyance.

La figure 2.9 montre les étapes de l'inférence d'une situation de contexte telle que proposée par [McKeever et al., 2009b]. Au niveau des capteurs, chaque donnée collectée est associée avec une mesure de croyance, intégrant la QoC. La croyance est propagée le long des chemins du graphe orienté au sein de cadres de discernement. Pour chaque situation de contexte, la croyance issue des sources pertinentes est alors fusionnée en une mesure de la croyance totale en utilisant la règle de combinaison des évidences adéquate.

2.4.3 Méthodes combinées

Certains travaux essaient de tirer profit de différentes techniques, en les utilisant à plusieurs niveaux de la gestion de contexte.

[Ricquebourg et al., 2008] et [Nzekwa et al., 2010] proposent une architecture multi-couche dans laquelle la méthode de gestion de contexte la plus adéquate est sollicitée. [Nzekwa et al., 2010] définit un système complexe d'algorithmes d'apprentissage qui s'appuie sur les méthodes de *Dempster-Shafer* et *les réseaux Bayésiens* dans la *composition horizontale* du système. [Ricquebourg et al., 2008] fusionne les données des capteurs. La combinaison de données est réalisée à l'aide de la règle de Bayes de deux façons :

– au niveau de la modélisation à l'aide de la règle de Bayes,
– par la règle de Bayes directement, où l'information issue d'une source vient mettre à jour

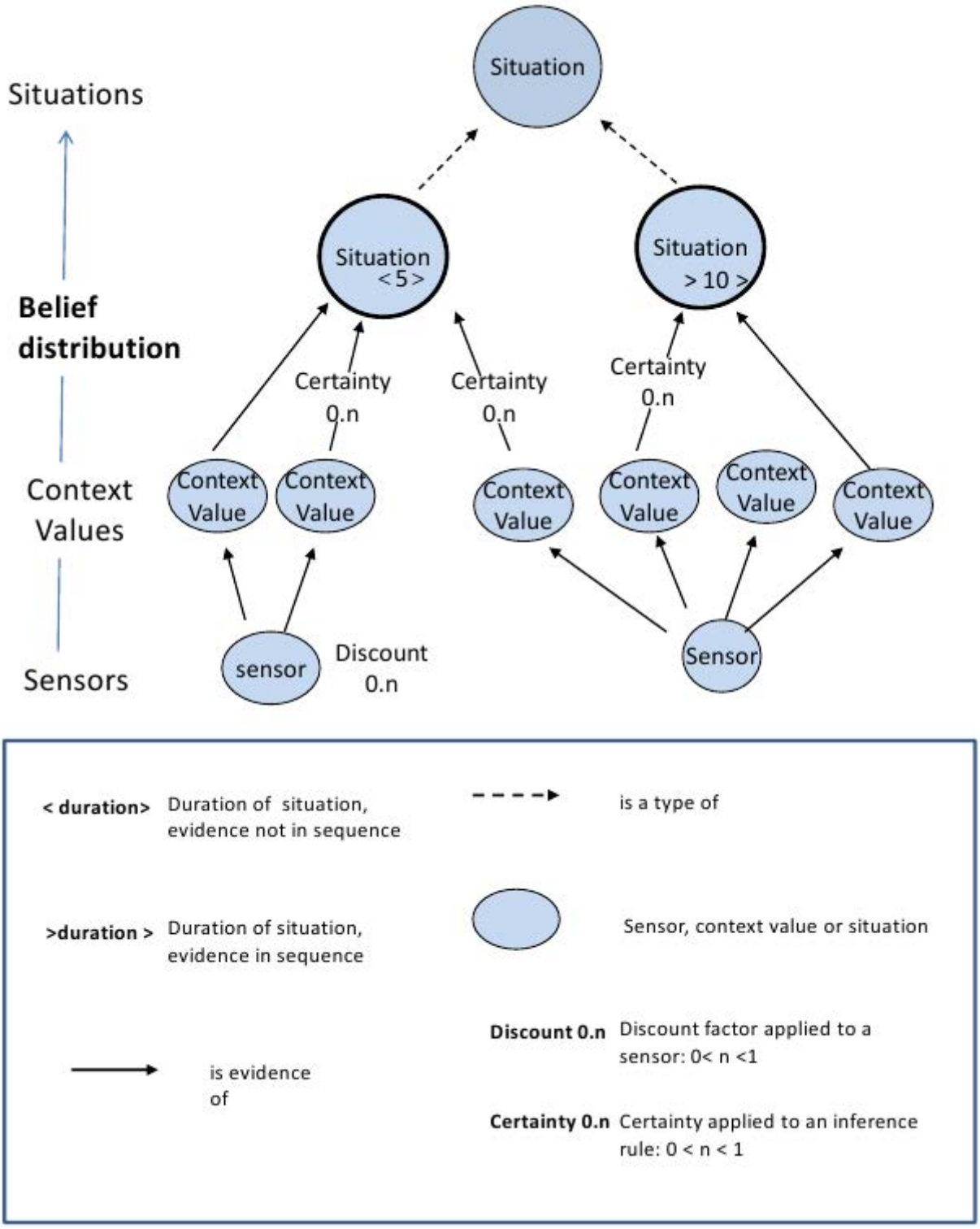


FIGURE 2.9 – Diagramme d'inférence de situations [McKeever et al., 2009b]

l'information estimée d'après les sources précédentes.

2.4.4 Synthèse

Lorsque les informations de contexte collectées sont incertaines, le risque existe de baser une décision sur des informations erronées [Chen and Kotz, 2000, Dey et al., 2001b], compromettant ainsi l'adoption par les utilisateurs de l'informatique sensible au contexte si le comportement de l'application ne correspond pas à leur attente. Nous avons présenté quelques-unes des méthodes de gestion de l'incertitude à la section 2.4.1. Cependant, le coût reste encore très élevé pour raisonner sur des informations incertaines vu la complexité des solutions à mettre en œuvre [Dargie and Hamann, 2006b, Ranganathan et al., 2004a].

C'est pourquoi d'autres travaux cherchent à réduire cette incertitude par la fusion de plusieurs informations de contexte issues de sources différentes afin d'améliorer la qualité de l'information résultante. Dans la section 2.4.2, nous avons discuté des méthodes non probabilistes pour la fusion et l'inférence de contexte. Nous avons ensuite abordé la théorie des croyances de Dempster-Shafer qui connaît un regain d'intérêt ces dernières années pour l'inférence de situations de contexte [McKeever et al., 2009b].

2.5 Modélisation de la qualité de contexte

La notion de contexte a engendré de nombreux travaux autour de sa définition et de sa modélisation [Bettini et al., 2010]. Les principales familles de modélisation de contexte sont la définition de profils (p.ex. CC/PP [Klyne and al., 2007]), les bases de données (p.ex. CML [Henricksen and Indulska, 2006]), l'IDM (Ingénierie Dirigée par les Modèles) et les ontologies (p.ex CONON [Wang et al., 2004]). Mais rares sont les modèles qui prennent en compte la qualité de contexte. Nous abordons dans cette section les travaux de modélisation de la qualité de contexte qui ont permis de mieux appréhender les relations entre les différents concepts manipulés et entités en jeu. Plusieurs approches existent. Nous commençons par présenter les travaux basés sur une modélisation en UML issue de l'IDM puis nous passons en revue plusieurs approches à base d'ontologies.

2.5.1 Modélisation UML

Plusieurs travaux proposent des méta-modèles de contexte et de sensibilité au contexte avec des approches d'ingénierie dirigée par les modèles. Les langages de méta-modélisation utilisés sont MOF (*Meta-Object-Facility*) [OMG, 2006], EMF (*Eclipse Modeling Framework*) [Budinsky et al., 2008], ou le profil UML [OMG, 2003]. Ces modèles permettent aux intergiciels de pouvoir prendre en compte des espaces d'informations de contexte variables et extensibles. Nous présentons dans cette section les travaux les plus significatifs proposant des modèles de contexte décrits en UML et qui intègrent la qualité de contexte.

2.5.1.1 Travaux de l'Université de Dublin (Irlande)

[McKeever et al., 2009a] construisent un modèle de qualité de contexte avec les principes suivants : Les problématiques de la modélisation de la qualité sont différentes dans chaque couche de gestion de contexte (c'est-à-dire dans les couches capteurs, abstraction de contexte et analyse de situation de contexte) [Ye et al., 2008]. De plus, il est nécessaire de mettre en évidence de manière explicite les agrégations entre les paramètres de qualité à travers les différentes couches de gestion de contexte. Par ailleurs, le modèle de qualité de contexte doit être extensible afin de pouvoir prendre en compte les besoins spécifiques des systèmes.

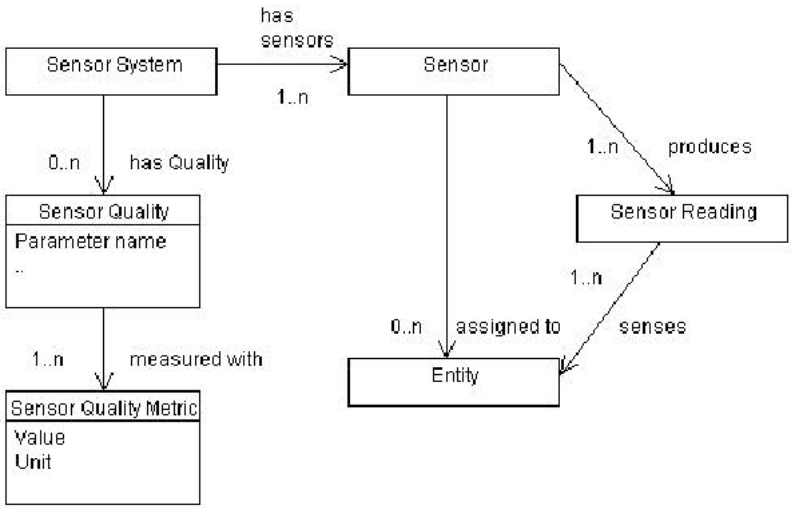


FIGURE 2.10 – Modèle de qualité des capteurs [McKeever et al., 2009a]

Dans le modèle présenté, les capteurs représentent n'importe quelle source de données physique (mesure dans l'environnement de contexte par exemple) ou logique (valeur calculée, valeur extraite d'un profil utilisateur du système, etc.). La Figure 2.10 montre qu'un capteur (*sensor*) peut avoir zéro ou plusieurs paramètres de qualité de contexte. Pour chaque paramètre de qualité, une ou plusieurs métriques peuvent être déterminées.

La Figure 2.11 montre la modélisation de la transformation des données collectées par le capteur en une information de contexte abstraite et significative pour l'application par l'acheminement des données du capteur à travers un ou plusieurs filtres. L'événement de contexte est modélisé comme une classe d'association UML. Il peut être également modélisé comme une association entre deux entités. Le modèle permet de modéliser toute relation entre les événements de contexte avec la qualité associée.

2.5.1.2 Travaux de l'Université de Vienne (Autriche)

[Manzoor et al., 2008] considèrent plusieurs objectifs lors de la création du modèle : Quels sont les besoins requis par le système de contexte ? Quel est le but de la collecte des informations

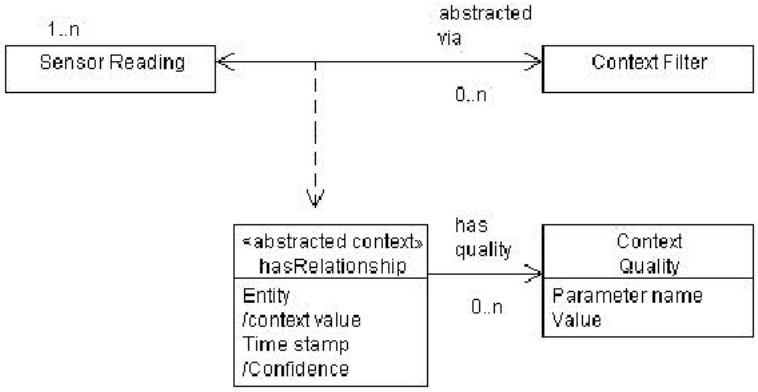


FIGURE 2.11 – Modèle de qualité des événements de contexte [McKeever et al., 2009a]

de contexte ? Comment ces informations vont être utilisées ?

La Figure 2.12 représente la définition du type paramètre de qualité de contexte *QoCParameterType*. Ce nœud XML contient une séquence d'éléments : ce sont les différents paramètres de qualité de contexte utilisés (*fraîcheur, confiance, complétude et importance*). Le type de ces éléments est *QoC-Decimal* qui est défini comme un type strict ayant l'intervalle de valeur $[0.0, 1.0]$

```
<xs:complexType name ="QoCParametersType">
    <xs:sequence>
        <xs:element name="Up-to-dateness" type="ci:QoC-Decimal"/>
        <xs:element name="Trust-worthiness" type="ci:QoC-Decimal"/>
        <xs:element name="Completeness" type="ci:QoC-Decimal"/>
        <xs:element name="Significance" type="ci:QoC-Decimal"/>
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="QoC-Decimal">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value ="0.0"/>
        <xs:maxExclusive value ="1.0"/>
    </xs:restriction>
</xs:simpleType>
```

FIGURE 2.12 – [Manzoor et al., 2008]

2.5.1.3 Travaux de l'Université de Twente (Pays-Bas)

[Neisse et al., 2008, Neisse, 2012] présente un *modèle* de contexte qui définit une *entité* identifiée par un attribut d'*identité* (Figure 2.13). Cet attribut peut être associé avec un *contexte*

particulier à un moment donné dans le temps estampillé pour être lié au contexte, c'est le *timestamp* dans la Figure 2.13.

Ce modèle définit explicitement plusieurs définitions de certains paramètres de qualité tels que la précision, la résolution spatiale et la probabilité. Le modèle est présenté comme générique et extensible, il supporte les différents paramètres de qualité de contexte les plus utilisés.

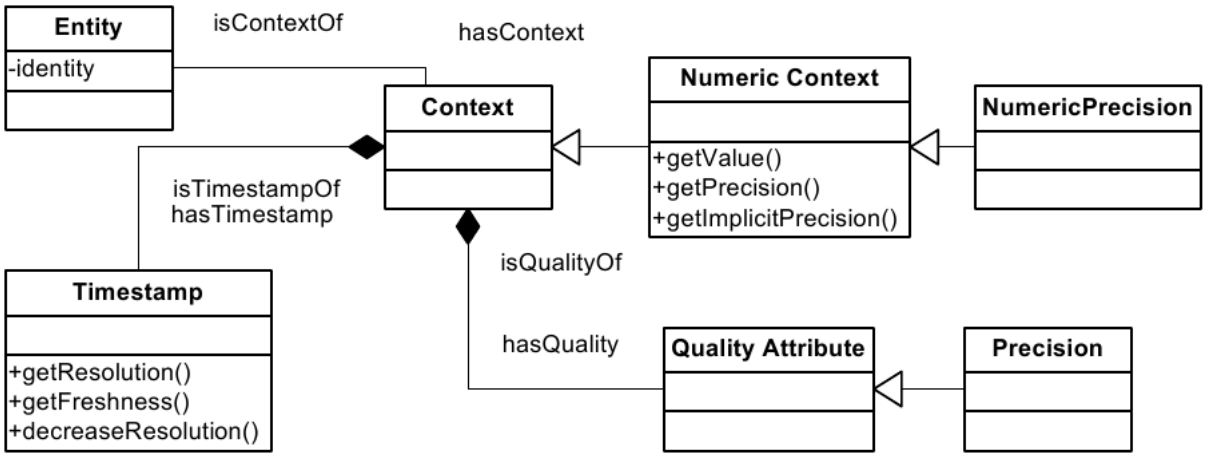


FIGURE 2.13 – Modèle UML de la QoC [Neisse et al., 2008]

Dans le modèle conçu pour la *confiance* (Figure 2.14), ce paramètre de qualité de contexte est défini comme le degré de croyance que peut avoir une entité de confiance «*trustor*» en une entité la méritant, par rapport à un aspect particulier. Les entités donnant la confiance peuvent interpréter la mesure de confiance de l'entité méritante d'une manière indépendante ou en combinant une série de mesures basée sur l'honnêteté, la compétence, la réputation, la crédibilité ou la fidélité. Dans le cas de la gestion de contexte, cette mesure de confiance se base donc sur la fourniture des informations de contexte concernant le propriétaire du contexte selon un niveau de qualité de contexte défini.

2.5.2 Ontologie

Une ontologie est une description sémantique, structurée et formelle des concepts d'un domaine et de leurs inter-relations [Uschold and Grüninger, 1996]. Elle se présente sous la forme d'un ensemble structuré de termes et de concepts organisés dans un graphe avec des relations sémantiques, et des relations de composition et d'héritage entre ces concepts. Les ontologies sont souvent associées à des moteurs d'inférence pour raisonner sur les informations de contexte selon des règles d'inférence.

Nous présentons ci-après les principaux travaux utilisant les ontologies pour exprimer un modèle de qualité de contexte.

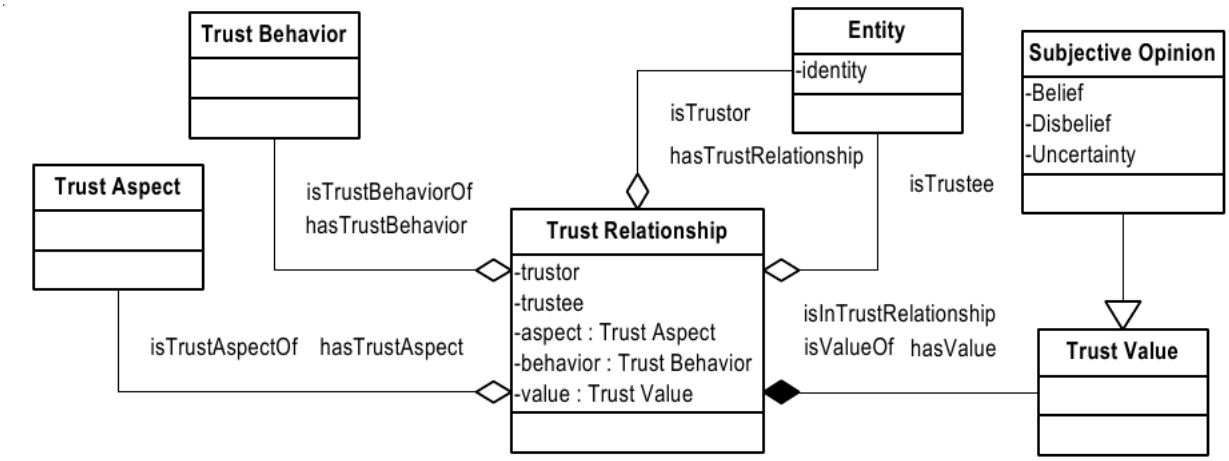


FIGURE 2.14 – Modèle UML de la confiance [Neisse et al., 2008]

2.5.2.1 Travaux de l'Université de Louvain (Belgique)

Dans [Preuveneers and Berbers, 2006], les auteurs proposent le système CODAMOS et définissent une ontologie de contexte basique, mais générique et extensible. Elle est construite autour de quatre entités principales :

Utilisateur Selon Dey [Dey et al., 2001a], les informations de contexte ne sont pertinentes que si elles influent sur une tâche de l'utilisateur. L'entité utilisateur est liée au profil, au rôle, à la tâche et même à l'humeur qui peuvent influer sur l'adaptation de l'application.

Environnement L'utilisateur inter-réagit avec l'environnement qui l'entoure et une application bien informée de cet environnement peut produire des décisions correctes. L'entité environnement est liée avec la localisation (relative ou absolue), le temps et les conditions climatiques comme la température, la pression, etc. mais également avec la plate-forme logicielle utilisée.

Plate-forme L'entité plate-forme fournit une description du matériel de l'appareil de l'utilisateur ainsi que le système et les logiciels installés sur cet appareil. La description de la plate-forme est importante pour la génération de code ou lors du déploiement dynamique ou statique des services sur l'appareil de l'utilisateur. C'est pourquoi les détails au niveau logiciel comme le nom de l'application, la version et l'édition, la machine virtuelle et le système d'exploitation permettent d'adapter les services demandés. Il est aussi important de connaître les caractéristiques matérielles telles que le type du processeur, la mémoire volatile ou permanente pour les mêmes raisons.

Service Dans plusieurs domaines de l'informatique, le concept de service est une entité qui offre une fonctionnalité particulière. [Preuveneers and Berbers, 2006] se concentre sur l'adaptation et l'interaction dynamique des services avec le contexte. Pour effectuer une description multi-niveau, les auteurs étendent leur ontologie avec un service appelé OWL-s [8] destiné aux services web et au web sémantique mais qui fournit un système riche et standard pour décrire les services en général.

Les entités attachées au concept service sont les suivantes. Le profil de service est une description lisible destinée à l'utilisateur ou à l'administrateur. Le modèle de service décrit le service en détail (exemple : flux de contrôle, flux de données). Les bases du service fournissent des informations sur l'implémentation du service.

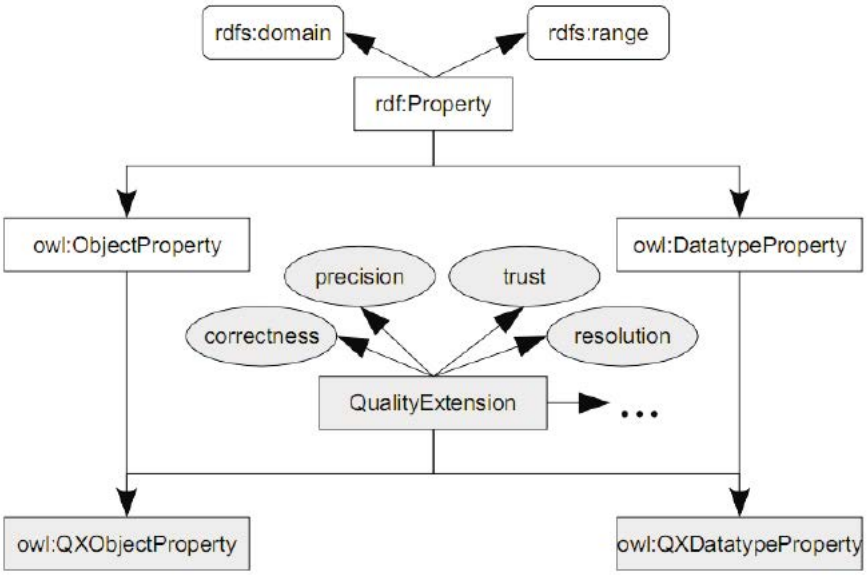


FIGURE 2.15 – Extension du langage OWL avec les paramètres de QoC [Preuveneers and Berbers, 2006]

L'extension du langage OWL proposée par [Preuveneers and Berbers, 2006], illustrée à la Figure 2.15, consiste à introduire deux nouveaux types de propriétés : les *QXObjectProperty* et *QXDDatatypeProperty*. La définition du type de propriété *DatatypeProperty* en langage OWL étendu est représentée à la Figure 2.16. La Figure 2.17 représente la modélisation d'un capteur en langage OWL avec extension. Le nouveau type de propriété *QXDataTypeProperty* indique

8. The OWL Services Coalition : OWL-S : Semantic Markup for Web Services. http ://www.daml.org/services/owl-s/1.0/owl-s.html (2003)

que le capteur fournit la température (*hasTemperature*) associée avec différents paramètres de qualité (*precision*, *correctness*, *resolution*).

```
<owl:Class rdf:ID="QualityExtension" />

<owl:DatatypeProperty rdf:about="#precision">
  <rdfs:domain rdf:resource="#QualityExtension" />
  <rdfs:range rdf:resource="&xsd;#int" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#correctness">
  <rdfs:domain rdf:resource="#QualityExtension" />
  <rdfs:range rdf:resource="&xsd;#int" />
</owl:DatatypeProperty>
...

<owl:Class rdf:ID="QXDatatypeProperty">
  <rdfs:subClassOf rdf:resource="&owl;#DatatypeProperty" />
  <rdfs:subClassOf rdf:resource="&owl;#QualityExtension" />
</owl:Class>
<owl:Class rdf:ID="QXObjectProperty">
  <rdfs:subClassOf rdf:resource="&owl;#ObjectProperty" />
  <rdfs:subClassOf rdf:resource="&owl;#QualityExtension" />
</owl:Class>
```

FIGURE 2.16 – Sérialisation de la classe *QualityExtension* en langage OWL [Preuveneers and Berbers, 2006]

2.5.2.2 Travaux de l'Université de Zhejiang (Chine)

[Tang et al., 2007a] proposent un modèle de qualité de contexte en construisant une ontologie spécifique pour la qualité. Le modèle est créé avec le langage *OWL-DL* (Web Ontology Language Description Logics). La classe *Application* souscrit à la classe *Property* (propriété). En plus d'un statut et de relations, la classe proriété a principalement une qualité associée. La classe qualité a un ou plusieurs paramètres et est spécifiée par l'application. La classe *Parameters* (paramètres) a une valeur et peut être l'un des différents paramètres de qualité de contexte tels que la précision ou la fraîcheur. La Figure 2.18 montre les interactions entre les différentes classes.

```
<owl:Class rdf:ID="Sensor" />

<qx:QXDatatypeProperty rdf:about="#hasTemperature">
   <rdfs:domain rdf:resource="#Sensor" />
   <rdfs:range rdf:resource="&xsd;#int" />
   <qx:precision>95</qx:precision>
   <qx:correctness>100</qx:correctness>
   <qx:resolution>1</qx:resolution>
   ...
</qx:QXDatatypeProperty>
```

FIGURE 2.17 – Représentation d'un capteur avec des paramètres de QoC à l'aide de l'extension du langage OWL [Preuveneers and Berbers, 2006]

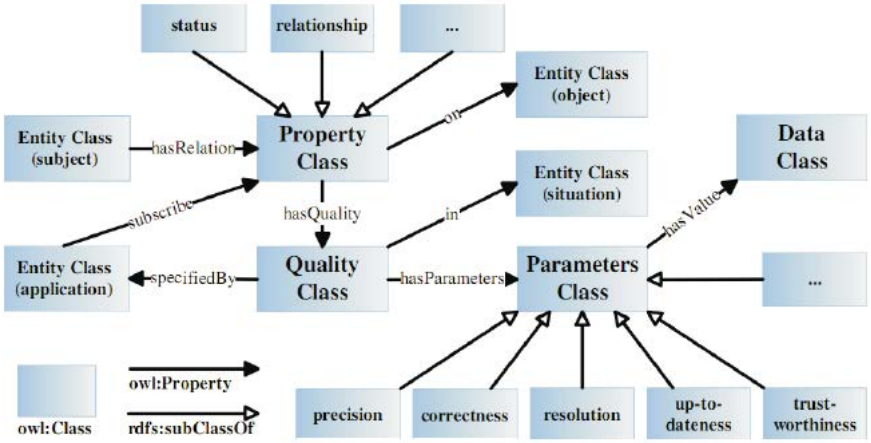


FIGURE 2.18 – Modélisation de la qualité de contexte en langage OWL [Tang et al., 2007a]

2.5.2.3 Travaux de l'Université de Grenoble (France)

Dans [Bringel Filho and Martin, 2008, Bringel Filho et al., 2010, Filho, 2010], les auteurs revisitent les différentes notions de qualité de contexte dans l'objectif de proposer une famille de modèles de contrôle d'accès pour les systèmes ubiquitaires. Ils classifient la qualité de contexte en deux groupes : les *paramètres de qualité* et les *indicateurs de qualité*. Les **paramètres de qualité (QoCP)** sont «*toute information capturée de l'environnement et qui peut être utilisée pour mesurer les QoCI*». Par exemple, `captureTime`, `lifeTime` et `currentTime` sont utilisés pour la mesure de la *Fraîcheur*. Les paramètres de qualité sont à rapprocher des sources de qualité de contexte décrites par [Manzoor et al., 2008]. Les **indicateurs de qualité (QoCI)**

correspondent à «*tout aspect de qualité qui peut être évalué par le système et utilisé dans la description de la qualité de l'information contexte*», et sont comparables aux paramètres de QoC définis par [Kim and Lee, 2006a] et [Manzoor et al., 2008].

Le tableau 2.5 représente la relation de chaque indicateur de qualité de contexte avec les paramètres qui lui sont associés. Par exemple, afin de définir la *sensibilité*, les auteurs utilisent les paramètres `numberOfDisclosureLevel` et `currentDisclosureLevel` qui peuvent être déterminés dans le cas de la localisation par la Table 2.6

QoC Indicator	QoC Parameter
Up-to-dateness	captureTime, currentTime, lifeTime
Sensitiveness	numberOfDisclosureLevel, currentDisclosureLevel
Access Security	CurrentSecurityLevel, NumberOfSecurityLevel
Completeness	NumberOfAnsweredRequest, NumberOfRequest
Precision	NumberOfPrecisionLevel, CurrentPrecisionLevel, ProcessAccuracy
Resolution	NumberOfGranularityLevel, CurrentGranularityLevel, EntityLocation

TABLE 2.5 – Relation entre QoCI et QoCP [Bringel Filho et al., 2010]

Les auteurs définissent une ontologie de contexte de haut niveau (`Context Top Ontology`, voir figure 2.19) où la qualité de contexte est construite à partir des classes principales `QoCI` et `QoCP`. La classe `ElementaryElement` représente les données de contexte brutes. Le lien entre un élément de contexte et les paramètres de QoC est établi à travers la propriété `hasQoCP`. La classe

level	Indoor	Outdoor
0	Undisclosed	Undisclosed
1	(LMK)	(C)
2	(LMK, FLR)	(C,A3)
3	(LMK, FLR, LOC)	(C,A3,A6-STS)
4	-	(C,A3,A6-STS,HNO-HNS)
5	-	GPS coordinates

TABLE 2.6 – Localisations intérieures et extérieures associées aux niveaux de sensibilité [Filho, 2010]

`QoCI` modélise les indicateurs de QoC. Afin d'associer les éléments de contexte avec `QoCI`, la propriété `hasQoI` est définie. Les auteurs utilisent la réification pour spécifier les relations *ternaires* (`QoCCompDerInfMtd`) entre les éléments de contexte calculés (`ComputedElement`), l'indicateur deBayes1763 `QoC` qui sera calculé et la méthode de calcul `computationMethod`.

2.5.3 Synthèse

La création de modèles de contexte prenant en compte la qualité de contexte pour définir les relations entre les différents concepts et entités en jeu dans les applications ubiquitaires est indispensable. L'un des intérêts de l'ingénierie dirigée par les modèles telle que préconisée par l'OMG est de pouvoir relier des éléments du modèle des applications avec des éléments du modèle de contexte et ainsi de pouvoir définir des points de variation de l'application en fonction de situations d'adaptations.

Lorsqu'une approche par ontologies est choisie dans un environnement ubiquitaire, plusieurs questions se posent pour sa mise en œuvre. La taille de la base de connaissances lui permet-elle d'être stockée sur un terminal mobile ? Comment sont contrôlées les ressources consommées par le moteur d'inférence et par le système de gestion de la base de connaissances ? L'avantage des ontologies est l'adaptation du système à l'évolution de l'environnement. Le coût de mise à jour de la base des connaissances peut, par contre, être significatif et freiner la mise en œuvre de cette approche [Chen et al., 2003], [Preuveneers et al., 2004], [Preuveneers and Berbers, 2007b]. Malgré leur popularité croissante, les ontologies restent limitées pour prendre en compte les incertitudes liées aux informations de contexte [Henricksen, 2003]. Nous considérons qu'il n'est donc pas envisageable de reposer exclusivement sur des moteurs d'inférence ontologiques pour la gestion de contexte, également pour des raisons de performances [Bouzeghoub et al., 2010].

2.6 Conclusion

Dans ce chapitre nous avons étudié les différents aspects de la qualité contexte, tout d'abord la définition de la qualité qui a évolué depuis sa première identification. Ensuite la classification des paramètres de qualité et l'étude de leurs méthodes d'évaluation : il y a les paramètres généraux qui sont présents dans les différentes applications sensibles au contexte comme la précision, l'exactitude, et la confiance notamment et d'autres spécifiques au domaine. Nous avons présenté différents aspects du calcul des méta-données de qualité et parfois une différence de définition pour le même paramètre ce qui témoigne d'une richesse des concepts manipulés dans différents domaines d'applications.

Nous mettons également en évidence une organisation des informations de qualité en couches, avec des paramètres de qualité de bas niveau directement liés aux sources de contexte tels que la précision, la complétude ou la fraîcheur, et d'autres abstraits qui sont de plus haut niveau d'abstraction et d'analyse comme la confiance ou l'importance.

Dans le tableau 2.7, nous résumons les travaux qui ont participé à l'évolution de la perception de la qualité de contexte, à sa définition et à l'identification de ses différentes dimensions.

D'autres travaux se sont focalisés sur la modélisation de la qualité de contexte soit en s'appuyant sur leur propre définition, soit en étudiant les travaux cités dans le tableau 2.7 en adaptant le modèle au domaine étudié. Parmi les techniques de modélisation utilisées, les travaux que nous avons présentés font appel à la modélisation objet en UML ou bien à la modélisation sémantique à l'aide d'ontologies. Le tableau 2.8 résume l'orientation de ces travaux, la méthode utilisée et le domaine d'application

Les imperfections inhérentes aux informations de contexte peuvent être réduites le plus possible par des méthodes mathématiques ou non telles que celles décrites à la section 2.4, cependant elles ne peuvent pas être complètement levées. Il est donc important de **prévoir les mécanismes nécessaires pour limiter l'impact que peuvent avoir ces imperfections**, notamment en permettant au système de prendre en compte leur existence même. C'est ainsi que la qualité des informations de contexte devient primordiale pour les applications sensibles au contexte.

Deux approches, l'une homogène et l'autre hétérogène, peuvent être envisagées pour intégrer la qualité de contexte dans un canevas logiciel [Henricksen, 2003].

- Une approche homogène permet d'associer à chaque information de contexte une mesure numérique de sa qualité. Ceci facilite les comparaisons et la composition des mesures de qualité. Cependant, une approche homogène n'est pas toujours possible car la qualité ne peut pas toujours être déterminée de manière systématique.
- Une approche hétérogène part du principe qu'il ne peut y avoir une mesure unique et une représentation standard du niveau de qualité d'une information de contexte. Chaque type d'information possède ainsi son propre ensemble de mesures de qualité.

C'est cette dernière approche que nous suivons dans COSMOS [Conan et al., 2007a] au sein duquel nous mettons en place une gestion de QoC souple et flexible, fonction de la nature des informations de contexte et des capacités des sources de contexte. Ceci permet également une meilleure optimisation des performances en modulant le coût de calcul des critères de qualité en fonction de ce qui est effectivement nécessaire. Comme indiqué dans le récent article [Conti et al., 2012] discutant des défis à relever pour une convergence du monde cybernétique et du monde physique, des efforts de recherche dédiés à la QoC restent nécessaires pour explorer des solutions intergicielles performantes et dynamiques.

Par ailleurs, dans notre approche, nous avons choisi de suivre les principes de l'IDM qui permet de définir des liens entre la modélisation de contexte, grâce à laquelle il est possible d'exprimer des situations de contexte, et la modélisation de la *sensibilité* au contexte, qui associe ces situations de contexte avec des entités de l'application. Ceci contribue à étendre les travaux de modélisation existants et présentés dans ce chapitre en les intégrant dans une démarche globale pour faciliter le développement des applications et bénéficier des outils de vérification automatique et de génération de code associés à l'IDM.

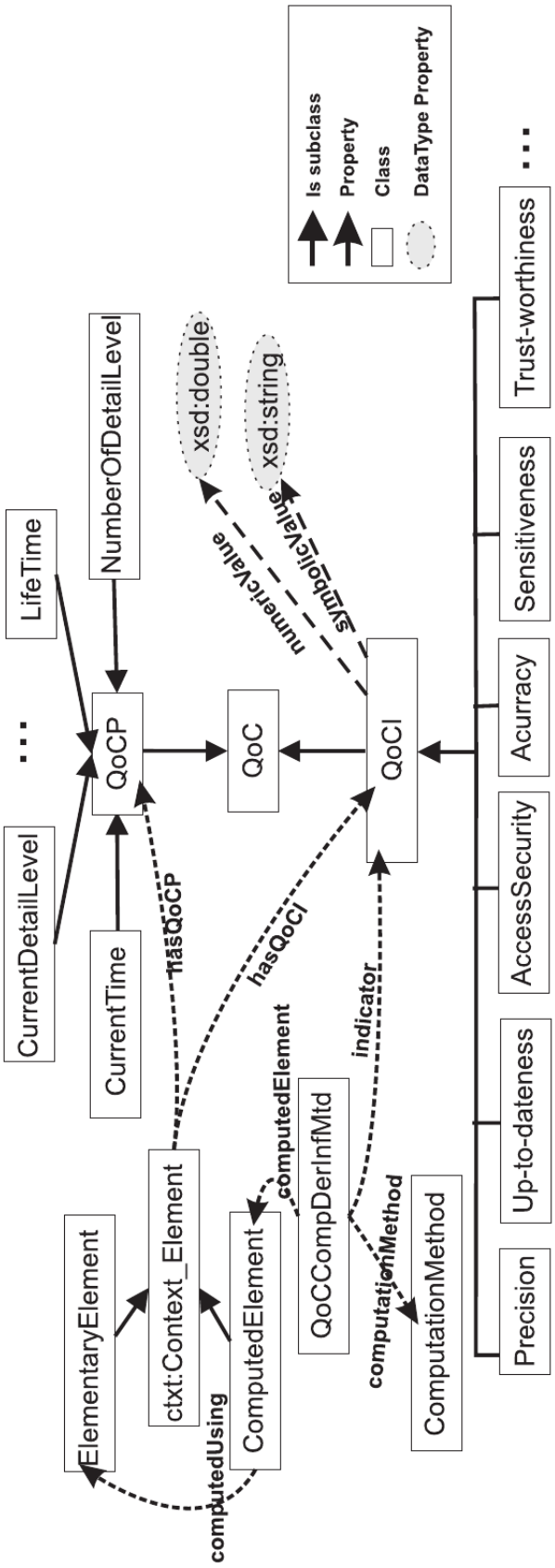


FIGURE 2.19 – Ontologie de la qualité de contexte [Bringel Filho et al., 2010]

Travaux	Résumé	Critères de QoC	Commentaire
[Buchholz et al., 2003] [Krause and Hochstatter, 2005]	Première définition de la Qualité de Contexte	Exactitude, complétude sécurité d'accès cohérence de la représentation	Premiers travaux définissant la qualité de contexte pour les systèmes sensibles au contexte
[Kim and Lee, 2006a]	Relation entre dimensions de qualité de l'information et paramètres de QoC	Résolution, précision valeur-ajoutée,réputation accessibilité interprétabilité objectivité, maniabilité	Énumération des qualités avec classification : Dimensions et Paramètres
[Manzoor et al., 2008] [Manzoor, 2010]	Sources et paramètres de qualité de contexte	fraîcheur, confiance complétude, précision importance, exactitude	Méthodes d'évaluation des paramètres de qualité
[Neisse et al., 2008] [Neisse, 2012]	Définitions basées sur le modèle OSI	exactitude, précision résolution temporelle résolution spatiale fraîcheur	La précision comme concept central
[Bringel Filho et al., 2010] [Filho, 2010]	Indicateurs et paramètres de qualité de contexte	Fraîcheur Complétude importance Incohérence	Application au contrôle d'accès

TABLE 2.7 – Synthèse des travaux traitant de la qualité de contexte

Travaux	aspect	méthode	domaine	Framework
[Preuveneers and Berbers, 2006]		Ontologie	Réseaux de capteurs sans fil	CODAMOS
[Tang et al., 2007a]		Ontologie		-
[Manzoor et al., 2008]	Analyse de la qualité	XML	Localisation	[Dorn et al., 2006]
[Neisse et al., 2008]	Fournisseur de confiance	UML	Confiance Sécurité	-
[McKeever et al., 2009b]	modélisation prédiction	UML Demster-Shafer		
[Bringel Filho et al., 2010]	Optimisation multi-dimensions	Ontologie	Contrôle d'accès Sécurité	

TABLE 2.8 – Synthèse des modèles de QoC

Deuxième partie

Contributions

Chapitre 3

Intégration de la gestion de la QoC dans COSMOS

Sommaire

Nous présentons dans ce chapitre la première contribution de cette thèse concernant l'intégration de la gestion de la qualité de contexte au sein de la plate-forme COSMOS. Notre travail étant une extension de COSMOS, le chapitre commence par décrire les principes de conception sous-jacents à COSMOS puis détaille son architecture. Nous expliquons ensuite comment nous proposons de spécialiser les composants nœuds de contexte pour traiter de la qualité des informations de contexte. Nous terminons ce chapitre par la proposition d'un mécanisme de gestion fine de la qualité de contexte permettant de prendre en compte l'importance relative des différents critères de qualité considérés par une application.

Ce chapitre a donné lieu aux publications [Abid et al., 2009b] et [Abid and Chabridon, 2011].

3.1 COSMOS : Gestionnaire de contexte orienté composant

3.1.1 Introduction

Les environnements ubiquitaires imposent des contraintes fortes sur la conception et le développement des applications. Au-delà de l'action d'adaptation elle-même, la prise de décision pour le déclenchement de cette adaptation est un problème complexe pour lequel peu de solutions existent. Cette décision repose sur une collecte, une analyse et une synthèse des nombreux paramètres physiques et logiques fournis par le contexte d'exécution. Pour cela, COSMOS a été proposé pour la gestion des informations de contexte.

COSMOS (**CO**ntext entitie**S** co **M**positi**O**n and **S**haring) [Conan et al., 2007a, Conan et al., 2007b, Rouvoy et al., 2008, Conan et al., 2008] [9] est un canevas logiciel orienté composant pour la gestion d'informations de contexte dans les applications sensibles au contexte. Il est developpé en licence libre et son principal contributeur est l'équipe MARGE de Télécom SudParis. Il est développé selon trois principes : séparation entre les activités de collecte et de synthèse des données de contexte, organisation des politiques de gestion de contexte en assemblage de composants logiciels et utilisation systématique de patrons de conception.

Ainsi les objectifs de COSMOS sont clairement identifiés :

- composer des informations de contexte de manière déclarative, à opposer aux approches «par programmation» utilisées dans les canevas logiciels existants. Ceci doit faciliter la conception par composition, l'adaptation et la réutilisation des politiques de gestion de contexte.
- isoler chaque couche de l'architecture de gestion de contexte des autres couches afin de promouvoir la séparation des préoccupations et des cycles de vie des informations de contexte.
- fournir les concepts «système» pour gérer finement les ressources consommées par les différents traitements d'inférence.

9. http ://picoforge.int-evry.fr/projects/svn/COSMOS

3.1.2 Architecture de gestion de contexte

L'architecture générale d'un gestionnaire de contexte présentée à la figure 3.1 est inspirée de [Coutaz et al., 2005, Dey et al., 2001b, Yau et al., 2002]. La première couche de l'architecture est constituée de la collecte des données numériques brutes en provenance du système d'exploitation, des capteurs à proximité et directement accessibles à partir du terminal, des préférences données par l'utilisateur, et des informations de contexte en provenance des autres terminaux. Cette couche a pour rôle d'intégrer dans l'architecture globale du service de gestion de contexte les nombreux canevas logiciels utilisés pour les différentes sources d'informations de contexte. La couche de collecte fournit les données numériques brutes à la base du traitement par un *processeur de contexte*. Les données obtenues à l'aide des processeurs de contexte sont symboliques et constituent la perception du contexte. La couche suivante vers l'application exprime la sensibilité au contexte de l'application. Elle consiste, par exemple, en l'identification des situations d'adaptation de l'application cliente. Cette sensibilité au contexte est gérée par les conteneurs ou membranes des composants métier. La dernière couche est l'exploitation des informations de contexte par les entités de l'application métier.

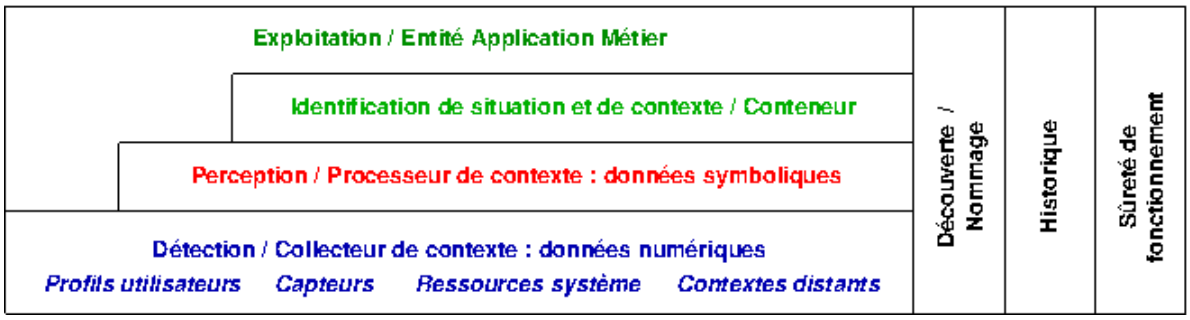


FIGURE 3.1 – Architecture générale d'un gestionnaire de contexte

3.1.3 Orientation composants

COSMOS est conçu en appliquant les principes de base d'un intergiciel : le canevas logiciel est construit à partir d'éléments génériques, spécialisables et modulaires afin de composer plutôt que de programmer. Il s'agit donc d'une architecture à base de composants pour la gestion d'informations de contexte.

Cette architecture à base de composants apporte une vision unifiée dans laquelle les mêmes concepts (composant, liaison, interface) sont utilisés pour développer les applications et les différentes couches intergicielles et systèmes sous-jacentes. Cette vision unifiée en facilite également la conception et l'évolution. Elle autorise en outre une vision hiérarchique dans laquelle l'ensemble « canevas et application » peut être vu à différents niveaux de granularité.

Cette notion d'architecture logicielle associée à l'approche orientée composant permet la composition des entités logicielles indépendamment de leurs implantations, rendant ainsi plus

aisée la compréhension de l'ensemble. La notion d'architecture logicielle favorise aussi la dynamicité en autorisant la redéfinition des liaisons de tout ou partie du canevas, voire de l'application à l'exécution. Ainsi, la re-configuration et l'adaptation des contextes nouveaux non prévus au départ en sont facilitées.

L'originalité de COSMOS est également l'expression de la composition de contexte dans un language de description d'architecture (ADL) et la projection de cette architecture sur un graphe de composants. Le modèle de composants et l'ADL utilisés sont ceux de Fractal [10] que nous présentons ci-après.

3.1.3.1 Composant Fractal

Les caractéristiques principales du modèle de composant Fractal sont motivées par l'objectif de pouvoir construire, déployer et administrer des systèmes complexes tels que des intergiciels ou des systèmes d'exploitation. Le modèle est ainsi basé sur les principes suivants :

- **Composant composite** : composant qui contient un ou plusieurs sous-composants pour l'obtention d'une vue uniforme des applications à différents niveaux d'abstraction.
- **Composant partagé** : sous-composant de plusieurs composites englobants pour modéliser les ressources et leur partage, tout en préservant l'encapsulation des composants.
- **Capacité d'introspection** : nécessaire pour l'observation d'une exécution d'un système.
- **Capacité de reconfiguration** : pour déployer et configurer dynamiquement un système.

3.1.3.2 Fractal ADL

Fractal est associé au langage de description d'architecture appelé Fractal ADL [Leclercq et al., 2007]. Basé sur une syntaxe XML, ce langage permet d'exprimer des assemblages de composants Fractal. La définition du type du document (DTD) et la chaîne de traitement de ce langage peuvent être étendues.

3.1.4 Nœud de contexte COSMOS

L'observation du contexte joue un rôle prépondérant dans le bon fonctionnement des applications ubiquitaires et est mise en place avec COSMOS par des politiques d'observation. Ces politiques sont décomposées en des composants logiciels à grain fin appelés *nœuds de contexte*. La figure 3.2 montre l'architecture de ce composant.

Toutes les informations de contexte sont des composants étendant le composite abstrait ContextNode. Les interfaces `Pull` et `CPush` sont les interfaces pour l'observation et la notification, respectivement. Les rapports d'observation sont des messages, constitués de sous-messages et de blocs (en anglais, `chunks`) typés. Dans COSMOS, toutes les informations élémentaires des rapports d'observation des entités observables liées aux collecteurs donnent lieu à un bloc typé.

Un ContextNode contient au moins un opérateur (composant primitif abstrait ContextOperator) et est connecté à un service de connexion en mode message. De manière générique, le

10. http ://fractal.ow2.org/

composant ContextOperator prend en entrée les données obtenues par observation ou suite à une notification, effectue un traitement, et si besoin (observation ou notification passante), transfère les résultats du traitement vers la sortie par réponse à l'observation ou par notification. Le concepteur de nouveaux opérateurs doit simplement spécialiser le composant ContextOperator en définissant les traitements à effectuer.

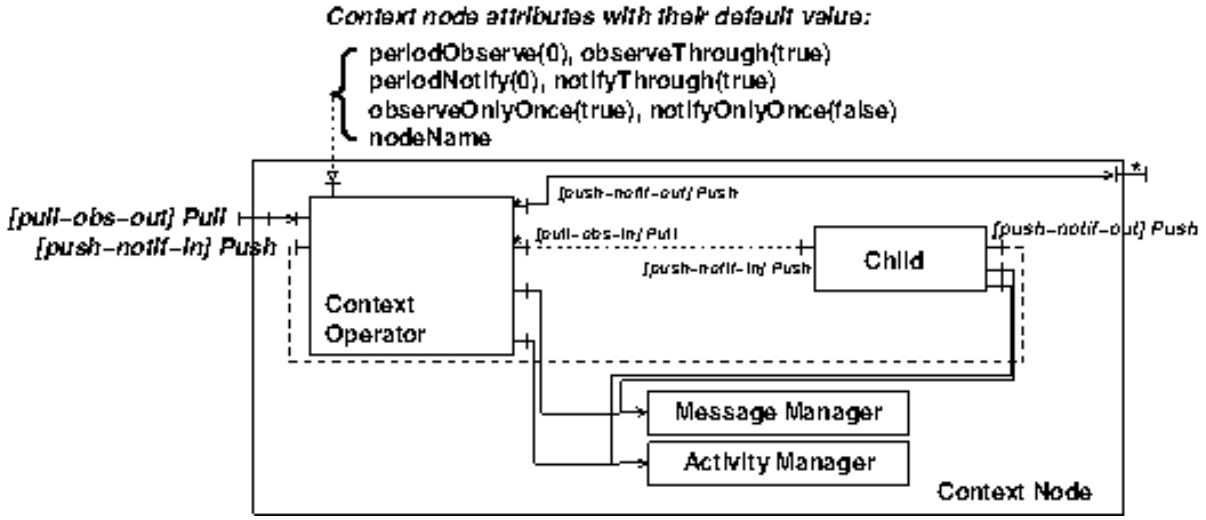


FIGURE 3.2 – Architecture d'un nœud de contexte COSMOS

Un nœud de contexte possède des propriétés qui définissent son comportement

- passif ou actif : un nœud passif est utilisé par des activités extérieures au nœud qui l'interrogent pour obtenir une information. Un nœud actif est équipé d'une activité pour exécuter une tâche donnée ;
- observation ou notification : les communications peuvent s'opérer du bas vers le haut ou du haut vers le bas de l'arborescence de nœuds de contexte. Les notifications correspondent aux messages envoyés par les nœuds à leurs parents (du bas vers le haut), tandis que les observations sont déclenchées par un nœud parent. Un nœud de l'arborescence peut être à la fois observateur et notificateur actifs ;
- passant ou bloquant : un nœud passant propage les observations et les notifications. Dans le cas bloquant, le nœud observé fournit l'information de contexte qu'il détient sans observer les nœuds enfants, et le nœud notifié modifie son état interne sans notifier les nœuds parents ;
- nommage : les noms des nœuds de contexte sont uniques pour permettre les parcours dans le graphe et les configurations.

Un nœud récupère des informations de contexte de nœuds enfants de la hiérarchie et infère une information de plus haut niveau d'abstraction. Le traitement correspondant est effectué dans l'opérateur de contexte. Le cycle de vie des nœuds de contexte enfants est contrôlé par les nœuds de contexte parents. Pour la gestion des activités, les nœuds de contexte actifs enregistrent

leurs activités auprès d'un gestionnaire d'activités. Ainsi, le gestionnaire d'activités, lui-même paramétrable, peut créer une activité par traitement (observation ou notification) ou bien une activité par nœud de contexte ou encore une activité pour tout ou partie de la hiérarchie. Pour la consommation mémoire, un gestionnaire de messages gère des réserves de messages et autorise des duplications par référence ou par valeur.

3.1.5 Politique de contexte

Une politique de contexte est un ensemble de nœuds de contexte interconnectés qui forment des niveaux de contexte. Les composants des niveaux du bas sont en relation avec les Context-Collectors qui sont directement liés aux sources de contexte et à l'environnement du contexte. Les niveaux intermédiaires analysent les informations brutes et appliquent les actions requises comme le filtrage, le reformatage, et éventuellement l'inférence. Les couches hautes sont des couches d'abstraction responsables du pilotage et de l'observation. Elles sont le point de contact avec l'application sensible au contexte.

Un exemple de politique de contexte définie avec COSMOS est présenté à la figure 3.3. Les nœuds du graphe sont repérés par leur nom, chacun indiquant le type d'opérateur du nœud de contexte. Les arcs du graphe représentent les relations d'inclusion, y compris les partages de nœuds enfants par plusieurs nœuds parents. À côté d'un nœud, figurent ses propriétés : actif/passif, bloquant/passant, etc. Dans notre exemple, la plupart des nœuds actifs observent ; seuls les nœuds détectant les changements d'états (« détecteur de changement des préférences utilisateur » et « détecteur de connectivité ») et de décision notifient ces changements vers l'application.

Les nœuds sans flèche sont passants pour l'observation et la notification. Dans le cas du nœud gestionnaire du réseau WiFi (en bas à droite de la figure), le nœud est bloquant pour l'observation car il gère lui-même une tâche d'observation. Le nœud gestionnaire du réseau WiFi permet de regrouper toutes les informations (qualité du lien, débit et « le débit est-il variable ? ») pour éviter que les trois nœuds parents ne multiplient les accès aux informations système et ne provoquent un grand nombre d'appels système.

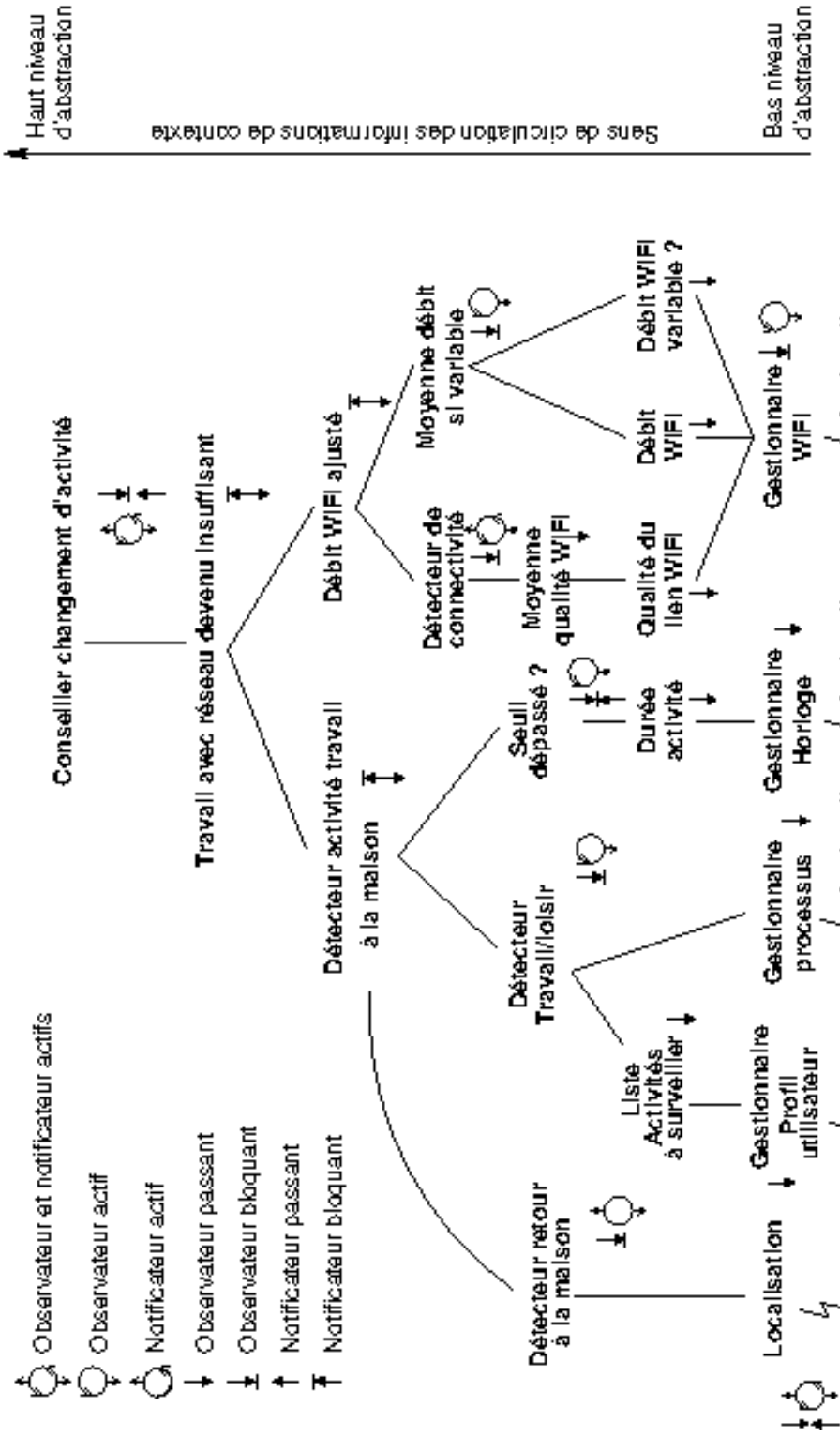


FIGURE 3.3 – Exemple d'une politique de contexte

3.2 Gestion de la qualité de contexte dans COSMOS

3.2.1 Préservation de la compatibilité

La section précédente montre l'importance de l'interaction des *nœuds de contexte* dans la composition d'une *politique de contexte*. Cet élément principal doit donc garantir l'évaluation, l'interprétation et l'acheminement de la qualité de contexte dans les différentes couches de la politique de contexte.

Pour ce faire, nous définissons dans une première approche les besoins de gestion de qualité dans un *nœud de contexte*. Le premier besoin consiste à préserver la compatibilité entre un nouveau nœud de contexte (supportant la QoC) et un nœud COSMOS standard (sans QoC). Le nouveau composant devra au moins contenir les sous-composants MessageManager et ActivityManager. Il doit également avoir des interfaces de type `Pull, Push` pour se connecter aux autres composants.

3.2.2 Nœud de qualité de contexte

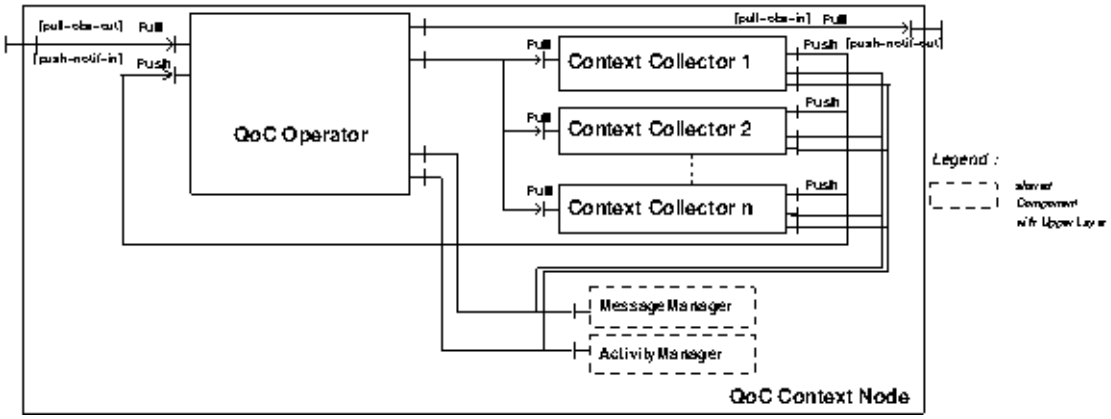


FIGURE 3.4 – Architecture du nœud de la qualité de contexte QoC Context Node

Nous avons défini un nœud de contexte COSMOS dédié à la gestion de la qualité de contexte qui respecte les exigences définies à la section 3.2.1. Il s'agit du QoCContextNode représenté à la figure 3.4 ; il se compose des mêmes éléments que le nœud de contexte standard à l'exception de l'*opérateur* qui est remplacé par un élément dédié à la gestion de la qualité. Le nœud de qualité de contexte gère la compatibilité avec le reste de l'architecture et laisse donc la gestion de qualité au composant dédié.

Nous avons construit également une famille d'opérateurs de qualité compatibles avec les nœuds COSMOS spécialisée dans l'inférence de qualité (voir Figure 3.5). Ces opérateurs héritent de l'opérateur de contexte de COSMOS et fournissent les fonctions mathématiques nécessaires à la combinaison de valeurs de qualité de contexte.

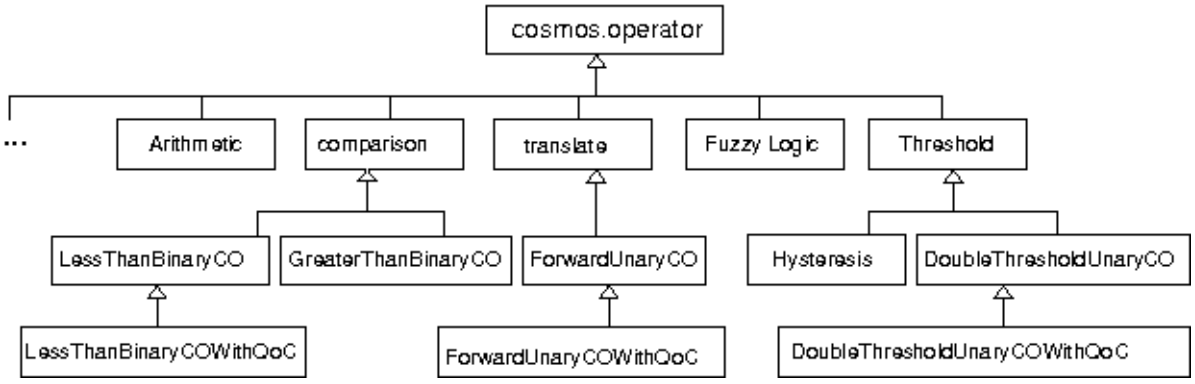


FIGURE 3.5 – Diagramme de classe de la famille des QoCOperator

3.2.3 Opérateur de qualité

L'opérateur de qualité QoCOperator est un composant composite appartenant au nœud de gestion de qualité de contexte QoCContextNode. Son rôle est l'extraction des informations requises, le calcul de la qualité et et le formatage des messages acheminés à l'aide du Message Manager (cf. Figure 3.4).

Les informations de QoC (qui font partie des informations de contexte) sont transmises au QoCParameterOperator spécifique pour permettre le calcul des valeurs de QoC. Ensuite, une fois calculées, ces valeurs peuvent être soit ajoutées dans un morceau de message déjà existant ou, profitant de la souplesse de COSMOS, ils peuvent être placés dans un nouveau morceau de message dédié à des informations communes comme l'estampille de temps par exemple. Dans ce mode, tous les messages d'information sur le contexte sont enrichis avec des méta-données de QoC.

Comme montré à l'intérieur du composant 3.6, les informations de contexte qui arrivent des nœuds fils du *QoCContextNode* ou des *Context Collectors* arrivent des interfaces externes (`pull-obs-in` ou `push-notif-out`) et passent au composant QoCAwareOperator qui les analyse et extrait les informations utiles à la génération de qualité.

Le QoCAwareOperator gère la qualité du contexte suivant les cas présentés dans le tableau 3.1.

3.2.4 Opérateur de calcul des paramètres de qualité

Chaque QoCParameterOperator génère un paramètre de qualité spécifique comme la *exactitude, précision, fraîcheur*, etc. Le calcul de la qualité est effectué en se basant sur les définitions et les équations étudiées dans la section 2.3 du chapitre 2.

Le choix des composants QoCParameterOperator dépend de la construction de la politique de contexte ainsi que de l'application consommatrice de la qualité de contexte. Nous proposons à la figure 3.7 une première liste des opérateurs les plus utilisés dans les applications sensibles

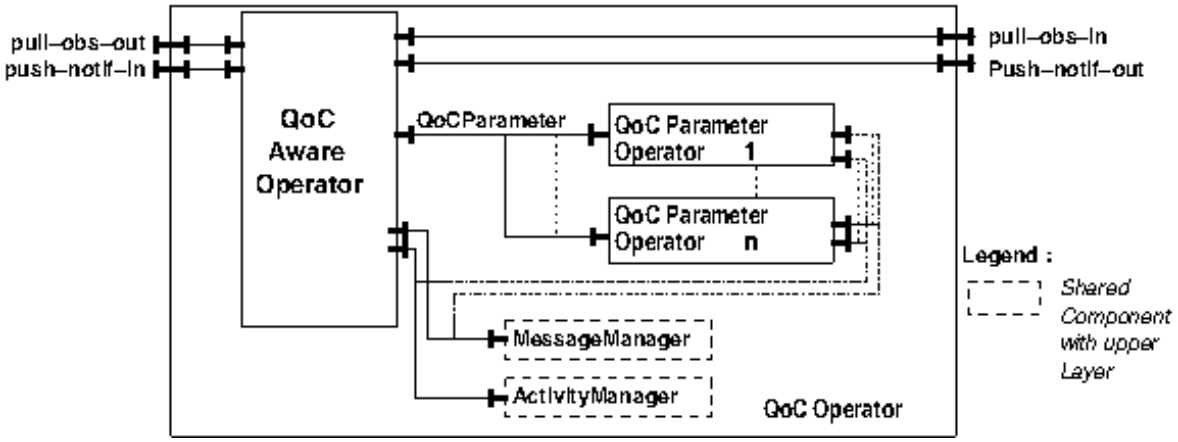


FIGURE 3.6 – Architecture d'un opérateur de QoC

Cas de gestion	Action de l'opérateur
Génération de la QoC	L'opérateur a comme mission principale la gestion de la la génération de la qualité à partir des informations de contexte. L'opérateur prépare les données et envoie une requête de calcul de la qualité aux opérateurs de QoC
Filtrage de la QoC	L'opérateur joue le rôle de filtre de qualité afin d'alléger les messages envoyés aux nœuds parents. À partir d'un certain seuil de fraîcheur, par exemple, l'opérateur n'intègre plus cette qualité dans les messages.
Inférence de la QoC	L'opérateur a dans ce cas le rôle d'intégrateur, il infère les différents messages de qualité envoyés des nœuds fils et il implémente l'une des techniques étudiées dans la section 2.4.2
Mise à jour de la QoC	Dans le cas de l'acheminement de la qualité dans les informations de contexte collectées, peuvent se présenter de nouveaux éléments plus pertinents et qui peuvent améliorer la qualité. L'opérateur peut alors décider de régénérer la qualité en faisant appel aux opérateurs de type : QoCParameterOperator

TABLE 3.1 – Gestion de la QoC au sein de l'opérateur de contexte

au contexte mais d'autres opérateurs peuvent facilement être ajoutés.

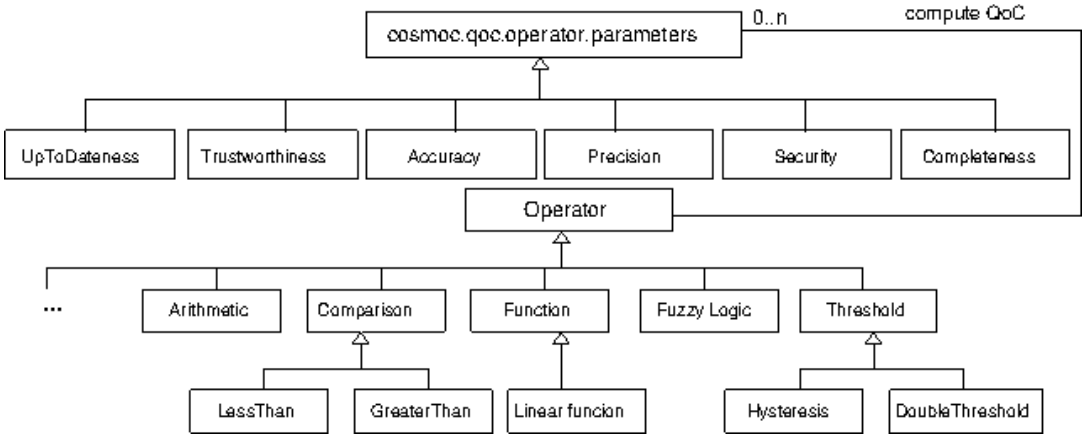


FIGURE 3.7 – Classes pour le calcul des paramètres de qualité

3.2.5 Exemple

La figure 3.8 présente un exemple minimal d'un nœud de qualité de contexte écrit en `Fractal ADL`. Nous avons simplifié la construction de l'exemple afin de montrer le mécanisme de création de la qualité. Toutefois un exemple complet est présenté dans l'annexe B.1. Le nœud de contexte se compose d'un opérateur ForwardUnaryCO qui envoie les messages collectés au nœud parent et de deux opérateurs de qualité : **complétude** (`parameter`) et **fraîcheur** (`parameterbis`). Ces opérateurs reçoivent des informations constantes respectivement le nombre de données utiles collectées, $param = 2$ attributs, et *l'estampille des informations*, $param = 2.5sec$.

```
<definition name "helloworld.HelloWorld" extends "cosmos.core.ContextNode">
        <component name "collector" definition "helloworld.CollectorHello
                                        (collectorname CollectorHello)"/>
        <component name "operator" definition "helloworld.ForwardUnaryCO
                                        (nodename ForwardHello)"/>

        <component name "parameter" definition "cosmos.qoc.operator.QocParameterOperator
                                        (param >'2', nodename >QocParameterOperator)"/>
        <component name "parameterbis" definition "cosmos.qoc.operator.QocParameterOperator
                                        (param >'2.5', nodename >QocParameterOperatorbis)"/>

        <binding client "operator.pull obs in" server "collector.pull obs out"/>
        <binding client "collector.push notif out" server "operator.push notif in"/>
        <binding client "collector.message manager" server "message manager.message manager"/>

        <binding client "operator.par1" server "parameter.getvalue"/>
        <binding client "operator.par2" server "parameterbis.getvalue"/>

</definition>
```

FIGURE 3.8 – Description en Fractal ADL d'un composant Helloworld utilisant COSMOS

Ce travail de conception a donné lieu à la publication [Abid et al., 2009b], qui est jointe à la

fin de ce manuscrit. Par la suite, nous avons amélioré cette architecture initiale.

3.3 Opérateur de qualité de contexte amélioré

L'architecture présentée dans la section précédente est utile pour les applications qui s'intéressent à la QoC en même temps qu'aux informations de contexte elles-mêmes, c'est-à-dire quand les informations de QoC sont systématiquement nécessaires. En conséquence, les paramètres de QoC sont fortement liés à ce contexte. Il en résulte des analyses plus fiables et précises. Toutefois, cette méthode nécessite du temps et des ressources pour créer un nouveau morceau de message ou d'en mettre à jour un qui existe déjà pour chaque information de contexte. Ceci augmente également le coût de la transmission de cette information.

Afin de permettre une gestion efficiente de la qualité de contexte, nous avons créé un nouveau nœud de contexte (figure 3.9), qui tient compte des différents cas d'utilisation de la qualité.

3.3.1 Transfert des informations de QoC

Nous nous attachons à proposer une plateforme la plus flexible possible en proposant deux modes pour transférer les informations de QoC et un mode permettant de transmettre le contexte sans QoC attachée lorsque cela n'est pas nécessaire. Dans le premier mode, il s'agit d'attacher les informations de QoC en tant que méta-données au contexte avant de les transmettre aux couches supérieures. Le deuxième mode transmet les informations de QoC séparément du contexte lui-même. Le dernier mode correspond à un mode sans QoC permettant de suspendre les traitements relatifs à la QoC lorsque cela n'est pas nécessaire ou pour des raisons de performance par exemple.

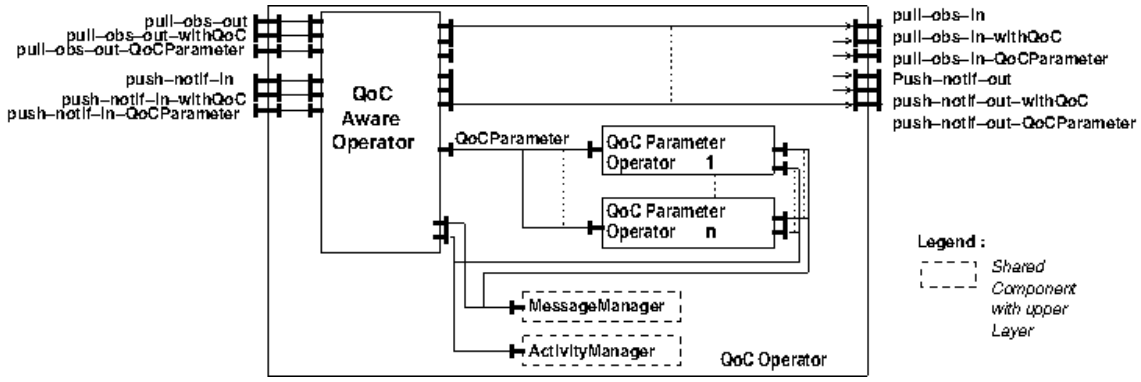


FIGURE 3.9 – Version améliorée de l'opérateur QoCOperator

3.3.2 Ajout de la QoC aux informations de contexte initiales

Les informations de QoC sont transmises à des opérateurs de type QoCParameterOperators pour permettre le calcul des paramètres de QoC. Ensuite, les valeurs des paramètres de QoC

peuvent être ajoutées dans un morceau de message (`message chunk`) existant, ou bien, profitant de la flexibilité de COSMOS, dans un nouveau morceau de message regroupant des méta-données de QoC courantes telles qu'une estampille. (cf. Figure 3.10-1) Dans ce mode, tous les messages contenant des informations de contexte sont enrichis des méta-données de QoC. Ce mode est donc recommandé lorsque les applications ont besoin de connaître, de la manière la plus précise possible, la QoC concernant chaque information de contexte. Cependant, cela nécessite des ressources en temps de calcul et en mémoire pour la gestion et la transmission des morceaux de message.

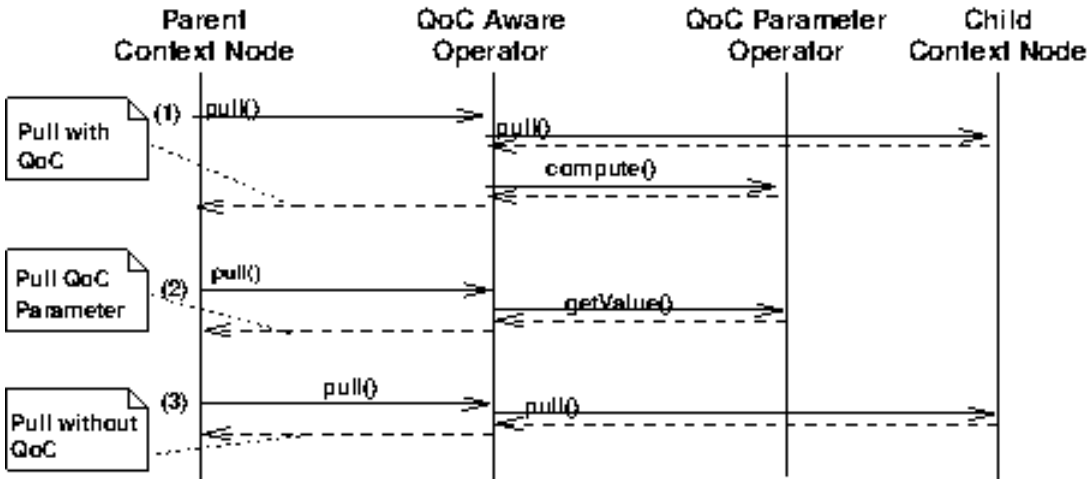


FIGURE 3.10 – Diagramme de séquence pour le mode Pull

3.3.3 Transmission de la QoC séparément

Périodiquement ou bien sur demande, l'information de QoC peut être transmise seule aux couches supérieures. (cf. Figure 3.10-2) Ce mode est bien adapté pour des applications ne nécessitant pas l'information de QoC avec une forte priorité ou, à l'inverse, ayant besoin de la QoC au plus tôt même si un nouveau contexte n'a pas été généré. Dans ce mode, la QoC est fournie à la demande de l'application (mode Pull) ou bien dans un rapport périodique (mode Push) ; dans les deux cas, ceci n'engendre pas un coût significatif. La limitation de ce mode est que l'information de QoC n'est pas fortement liée à une instance de contexte particulière. L'information de QoC qui est transmise peut correspondre à la dernière information de QoC calculée ou bien à une moyenne des valeurs calculées mais non encore transmises.

3.4 Architecture finale : une nouvelle approche

L'architecture proposée depuis le début du chapitre respecte les contraintes établies et atteint les objectifs de préservations de compatibilité avec le nœud de contexte standard ainsi que

l'interconnexion avec les interfaces. Toutefois, pour passer de l'échelle de prototype à l'échelle de la famille des opérateurs standard que peut compter le intergiciel COSMOS, il y a une implémentation considérable à mettre en œuvre pour mettre la famille d'opérateurs de QoC au même niveau. D'autant que le travail nécessaire est proportionnel à l'évolution même du intergiciel.

Nous avons décidé de repenser la structure du nœud de qualité tout en préservant les avantages aquis. La Figure 3.11 montre le résultat de l'architecture finale du composant de qualité de contexte. Ce composant composite a été conçu pour utiliser directement les *opérateurs de contexte* standards de COSMOS et les *opérateurs de paramètre de qualité*.

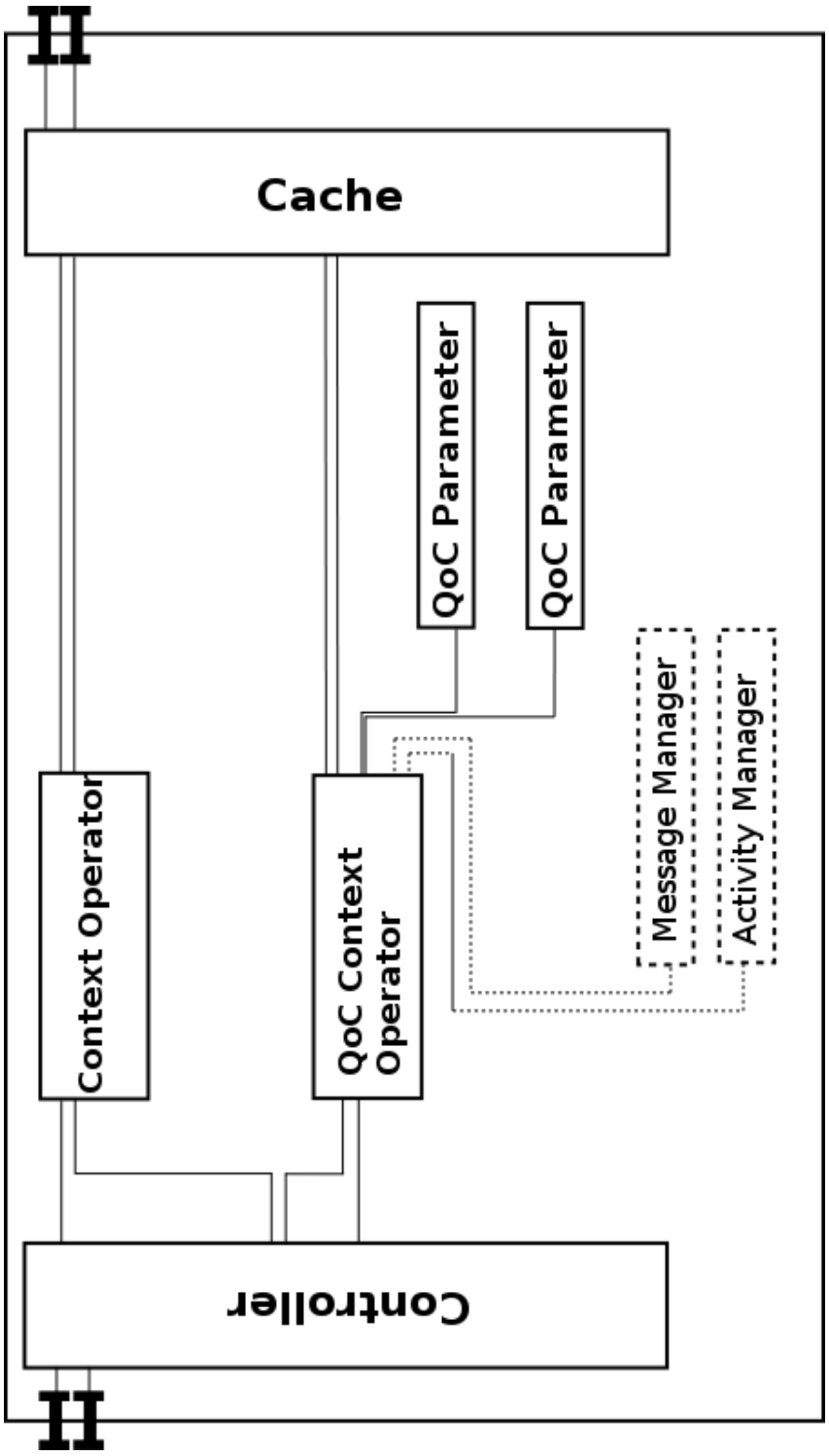


FIGURE 3.11 – Architecture du nœud de gestion de la QoC dans COSMOS

3.4.1 QoC Context Operator

La gestion de la qualité est maintenant totalement séparée dans l'opérateur de qualité qui s'interface avec les opérateurs de paramètre avec plus de liberté puisqu'il n'y a plus de contrainte de compatibilité. En effet, les interfaces standard sont prises en charge par l'opérateur de contexte.

Le QocContextOperator a pour rôle le calcul de la qualité qui arrive des nœuds fils mais également des informations de contexte générées par l'opérateur du nœud de contexte de qualité lui-même. Les informations sont obtenues par le composant de Cache et le composant Controller

Le Composant a une relation étroite avec les QoCParameterOperator (voir figure 3.11) et il est implémenté on considérant les demandes de l'application sensible au contexte tout comme le composant ContextAwareOperator de l'ancienne approche.

3.4.2 Cache

Lors de l'introduction du QoCOperator dans l'architecture initiale de COSMOS, le composant a besoin éventuellement des informations de contexte acheminées par les nœuds fils (par notification ou observation) tout autant que l'opérateur standard. Or les interfaces de notification (push-notif-out) et d'observation (pull-obs-in) ne permettent pas la connexion avec un seul composant.

C'est pourquoi le composant Cache suit le patron de conception de ChildFaçade, issu du patron de Façade [Gamma et al., 1995], et représente un ensemble de nœuds enfants. Ce composant a pour rôle l'interconnexion avec les nœuds fils et de faire une copie de chaque flux de messages arrivant d'une action Push ou Pull et de la transmettre aussitôt à l'opérateur ContextOperator. Du point de vue de l'opérateur, le rôle du composant Cache est transparent et n'affecte pas ses interactions.

Le Cache sert aussi le même flux de messages à la demande de l'opérateur de qualité QoCContextOperator comme s'il était directement interconnecté avec les nœuds fils. Ceci lui permet d'extraire les informations de contexte utiles à la gestion de la qualité dans les différentes formes indiquées précédemment dans le tableau 3.1.

3.4.3 Controller

Le composant Controller est un composant de type ParentFaçade, issu du patron de Façade [Gamma et al., 1995], pour représenter un ensemble de nœuds parents.

Dans le cas où la qualité n'est pas requise par les nœuds parents, ce composant a pour rôle la transmission du flux des messages de l'opérateur ContextOperator vers ces nœuds tout en transparence. Par contre, si la qualité est exigée, il demande à l'opérateur de QoC les informations de qualité. Par conséquent, il inclut un message contenant la qualité obtenue du QoCContextOperator dans le flux des messages à envoyer vers les nœuds parents.

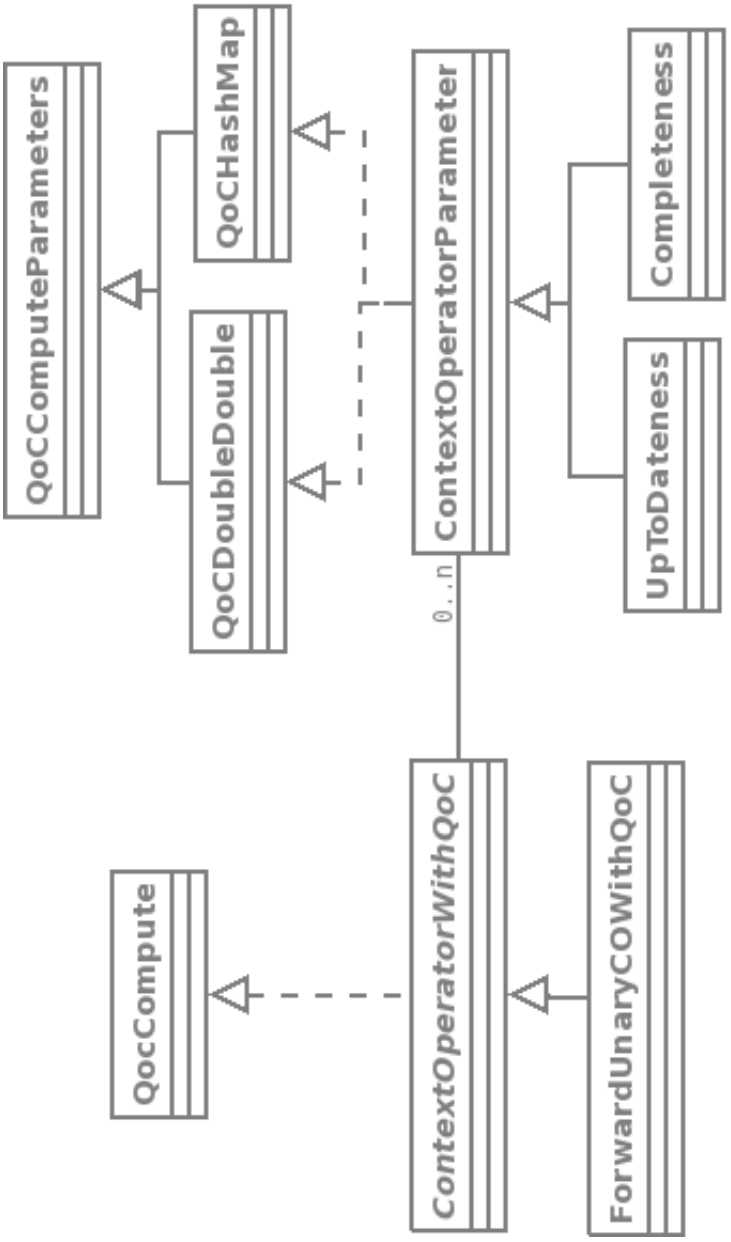


FIGURE 3.12 – Relations entre QoCContextOperator et ContextOperatorParameter pour le Calcul de la QoC

3.4.4 Protocole d'échange d'informations entre les nouveaux composants

L'architecture de gestion de qualité de contexte est implémentée avec l'intergiciel COSMOS. Elle bénéficie donc de son protocole de communication et peut échanger les mêmes types de données entre les différents composants.

Chaque composant possède une membrane construite avec l'intergiciel orienté message (MOM) Dream qui fournit les mécanismes d'observation (`pull`) et de notification (`push`) à travers le composant MessageManager. [11]

3.4.4.1 Relation entre Cache et QoCOperator

Le composant Cache implémente l'interface `qoc-data` (voir listing 3.1) qui permet l'interconnexion avec le composant QoCOperator. On peut remarquer la déclaration d'une demande de l'interface `"qoc-data"` à la ligne 14 du listing 3.3.

```
1  @Component(provides = {
2  @Interface(name = "pull-obs-out", signature = Pull.class),
3  @Interface(name = "push-notif-in", signature = Push.class),
4  @Interface(name = "qoc-data", signature = QocData.class)
5  })
6  @Membrane(controller = "activeDreamPrimitive")
7
8  public class Cache
9      implements Pull, Push,
10     QocData<Message>, PrepareStopLifeCycleController,
11     NeedAsyncStartController {
12 ...
13 }
```

Listing 3.1 – Cache

3.4.4.2 Relation entre QoCOperator et QoCParameterOperator

Le QoCParameterOperator implémente l'interface `QoCInterface` (listing 3.2) afin de présenter la qualité de contexte soit en la calculant, soit en donnant directement les résultats calculés. Une liaison avec le composant QoCOperator s'effectue en demandant l'interface `"parameter"` (ligne 8 du listing 3.3) qui permet la liaison avec un ou plusieurs composant de type QoCParameterOperator.

```
1  package cosmos.qoc.operator;
2
3  public interface QoCInterface <Output,Input> {
4          Output compute(Input input);
5                  Output getValue();
```

11. http://picoforge.int-evry.fr/projects/svn/cosmos/Manual/description.html

```
6                   }
```

Listing 3.2 – QoCInterface

```
1  @Component(provides = {
2  @Interface(name = "compute-qoc", signature = ComputeQoc.class)})
3  @Membrane(controller = "primitive")
4
5  public class QocOperator
6      implements ComputeQoc<Message, Message> {
7
8  @Requires(name = "parameter-",
9  cardinality = Cardinality.COLLECTION,
10    contingency = Contingency.OPTIONAL)
11    protected HashMap<String, QoCInterface>
12    parameter = new HashMap<String, QoCInterface>();
13
14 @Requires(name = "qoc-data",
15     cardinality = Cardinality.COLLECTION)
16     private Hashtable<String, QocData> qocData =
17         new Hashtable<String, QocData>();
18 }
```

Listing 3.3 – QoCOperator

3.4.4.3 Relation entre QoCOperator et Controller

De la même manière, l'implémentation et demande s'effectue autour de l'interface compute-qoc entre QoCOperator (listing 3.3 ligne 2) et le Controller (listing 3.4 ligne 18).

```
1  @Component(provides = {
2
3  @Interface(name = "pull-obs-out", signature = Pull.class),
4  @Interface(name = "push-notif-in", signature = Push.class)})
5  @Membrane(controller = "activeDreamPrimitive")
6
7  public class Controller
8      implements Pull, Push,
9      PrepareStopLifeCycleController,
10     NeedAsyncStartController{
11
12
13 /** Fractal client interfaces for computing context's
14 * Qoc data from the QocOperator <p> J2ME/CLDC:
15 * <tt>Hashtable</tt> instead of <code>Map</code>
16 * because <code>Map</code>not available.
17 */
18 @Requires(name = "compute-qoc", cardinality =
```

```
19            Cardinality.COLLECTION)
20  private Hashtable<String, ComputeQoc<Message, Message>> computeQoc =
21      new   Hashtable<String, ComputeQoc<Message, Message>>();
22  }
```

Listing 3.4 – Controller

3.5 Gestion fine de la qualité de contexte

Pour réussir à comparer la qualité des informations de contexte issues de différents capteurs qui envoient le même type d'information, nous proposons de construire une qualité globale qui tient compte de l'importance de chaque paramètre de qualité. Ceci permet d'avoir une comparaison plus conforme aux attentes de l'application sensible au contexte. La construction se fait en comparant les paramètres entre eux deux à deux pour donner un ordre d'importance relatif. À chaque comparaison, on rajoute au poids relatif du paramètre un poids suivant l'ordre d'importance.

Cette proposition a fait l'objet de la publication [Abid and Chabridon, 2011] qui est jointe à la fin de ce manuscrit. Nous invitons le lecteur à s'y reporter pour plus de détails.

3.6 Conclusion

Dans ce chapitre, nous avons étudié l'architecture des composants de COSMOS afin de construire un nouveau nœud qui permet de **gérer la qualité de contexte dans l'intergiciel**. Un premier objectif était de préserver la compatibilité avec les composants standard (sans QoC) et de proposer une famille de nœuds de contexte avec QoC prête à être utilisée dans la construction des politiques de contexte. La gestion de la qualité est traitée de manière à être efficiente et flexible en proposant deux modes de transfert de la qualité permettant de répondre à différents cas d'utilisation.

Nous proposons également les mécanismes nécessaires pour assurer une gestion de la QoC au sein de l'arborescence de la politique de contexte en traitant de méta-données de qualité provenant à la fois des composants enfants situés sur les couches inférieures et des traitements réalisés par l'opérateur du nœud lui-même.

Le créateur d'applications sensibles au contexte doit traiter les différentes étapes suivantes :

- la construction de la politique de contexte sous la forme d'une arborescence de nœuds de contexte
- la conception de la gestion de la qualité de contexte en choisissant les paramètres de qualité et les opérateurs de QoC après avoir déterminé les sources de l'information de contexte
- la définition des interactions de l'application avec la politique de contexte.

Si on rajoute la complexité de la construction des composants composites (voir en annexe A1 un exemple d'un nœud de contexte et de l'arborescence du code du projet associé), la tâche devient difficile pour les concepteurs d'applications. C'est pour cette raison que nous avons inscrit notre travail dans une démarche d'ingénierie dirigée par les modèles que nous présentons

au chapitre suivant. La gestion de la qualité des informations de contexte bénéficie alors de l'éco-système COSMOS dans lequel l'on trouve notamment l'outil cosmos2code qui permet de générer automatiquement une partie du code des applications.

Chapitre 4

Ingénierie dirigée par les modèles pour la construction d'applications ubiquitaires sensibles à la qualité du contexte

Sommaire

La complexité inhérente au développement d'applications sensibles au contexte nécessite l'utilisation de techniques d'ingénierie logicielle avancées comme les composants et les architectures logicielles ainsi que des mécanismes adaptés à l'évolution continue du contexte. Nous présentons dans ce chapitre nos travaux visant à tirer profit de l'ingénierie dirigée par les modèles afin de faciliter le développement d'applications ubiquitaires. Nous commençons par discuter des motivations de nos travaux et par les illustrer par une application de vente Flash dans un centre commercial. Nous présentons ensuite une vue globale du processus de conception de la sensibilité au contexte des applications ubiquitaires qui inclut la prise en compte des exigences de QoC et nous le relions aux plateformes logicielles développées par l'équipe MARGE. Finalement, nous spécifions le modèle de sensibilité au contexte de l'application de vente Flash et exprimons la politique de gestion de contexte incluant la prise en compte de la QoC.

Ce chapitre a donné lieu aux publications [Chabridon et al., 2011a], [Chabridon et al., 2012a] et [Chabridon et al., 2012b].

4.1 Motivations

Comme évoqué précédemment (voir section 2.5), les principales familles de modélisation de contexte sont la définition de profils ([Klyne and al., 2007]), les bases de données ([Henricksen and Indulska, 2006]), les ontologies ([Wang et al., 2004]) et l'IDM.

Dans notre approche, nous avons choisi de suivre les principes de l'IDM qui permet de définir des liens entre la modélisation de contexte, qui permet d'exprimer des situations de contexte, et la modélisation de la sensibilité au contexte, qui associe ces situations de contexte avec des entités de l'application. ContextUML [Sheng and Benatallah, 2005] est l'un des premiers modéles dédiés au domaine de la sensibilité au contexte. Il définit un méta-modèle pour la modélisation de services Web sensibles au contexte. Les éléments propres aux services Web, tels que *Service*, *Operation* et *Message*, sont représentés dans le modèle ainsi que les mécanismes d'adaptation comme *Binding* ou *Triggering*. ContextUML met l'accent sur les mécanismes d'adaptation plutôt que sur la modélisation de contexte, et ne traite pas de la qualité de contexte. Notre travail, quant à lui, permet aux concepteurs de l'application d'exprimer des contextes de haut niveau tels que des situations déterminées à partir d'obervations de contexte distribuées et d'inclure dès le début du processus l'analyse de la QoC. Cela contribue à améliorer la sensibilité au contexte de l'application et par conséquent à améliorer la qualité des adaptations effectuées sur l'application.

Nous proposons de commencer par une étape de méta-modélisation qui permet d'ajouter de nouveaux types d'entités, d'observables, de collecteurs, etc. À partir d'un méta-modèle de sensibilité au contexte, un modèle spécifique à une application est défini et permet de transmettre à la plateforme de gestion de contexte la description de sa sensibilité au contexte.

4.2 Scénario applicatif

Considérons le scénario de vente Flash dans un centre commercial qui suit. *Marie se rend dans le plus grand centre commercial de la région. Elle possède un téléphone mobile avec navigation GPS, communication 3G, WiFi et Bluetooth. Quand elle se gare sur le parking extérieur à 10 heures, elle reçoit un message l'informant de la disponibilité d'un nouveau service d'annonce de* ventes Flash *et lui proposant de le télécharger. Comme Marie est une cliente privilégiée de ce centre commercial, elle s'est déjà inscrite pour bénéficier de services et de conseils en indiquant ses goûts et produits préférés. Juste après avoir téléchargé la nouvelle application, elle reçoit une annonce indiquant qu'il lui reste 1 heure pour profiter d'une vente Flash en cours dans son magasin de sport préféré. À ce moment-là, la localisation de Marie n'est pas connue de manière exacte. Dans ce cas, l'application indique à l'écran la position estimée de Marie ainsi que la position du magasion, sans indication supplémentaire. Dans l'après-midi, Marie reçoit alors une nouvelle annonce pour une vente Flash qui va commencer dans un magasin venant juste d'ouvrir et qu'elle ne connaît pas encore. La vente Flash est exceptionnelle et ne dure que 15 minutes. La couverture réseau est à ce moment-là très bonne et la localisation de Marie est estimée de manière très précise grâce aux informations récupérées par le téléphone. Une carte du centre commercial apparaît alors sur l'écran du téléphone, centrée sur la position actuelle de Marie. Un tracé sur la carte lui indique clairement le chemin à suivre pour se rendre dans le magasin*

où se déroule la vente flash. Marie décide alors de prendre la direction de ce nouveau magasin. Tout au long du parcours, elle est informée du temps qu'il reste avant la fin de la vente Flash et la carte avec l'itinéraire est rafraîchie régulièrement. Grâce aux indications précises fournies par le service de vente Flash, Marie atteint le nouveau magasin dans les temps.

Ce scénario montre qu'il est essentiel qu'un service exploitant des informations décrivant le contexte d'un utilisateur puisse mesurer la qualité des informations de contexte. Par exemple, la localisation d'un utilisateur mobile peut être fournie par plusieurs techniques (p.ex. GPS, GPRS, Wifi), mais de manière plus ou moins fiable. Il est donc nécessaire de qualifier cette localisation. Nous proposons de prendre en compte des méta-données indiquant la qualité du contexte (QoC) à différentes étapes du processus de conception des applications sensibles au contexte. Plusieurs techniques de localisation pouvant être utilisées sur le téléphone, seule la position de l'utilisateur avec une qualité suffisante et la plus élevée parmi celles disponibles est effectivement affichée sur la carte. De plus, le service de vente Flash est adapté en fonction de la qualité de l'information de contexte qu'il utilise, à savoir la localisation. Une annonce de vente Flash est affichée sur le téléphone uniquement si la localisation est connue de manière suffisamment fiable. De plus, l'itinéraire précis est indiqué uniquement s'il peut être basé sur une bonne estimation de la position de départ de l'utilisateur. Il serait contre-productif de fournir des indications erronées à l'utilisateur et de l'envoyer dans une mauvaise direction. Dans le cas de la vente se déroulant dans le magasin de sport préféré de Marie, la qualité de la localisation de Marie n'est pas suffisante, le service n'affiche pas la carte détaillée avec l'itinéraire. Mais dans ce cas, Marie trouve facilement comment se rendre dans le magasin de sport et n'est pas gênée de ne pas avoir d'indications plus précises sur le chemin à suivre. Le niveau de service offert dépend ainsi de la qualité des informations de contexte.

Dans le reste de ce chapitre, nous présentons un processus de conception et un cadriciel générique pour manipuler la QoC. Pour valider notre approche, nous utilisons comme exemple fil rouge le prototype de l'application que nous avons développé pour ce scénario de vente Flash.

4.3 Processus de conception d'applications sensibles à la QoC

Dans cette section, nous décrivons le processus de conception que nous préconisons pour la construction d'applications sensibles au contexte, issu des travaux antérieurs de l'équipe MARGE[12] [Taconet et al., 2009]. Notre contribution porte sur les extensions apportées à ce processus pour la définition des exigences de QoC.

Le rôle de la gestion de contexte est de fournir des informations de contexte de haut niveau afin d'identifier des situations pouvant nécessiter une adaptation de l'application sensible au contexte. Cela consiste en trois tâches qui sont l'acquisition de données de contexte, le traitement de ces données par fusion, l'agrégation ou l'interprétation, et la présentation de situations de contexte pertinentes pour l'application. Notre processus de conception découple autant que faire se peut la conception de la gestion de contexte de la conception de l'application métier. Pour cela, nous ajoutons au domaine de la conception métier deux autres domaines dédiés : la gestion

12. http ://www-inf.it-sudparis.eu/MARGE

de contexte et la sensibilité au contexte, induisant les deux nouveaux rôles que sont le concepteur de la gestion de contexte et le concepteur de la sensibilité au contexte.

Nous avons suivi le modèle SPEM (Software Process Engineering MetaModel) [Object Management Group, 2008] pour définir les rôles, les activités et les résultats produits dans le domaine de la gestion de contexte. Le méta-modèle du gestionnaire de contexte utilisé, qui est COSMOS [Conan et al., 2007a] dans notre thèse, est en entrée de toutes les activités. Par conséquent, tous les modèles et implantions résultants sont conformes au méta-modèle de COSMOS et à son API. Le concepteur de la gestion de contexte définit les inférences d'informations de contexte de haut niveau telles que la distance entre deux localisations à partir de données brutes collectées par ce que nous appelons des *collecteurs de contexte*. Cela implique de définir des *nœuds de contexte* en utilisant un langage dédié au domaine de la gestion de contexte, à savoir le DSL COSMOS [13], dans lequel une inférence d'informations de contexte de haut niveau est une expression impliquant des données de contexte de plus bas niveau mettant en œuvre ce que nous appelons des *opérateurs de contexte*.

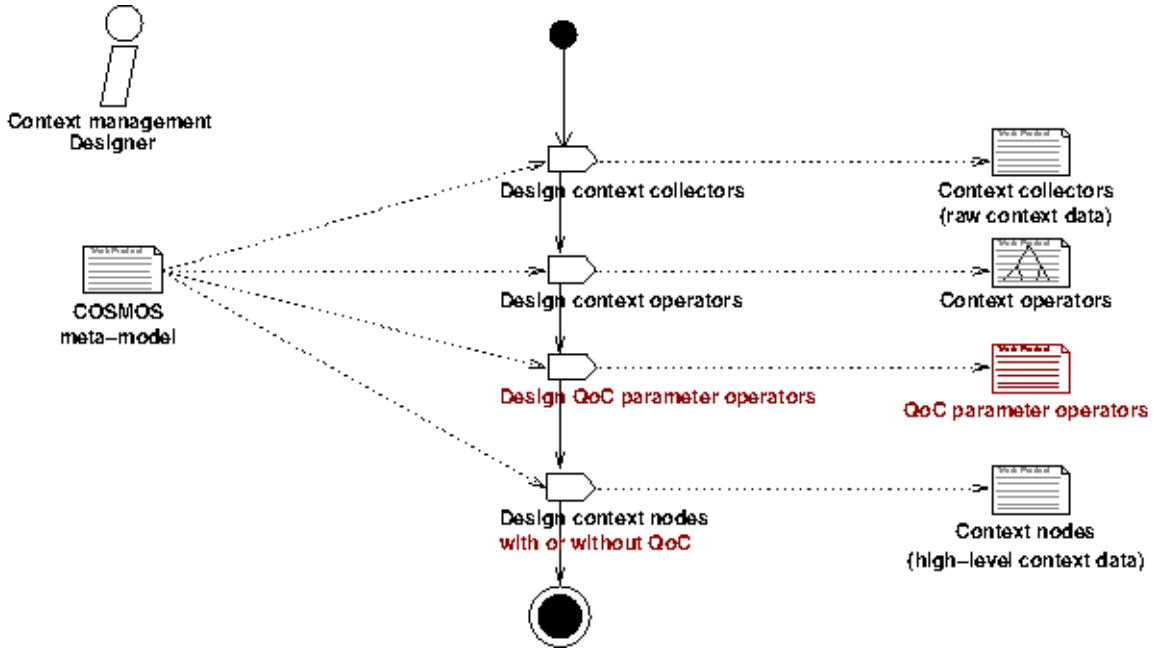


FIGURE 4.1 – Activités du concepteur de la gestion de contexte

Comme indiqué à la Figure 4.1, la première activité est la spécification de collecteurs de contexte à l'aide du DSL COSMOS et leur implantion en quelques lignes de code Java. Par exemple, le concepteur peut définir plusieurs collecteurs de contexte pour récupérer des positions brutes provenant de différents capteurs (p.ex. GPS, GPRS, Wifi) et différents équipements (p.ex. téléphones Android, iPhones). La seconde activité concerne la conception des opérateurs

13. http ://picoforge.int-evry.fr/projects/svn/cosmos

de contexte. La caractéristique importante de notre approche est que ces opérateurs permettent le calcul de la qualité de l'information de contexte en complément de la valeur elle-même de cette information de contexte, mettant ainsi en œuvre des *opérateurs de contexte sensibles à la QoC* et des *nœuds de contexte sensibles à la QoC*. Afin de rendre incontournable la qualification des informations de contexte, nous introduisons une troisième activité spécifique à la conception d'*opérateurs sur paramètre de QoC* qui calculent les méta-données de QoC (p.ex. la fraîcheur d'une information de contexte, fonction de son âge depuis la date de collecte et de sa durée de vie). La quatrième et dernière activité est constituée par la conception des entités représentant les informations de contexte de haut niveau à l'aide du DSL COSMOS. Cela revient à décrire les expressions de traitement des données de contexte. Ces expressions réutilisent les collecteurs de contexte, opérateurs de contexte et opérateurs sur paramètre de QoC conçus au cours des activités précédentes. Cela se fait grâce au DSL COSMOS qui permet la réutilisation par composition et évite ainsi au concepteur de la gestion de contexte de programmer ces expressions.

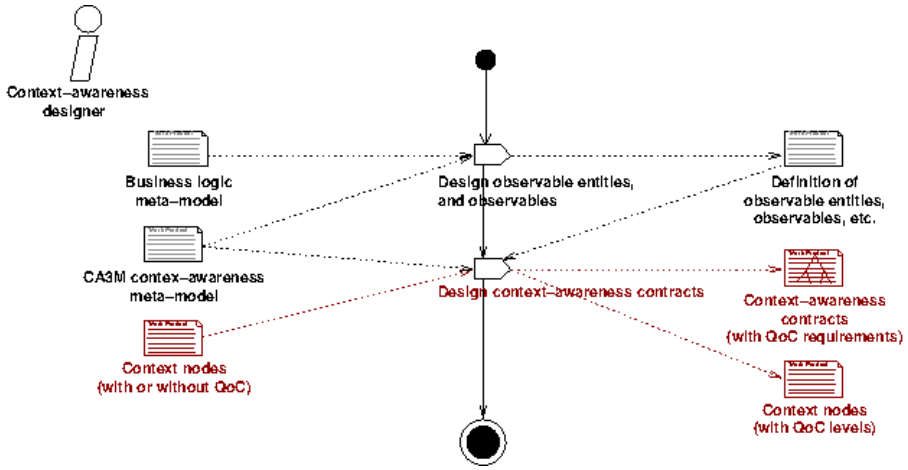


FIGURE 4.2 – Activités du concepteur de la sensibilité au contexte

La Figure 4.2 décrit les principales activités du concepteur de la sensibilité au contexte des applications. La première activité consiste en la spécification des entités devant être observées à l'exécution (que nous nommons *entités observables* ou *entités*), et la spécification de quelles données de contexte doivent être collectées ou inférées (appelées *observables* dans notre terminologie) sur ces entités. Par exemple, dans le cas de l'application de vente Flash, le client et les magasins sont des exemples d'entités à observer, la localisation du client et ses produits préférés étant des observables. Le méta-modèle de sensibilité au contexte CA3M [Taconet et al., 2009] ainsi que le méta-modèle de la logique métier des applications sont des entrées de cette première activité. En effet, le rôle du concepteur dans cette activité est d'étiqueter par des stéréotypes les éléments du modèle UML de l'application représentant les entités et les observables. La seconde activité consiste à faire le lien entre les artefacts produits par le processus de conception de la gestion de contexte et les artefacts de l'application métier au travers de la définition de contrats spécifiques. Par exemple, un contrat de sensibilité au contexte indique qu'une annonce de vente

Flash doit être déclenchée dès lors qu'une vente Flash correspondant aux préférences du client a lieu à proximité. Lors de la définition de ces contrats, le concepteur ajoute les exigences de QoC de l'application métier et configure les nœuds de contexte sensibles à la QoCavec les *niveaux de QoC* appropriés. Un niveau de QoC correspond à un niveau global de qualité calculé en agrégeant plusieurs paramètres de QoCtels que la fraîcheur et la confiance. Étant donné que les exigences de QoC sont propres à une application, la configuration de ces niveaux de QoC fait partie du domaine de la sensibilité au contexte et non de celui de la gestion de contexte.

4.4 Contrats de sensibilité au contexte et de QoC

Par analogie avec ce qui existe pour la qualité de service (QoS), [Buchholz et al., 2003] préconise la négociation de contrats de QoC. Nous proposons d'intégrer de tels contrats avec les contrats de sensibilité au contexte de COSMOS. Nous décrivons à la figure 4.3 le méta-modèle de sensibilité au contexte suivi par COSMOS, qui décrit différents types de contrat qui héritent du contrat de sensibilité au contexte. Le principal contrat mis en œuvre dans notre prototype est le contrat d'observation. Un contrat d'observation permet à l'application d'exprimer ses besoins concernant les informations de contexte qui sont pertinentes pour elle ainsi que les situations d'adaptation auxquelles elle est sensible et qu'elle souhaite détecter. L'observation collectée peut ainsi être qualifiée pour améliorer cette observation ainsi que les prises de décision qui en découlent. Les informations de qualité telles que l'exactitude, la précision, la fiabilité, la fraîcheur peuvent être exprimées dans les contrats de QoC et être collectées et traitées par la chaîne d'observation. COSMOS permet de considérer le gestionnaire de contexte comme un service d'observation pour les composants logiciels qui annoncent par contrat leurs exigences d'observation.

Un utilisateur de contexte peut ensuite vérifier que les informations de contexte fournies par les sources sont d'une qualité au moins égale au minimum défini dans le contrat. Il faut également prévoir la re-négociation des contrats permettant les modifications nécessaires en raison de la forte dynamicité d'un environnement pervasif.

Ainsi, la demande exprimée sur un contrat d'observation et de QoC d'une bande passante peut être enrichie par des caractéristiques liées à la fraîcheur de l'information (différence entre les dates de livraison et de collecte effective) ou par l'exactitude (en % par exemple). La contrainte sur la qualité des informations d'observation peut ensuite être utilisée lors de la recherche d'une source d'observation acceptable.

Lorsque plusieurs sources de contexte fournissent des informations sur une même entité observée, il est nécessaire de prévoir la manière dont doit se faire la sélection des sources de contexte à privilégier. Ceci repose en partie sur les contrats d'observation qui permettent de choisir les sources de contexte en fonction du niveau de QoC qu'elles peuvent garantir. Si plusieurs sources sont capables de satisfaire le contrat, il faut ensuite choisir des règles simples pour éviter une diminution des performances par des calculs d'inférence inutiles.

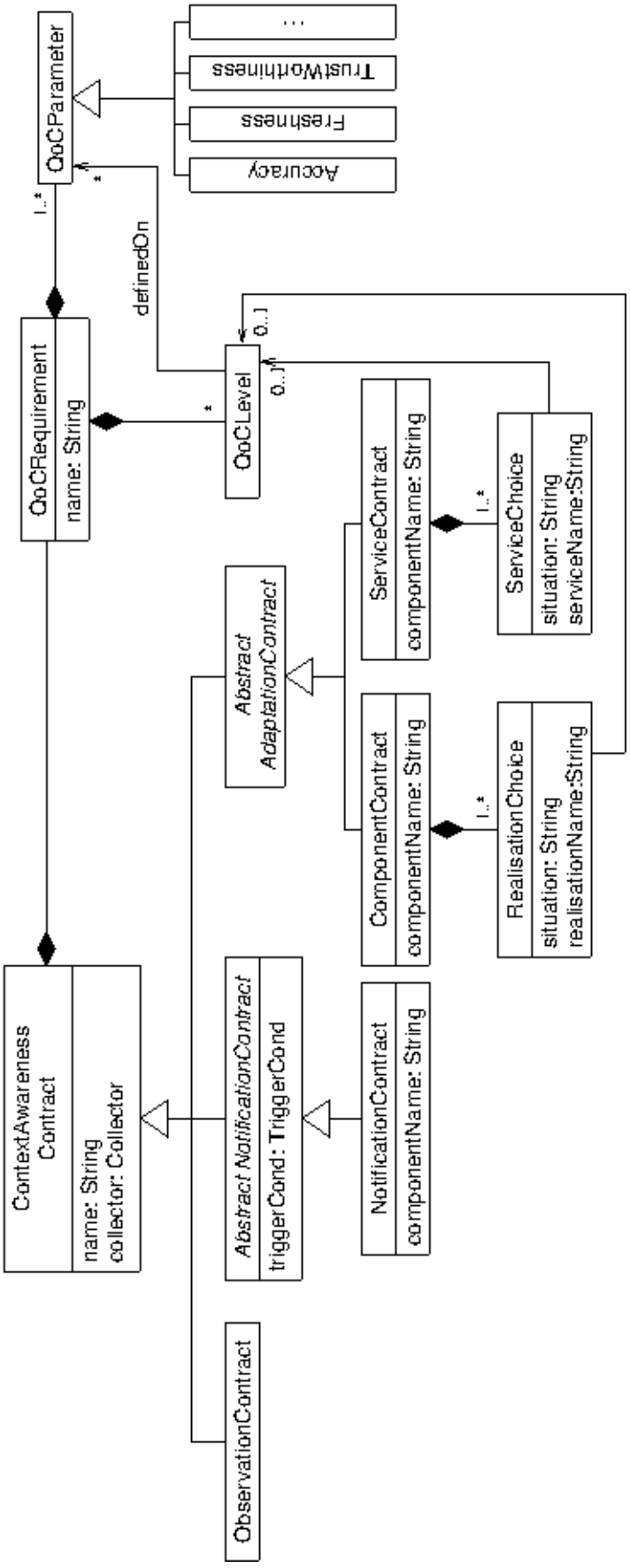


FIGURE 4.3 – Méta-modèle de sensibilité au contexte étendu avec la QoC

4.5 Lien avec les outils de l'éco-système de COSMOS

La Figure 4.4 présente une vue globale des plateformes logicielles associées avec le processus de conception que nous proposons. Elles constituent l'éco-système de COSMOS proposé par l'équipe MARGE. Concernant la gestion de contexte, nous utilisons comme vu précédemment le cadriciel COSMOS. Les politiques de contexte sont spécifiées dans le langage spécifique au domaine de la gestion de contexte proposé avec COSMOS, COSMOS DSL, derivé du méta-modèle de COSMOS. À partir de cette spécification, l'outil cosmos2code permet de générer la plus grande partie du code des nœuds de contexte, la définition de l'architecture orientée composant et le code source des classes. Chaque nœud de contexte contient un opérateur de contexte mettant en œuvre les instructions pour le calcul des données de bas et de haut niveau et leurs méta-données de QoC associées.

Concernant la sensibilité au contexte de l'application métier considérée, nous nous basons sur l'intergiciel CA3M (Context-Aware Middleware based on a context-awareness Meta-Model) [Taconet et al., 2009]. Les concepts proposés par CA3M permettent aux concepteurs de la sensibilité au contexte de relier les entités de l'application avec les sources de données de contexte via des politiques de contexte et de définir des contrats spécifiques entre l'application et le gestionnaire de contexte.

Dans ce chapitre, nous montrons comment compléter ces contrats avec les exigences de l'application en termes de QoC attendue. Ces exigences de QoC sont prises en compte lors de construction des politiques de contexte et de la génération du code des nœuds de contexte. Les exigences de QoC sont utilisées pour définir à la fois les composants de type *QoCParameterOperator* qui sont inclus dans la hiérarchie de nœuds de contexte et les niveaux de QoC qui seront appliqués pour filtrer les données de contexte préalablement au déclenchement d'une action d'adaptation de l'application.

4.6 Politique de contexte pour l'application de vente Flash

La figure 4.5 montre le graphe de la politique de contexte de l'application de vente Flash avec un focus sur la partie concernant le calcul de la localisation ajustée. La figure 4.6 donne la description à l'aide du DSL COSMOS correspondante. Ce sous-arbre n'est pas décrit entièrement : nous présentons uniquement les parties spécifiques à l'application. De plus, les nœuds provenant de la bibliothèque générique de COSMOS (indiqués en italique) sont décrits ailleurs. Cette spécification est ensuite fournie en entrée à l'outil cosmos2code pour générer l'architecture logicielle de l'application sous forme de composants ainsi que le squelette des composants primitifs. Le programmeur de l'application n'a alors plus qu'à compléter le code de l'application en écrivant quelques lignes de code Java pour les composants primitifs.

La politique de contexte exploite les différentes techniques de localisation pouvant être accessibles depuis un téléphone mobile telles que les réseaux 3G, la radio sans fil et le GPS, et effectue le choix d'une localisation stabilisée. Ce choix est guidé par la QoC de l'information de localisation. Nous considérons trois paramètres de QoC dans cette politique de contexte : l'**exactitude** (notée A pour *accuracy* sur le graphe), la **fraîcheur** (notée F) et la **confiance**

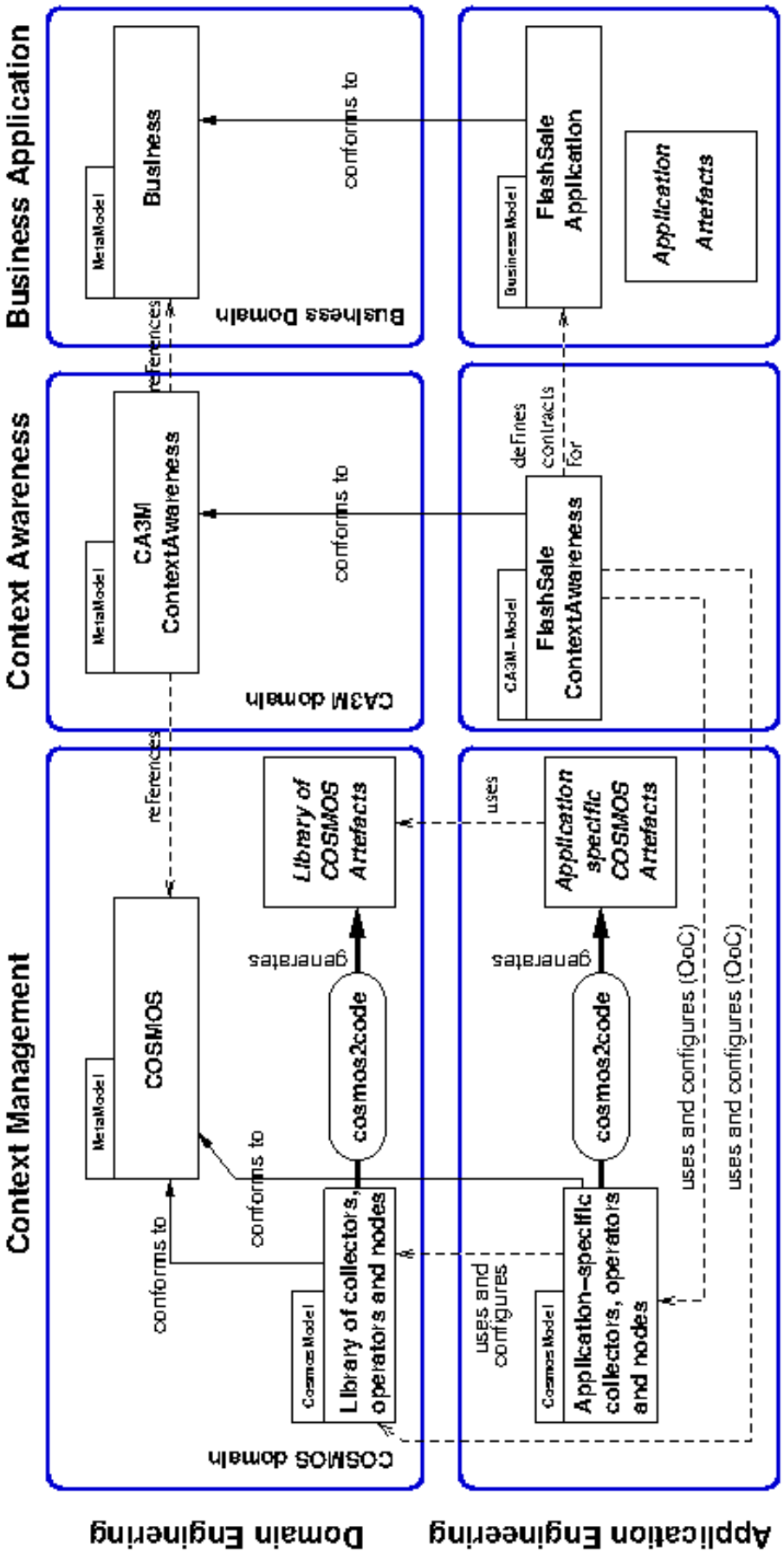


FIGURE 4.4 – Processus de conception et plateformes logicielles associées

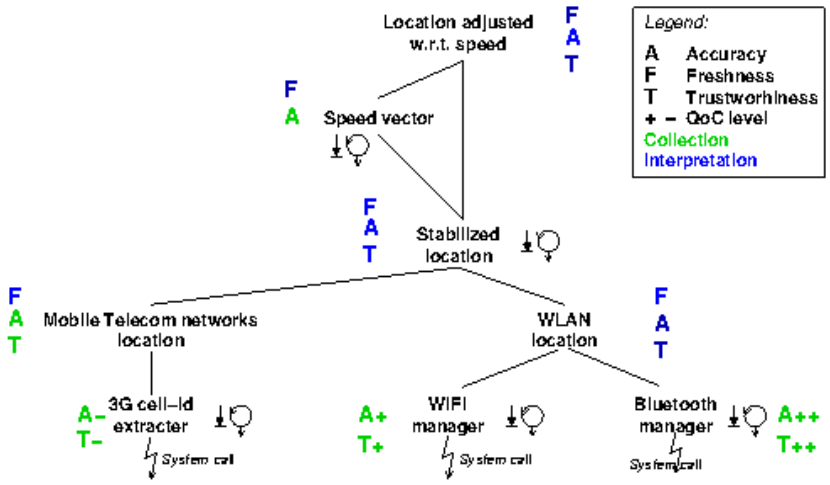


FIGURE 4.5 – Politique de contexte avec QoC pour l'ajustement de la localisation

(notée T pour *trustworthiness*). Pendant le processus d'inférence de la localisation, le nœud de contexte Stabilized location choice détermine la meilleure localisation en fonction des valeurs de la confiance, de la fraîcheur et de l'exactitude, dans cet ordre.

4.7 Conclusion

La production d'applications sensibles au contexte est une tâche complexe pour laquelle l'ingénierie dirigée par les modèles apporte des mécanismes et des outils essentiels. Nous avons présenté dans ce chapitre nos travaux en modélisation de la sensibilité au contexte avec prise en compte de la qualité de contexte. Notre contribution s'appuie sur une chaîne d'outillage complète et intégrée développée par l'équipe MARGE comprenant un méta-modèle de sensibilité au contexte (CA3M), un langage dédié à la gestion de contexte (COSMOS DSL) associé à un outil de génération de code (cosmos2code) et des artefacts déployés à l'exécution.

Nous avons démontré nos contributions sur l'exemple d'une application de vente Flash. De même que pour de nombreuses applications mobiles, la connaissance de la localisation de l'utilisateur est essentielle pour une vente Flash, qui se déroule en un lieu précis et pendant une période de temps limitée. Nous considérons que cette information de localisation est au cœur de l'application. Nous donnons ainsi à l'utilisateur la maîtrise de sa localisation qui est collectée et mesurée uniquement sur le téléphone de l'utilisateur, ce qui garantit le respect de sa vie privée. De plus, l'information de localisation pouvant être évaluée de manière plus ou moins précise, cela nécessite donc de prendre en compte les éventuelles incertitudes qui peuvent l'impacter. C'est pourquoi nous avons identifié trois critères de QoC comme étant particulièrement pertinents pour la localisation, à savoir l'exactitude, la fraîcheur et la confiance. D'autres paramètres de QoC pouvant être calculés par notre gestionnaire de contexte, des paramètres supplémentaires peuvent être ajoutés en tant que méta-données associées à la localisation.

```
package evry2mallqoc;
chunk position = {
    classname : evry2mallqoc.PositionChunk,
    typeparam : evry2mallqoc.PositionInfo
}
message location = {
    position, accuracy, freshness, trustworthiness
}
operator computeGprsLocation = {
    output : location,
    classname : evry2mallqoc.ComputeGprsPosition,
    input : gprs
}
node gprsLocation = {
    operator : computeGprsLocation,
    children : gprsExtracter
}
node stabilizedLocation = {
    operator : computeStabilization,
    children : gprsLocation gpsLocation wlanLocation
}
operator computeNonAdjustedLocation = {
    output : location,
    classname : evry2mallqoc.ComputeNonAdjustedLocation,
    input : location
}
node nonAdjustedLocation = {
    operator : computeNonAdjustedLocation,
    children : stabilizedLocation
}
operator computeAdjustedLocation = {
    output : location,
    classname : evry2mallqoc.ComputeAdjustedLocation,
    input : location speed
}
node adjustedLocation = {
    operator : computeAdjustedLocation,
    children : nonAdjustedLocation speedVector
}
configuration : gprsExtracter
    [observethrough=false] [periodobserve=10000]
configuration : stabilizedLocation
    [observethrough=false] [periodobserve=2000]
configuration : nonAdjustedLocation
    [observethrough=false] [periodobserve=3000]
configuration : adjustedLocation
    [observethrough=false] [periodobserve=5000]
```

FIGURE 4.6 – Politique de calcul de « Adjusted location » en COSMOS DSL

Troisième partie

Validation

Chapitre 5

Validation

Sommaire

Nous décrivons dans ce chapitre les travaux réalisés dans le cadre de la thèse en termes de validation. En collaboration avec les membres de l'équipe MARGE participant au projet FUI CAPPUCINO [14], nous avons participé au développement de deux prototypes : une application de détection de localisation en situation de mobilité et une application de vente Flash dans un centre commercial. Le premier prototype est introduit à la section 5.1 et est présenté plus en détail dans la publication[Chabridon et al., 2011b]. Cette application de détection de localisation a ensuite permis le développement d'une application de vente Flash, qui est décrite dans la section 5.2 et dans les publications [Chabridon et al., 2012a] et [Chabridon et al., 2012b].

Le projet FUI CAPPUCINO (Construction et Adaptation d'aPPlications Ubiquitaires et de Composants d'INtergiciels en environnement Ouvert pour l'industrie du commerce), qui s'est déroulé de 2007 à 2010, était un projet du Pôle de Compétitivité *Industrie du Commerce* de la région Nord-Pas-de-Calais. Il avait pour but de fournir des outils de niveau intergiciel pour la conception, le déploiement et l'exécution d'applications dans un environnement nomade et ubiquitaire ouvert prenant en compte le contexte de l'utilisateur. Les deux principaux participants industriels étaient le groupe Auchan et le GIE SI3SI du groupe 3 Suisses International, ce qui a permis de concevoir des scénarios applicatifs réalistes avec des utilisateurs finaux du grand public.

14. `http://www.cappucino.fr`

5.1 Application de détection de localisation

Déployée sur téléphone mobile, cette application permet de collecter les différentes informations de localisation pouvant être obtenues de plusieurs sources telles que GPS, WiFi, GSM, etc. Dans une première étape, une opération d'étalonnage permet d'enregistrer la signature de localisations significatives avec les caractéristiques des différents signaux reçus à cet endroit. Chaque localisation présentant un intérêt pour l'utilisateur est alors associée à un nom symbolique. La deuxième étape consiste à reconnaître si l'utilisateur se situe dans une zone significative et laquelle. Le composant Location Aggregator permet d'agréger les différentes informations de localisation disponibles en tenant compte de leur QoC respective afin de déterminer avec la meilleure exactitude possible la localisation réelle de l'utilisateur.

Un ensemble de services est fourni par l'application tels que la liste des localisations successives de l'utilisateur afin d'établir son itinéraire, et la délimitation de la zone géographique correspondant au meilleur niveau de qualité. Les composants de la politique de contexte permettant de mettre en œuvre ces services sont décrits à la figure 5.1. La figure 5.2 montre une vue des différentes étapes d'utilisation de l'application. Nos expérimentations ont permis de montrer que la prise en compte de la QoC par le Location Aggregator permet d'améliorer la probabilité de reconnaître la bonne zone géographique dans plus de 70% des cas.

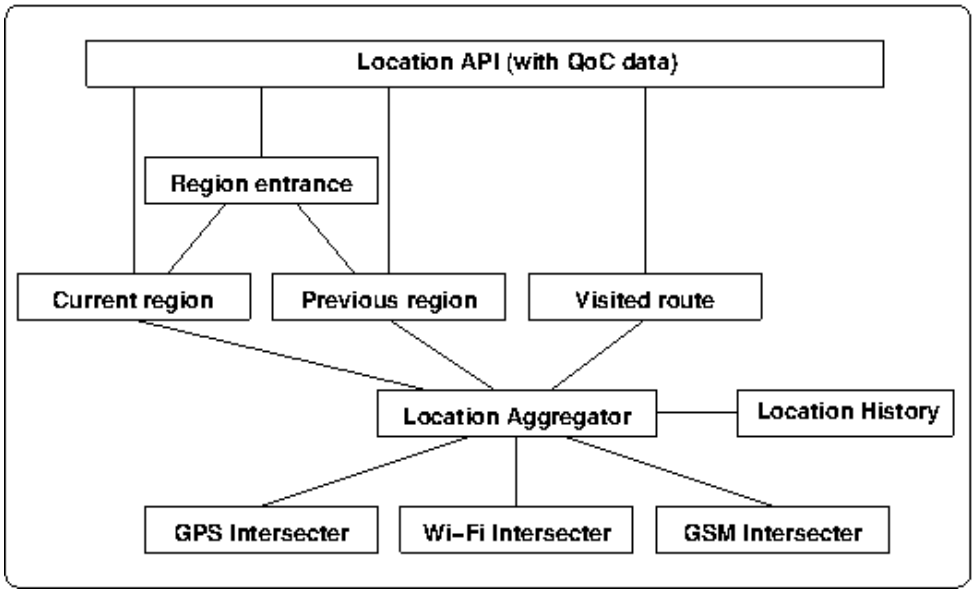


Figure 5.1 – Politique de contexte de localisation

La conception de cette proposition est présentée dans la publication [Chabridon et al., 2011b] qui est jointe à la fin de ce manuscrit. Nous invitons le lecteur à s'y reporter pour plus de détails.

a) Location survey b) Register a location c) Location detection

FIGURE 5.2 – Application de détection de localisation en action

5.2 Scénario de Vente Flash dans un centre commercial

Nous présentons dans cette section un scénario permettant de mettre en évidence l'intérêt de l'utilisation de méta-données de qualité pour les informations de contexte. Nous considérons le cas de la mise en place de ventes Flash dans une zone commerciale. Il s'agit d'envoyer des messages attractifs aux utilisateurs équipés de mobiles bénéficiant de la technologie que nous proposons pour les informer d'offres en cours sur des produits pouvant les intéresser dans un magasin situé à proximité de leur localisation actuelle. La rédaction de ces messages tient compte des éléments de contexte disponibles (profil utilisateur incluant les préférences produits, les règles de respect de la vie privée demandées) et de la qualité de ce contexte.

Ce scénario exploite l'une des spécificités d'une vente Flash qui est d'être valable pour une période de temps limitée et en un lieu précis. Nous montrons ainsi les capacités de notre plateforme en termes de performances, temps de réponse et prise en compte de la qualité des informations de contexte.

À l'heure actuelle, de nombreuses techniques de localisation existent mais possèdent des contraintes très variées en termes d'infrastructure nécessaire, coût, exactitude, mais surtout en termes de disponibilité en intérieur et en extérieur. Ce dernier critère est le plus discriminant car des techniques très efficaces en extérieur telles que les systèmes de navigation par satellite GNSS (Global Navigation Satellite Systems), dont l'un des représentants les plus populaires est le GPS (Global Position System), ne sont pas disponibles en intérieur. De nombreux travaux de recherche existent cependant et des expérimentations sont menées pour améliorer le positionnement en intérieur. Nous donnons dans la table 5.1 un récapitulatif des caractéristiques des techniques actuelles de localisation et dans la table 5.2 une brève classification des techniques de localisation. Pour chaque technique, la première ligne correspond à la localisation en intérieur et la deuxième à

Techniques	**Indoors**	**Outdoors**
Network of sensors	1 cm to 5 m	Not Suitable
RFID	< 1 m	< 1 m
WLAN	few m	Not Suitable
UWB	~ 10 cm	Not Suitable
Cell-Id	500 m to 10 km	100 m to 10 km
E-OTD (2G) / TDOA (3G)	$\gg 200$ m	< 100 m
GNSS	Not Available	few m
A-GNSS	10 m to Not Available	few m
Pseudolites	~ 10 cm	few m
Repeaters	~ 1 to 2 m	few m
Inertial	< 1 m (time dependent)	< 1 m (time dependent)
...	...	...

TABLE 5.1 – Caractéristiques des techniques de localisation — cf. table 9.4 de [Samama, 2008]

la localisation en extérieur. Les colonnes s'interprètent de la manière suivante. Le positionnement peut être de type R (relatif à une origine) ou A (absolu). *Acc* (*Accuracy*) correspond à l'exactitude de la localisation donnée, *Avail* (*Availability*) indique la disponibilité à la fois d'un signal et du positionnement, *Infra* indique la complexité de l'infrastructure requise et *Term* indique la complexité impliquée au niveau du terminal mobile. Les + et − qualifient l'intérêt relatif ou la difficulté, respectivement.

Le scénario que nous proposons se décline en deux versions, l'une pour l'extérieur et l'autre pour l'intérieur. Nous supposons dans chaque cas que la localisation de l'utilisateur peut être obtenue via différentes sources. Toutes les sources possibles ne sont pas forcément disponibles mais nous prévoyons leur intégration au sein de la plate-forme COSMOS.

Nous présentons dans les sections suivantes les deux versions du scénario.

5.2.1 Scénario en intérieur

La zone commerciale considérée dans ce scénario correspond à une galerie commerciale, représentant typiquement un environnement clos en intérieur potentiellement sur plusieurs étages comprenant un ensemble de commerces.

Nous supposons que lors de son entrée dans la galerie commerciale, l'utilisateur se déclare de manière volontaire auprès de la plate-forme COSMOS en se présentant à une borne d'accueil interactive. Une liste de services lui est alors proposée, parmi lesquels un service de localisation. Si l'utilisateur accepte d'être localisé via la plate-forme COSMOS, sa localisation pourra être récupérée via l'infrastructure de communication déployée dans la galerie marchande.

Le niveau de confiance attaché à l'information de localisation varie selon que cette localisation est fournie par l'infrastructure COSMOS ou bien par le terminal. Dans ce dernier cas, nous considérons que des mécanismes de masquage ou encore de génération de chemins fictifs

	Pos	Acc	Avail	Infra	Term
Sensor networks	R	+ + +	++	− − −	+
WLAN	A	+	++	−−	+
Telecommunication networks	A	−−	+	++	+
	A	−	+	++	+
GNSS	A	++	+	++	++
Inertial systems	R	+	++	+ + +	−
	R	+	+ + +	+ + +	−

TABLE 5.2 – Brève classification des techniques de localisation — cf. table 9.5 de [Samama, 2008]

[Krumm, 2009] peuvent être activés par l'utilisateur sur le terminal pour modifier l'information de localisation transmise et que la confiance que nous pouvons avoir dans cette information doit être diminuée.

Les informations de contexte manipulées dans ce scénario sont les suivantes :
- profil utilisateur : préférences parmi les produits du catalogue de vente en ligne, règles de QoC, exigences pour la protection de la vie privée,
- localisation : résultat d'une combinaison d'informations issues de plusieurs sources (GPS, WIFI, Bluetooth...),
- date courante,
- vitesse de déplacement.

Des méta-données de qualité sont attachées à la localisation et sont exploitées dans ce scénario :
- exactitude («positional accuracy»),
- précision (résolution),
- fraîcheur,
- confiance.

Les méta-données de qualité suivantes sont attachées à la vitesse de déplacement de l'utilisateur :
- précision (résolution),
- fraîcheur.

Nous distinguons différents cas selon les critères de fraîcheur, exactitude et confiance comme indiqué dans la table 5.3. Une valeur suffisante est indiquée par le symbole « + », tandis qu'une valeur insuffisante est notée « − ». Nous considérons dans la suite uniquement les quatre cas

Fraîcheur	Exactitude	Confiance	Cas
+	+	+	1
+	+	−	2
+	−	+	
+	−	−	
−	+	+	3
−	+	−	4
−	−	+	
−	−	−	

TABLE 5.3 – Différents cas selon la qualité de la localisation

pour lesquels l'exactitude de la localisation est suffisante. Selon le cas, un message est envoyé à l'utilisateur lorsqu'il entre dans une zone proche du lieu de la vente flash. Un tel message est envoyé une seule fois et une confirmation est demandée à l'utilisateur pour connaître l'adéquation entre l'offre et les préférences de l'utilisateur.

- Cas 1
  Si la fraîcheur et la confiance de la localisation sont suffisantes et si l'utilisateur est à proximité du lieu de la vente flash alors un message d'annonce très détaillé est envoyé indiquant le chemin pour se rendre sur le lieu de la vente flash : « Une vente flash vous attend au magasin XX pendant encore YY minutes sur le produit PP. Pour vous rendre à ce magasin à partir de l'endroit où vous êtes, il vous suffit de ... ».
- Cas 2
  Si la localisation indique que l'utilisateur est à proximité du lieu de la vente flash avec une fraîcheur et une exactitude suffisantes, mais la confiance est insuffisante alors un message est envoyé à l'utilisateur en indiquant une distance approximative à parcourir et non un itinéraire précis : « Une vente flash vous attend au magasin XX pendant encore YY minutes sur le produit PP Vous n'êtes qu'à MM mètres de ce magasin. »
- Cas 3
  Si la localisation indique que l'utilisateur est à proximité du lieu de la vente flash avec une exactitude et une confiance suffisantes, mais la fraîcheur est insuffisante, alors il s'agit d'utiliser la vitesse de déplacement pour calculer une prédiction de la localisation actuelle. Cette localisation ajustée recevra un niveau de confiance diminué en raison de l'introduction d'une prédiction et non d'une mesure réelle.
  Si la localisation ajustée est proche du lieu de la vente flash, alors un message sans indication détaillée sur l'itinéraire à suivre est envoyé : « Une vente flash vous attend au magasin XX pendant encore YY minutes sur le produit PP. »
- Cas 4
  La méthode de prédiction utilisée pour calculer une nouvelle localisation plus fraîche est ici plus « pessimiste » que dans le cas 3 pour limiter les risques d'une divergence trop

grande avec la trajectoire réelle.
Si la localisation ajustée est proche du lieu de la vente flash, alors est envoyé un message sans indication détaillée sur l'itinéraire à suivre comme dans le cas 3 : « Une vente flash vous attend au magasin XX pendant encore YY minutes sur le produit PP. »

Si l'utilisateur a signalé qu'il trouvait l'offre intéressante, mais la valeur suivante de la localisation indique qu'il ne s'y est pas encore rendu, un nouveau message peut alors être envoyé avec le décompte du temps restant : « Plus que YY minutes pour profiter de la vente flash. »

La figure 5.3 montre la politique de contexte COSMOS à mettre en œuvre pour réaliser le scénario proposé. La décision de transmission est prise en tenant compte du profil utilisateur, de la qualité de l'information de localisation et du contrat de QoC.

5.2.2 Scénario en extérieur

Dans ce scénario, l'utilisateur se déplace dans des rues commerçantes potentiellement piétonnes et comprenant une concentration importante de commerces. La localisation de l'utilisateur peut donc être connue par des moyens grand public tant qu'il est à l'extérieur des magasins. Plusieurs techniques de localisation peuvent être utilisées qui fournissent une qualité différente (cf. Table 5.2). Lorsque l'information de localisation est déterminée au niveau du terminal (réseaux de communication mobile ou GNSS), cela a un impact sur la confiance que la plate-forme COSMOS peut accorder à cette information.

Les sources d'information permettant de calculer la localisation dans ce scénario sont différentes de ce qu'il est possible de faire en intérieur, mais les différents cas considérés sont ensuite équivalents aux scénarios présentés précédemment (cf. Table 5.3). Les moyens d'obtenir les informations et la qualité attendue sont différents, mais le comportement de l'application est le même du point de vue de l'utilisateur.

5.2.3 Évaluation

Nous présentons dans cette section le résultat des expérimentations que nous avons menées pour évaluer les performances de l'application de vente Flash et mesurer le coût de la gestion de la qualité de contexte avec un téléphone Samsung Galaxy S2, avec le système d'exploitation Android 2.3.5, un processeur à deux cœurs ARM Cortex-A9 cadencés à $1.2GHz$, avec $16Go$ de ROM et $1Go$ de RAM ($660Mo$ initiallement disponibles), exécutant une machine virtuelle Java 1.5.

La table 5.4 indique la durée nécessaire pour créer l'ensemble de la politique de contexte de l'application de vente Flash. Avec ou sans gestion de la QoC, l'ensemble des nœuds de contexte nécessaires pour la politique de contexte de la vente Flash est créé en moins de $100ms$. Ce temps reste très raisonnable et n'impacte pas de manière significative le temps de déploiement sur un téléphone mobile.

La table 5.5 donne la durée d'une observation de la politique de contexte avec et sans gestion de la QoC dans le pire cas, c'est-à-dire lorsque tous les nœuds sont non bloquants (l'appel

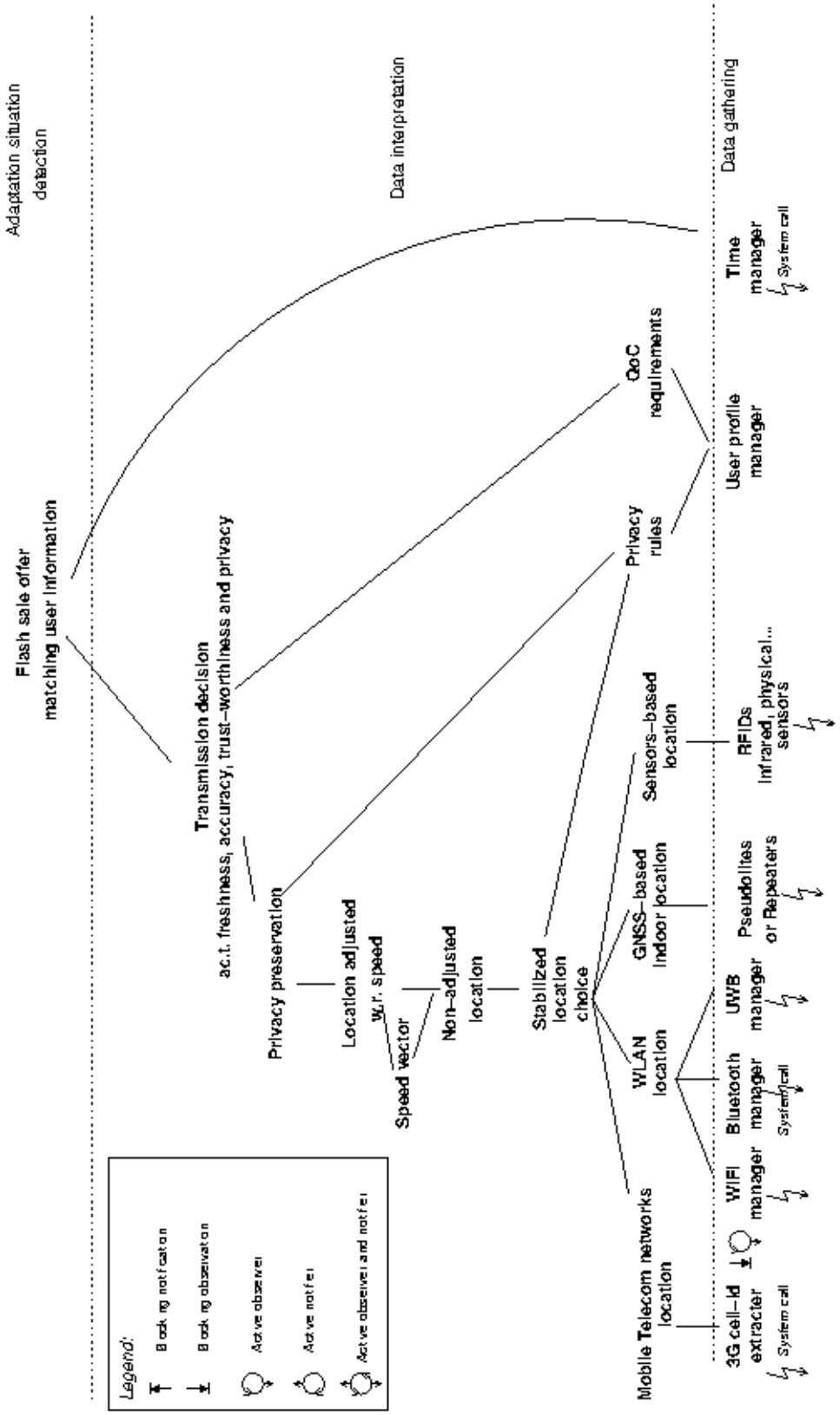


FIGURE 5.3 – Politique de contexte COSMOS pour l'application de Vente Flash

Cas de test	Moyenne (ms)	Intervalle de confiance à 95%
Sans QoC	69.53	[67.77..71.31]
Avec QoC	99.28	[97.04..101.53]

TABLE 5.4 – Durée de l'instantiation

parcourt tous les nœuds du graphe jusqu'aux nœuds feuilles). Le point à noter est que le surcoût lié à la gestion de la QoC est seulement d'environ $1ms$.

Cas de test	Moyenne (ms)	Intervalle de confiance à 95%
Sans QoC	15.2	[14.87..15.52]
Avec QoC	16.27	[15.54..17.00]

TABLE 5.5 – Durée d'une observation (cas défavorable)

Les résultats de l'évaluation de performance présentés dans cette section montrent que COSMOS permet d'effectuer la gestion de contexte entièrement sur un téléphone mobile. Le calcul des méta-données de QoC engendre un coût supplémentaire, expliqué par la nécessité d'ajouter des composants dans l'architecture mise en place, mais celui-ci reste faible. D'autres tests montrent que notre approche peut passer à l'échelle lorsqu'un grand grand nombre de nœuds de contexte est nécessaire. Par exemple, en moins de $100ms$, il est possible de parcourir 1 chemin de 121 nœuds ou 20 chemins de 5 nœuds. Il est aussi possible de créer près de 3000 nœuds sensibles au contexte avant de saturer la mémoire vive.

5.3 Conclusion

Dans ce chapitre, nous avons présenté les prototypes de deux applications pour téléphone mobile qui nous ont permis de valider nos travaux et de montrer la valeur ajoutée apportée par le fait de prendre en compte la qualité des informations de contexte.

Dans l'application de détection de localisation, la QoC permet d'améliorer la pertinence des résultats pour déterminer de manière la plus exacte possible la zone réelle où se situe l'usager.

Dans le cas de l'application de vente Flash, un meilleur service est rendu à l'usager en adaptant le message d'information qui lui est transmis en fonction de la QoC de l'information de localisation. Ceci facilitera l'adoption des applications sensibles au contexte par les usagers du grand public.

Nous avons par la suite présenté les résultats d'évaluation des performances de l'application de vente Flash sur un téléphone mobile sur Android afin de mesurer le surcoût engendré par le calcul de la QoC. Les résultats de l'évaluation montrent que pour 95% des cas, le chargement de la politique de contexte prend moins $100ms$ ce qui est transparent pour l'utilisateur dès la première utilisation. De plus, à chaque action impliquant un changement de contexte et donc la traversée des nœuds de la politique de contexte, l'ajout de la qualité n'entraîne pas plus de $2ms$

de surcoût pour un total de 17ms sur les temps de réponse de l'application dans 95% de cas .

Chapitre 6

Conclusion et Perspectives

6.1 Synthèse des contributions

L'informatique sensible au contexte vise à réduire la quantité d'informations explicites qu'un utilisateur doit fournir pour que le système accomplisse la tâche souhaitée. Cela passe par la perception du contexte de l'utilisateur pour connaître sa situation actuelle et mieux déterminer comment adapter le système ambiant pour qu'il ait le comportement adéquat satisfaisant les attentes de l'utilisateur. Le contexte revêt une importance fondamentale en permettant d'enrichir les activités humaines [Coutaz et al., 2005].

Comme nous l'avons montré dans le chapitre 2 de cette thèse, encore peu de travaux de recherche prennent en compte la qualité de contexte dans le cadre de l'intelligence ambiante. Cette thèse est ainsi une avancée pour le développement de solutions intergicielles flexibles et performantes pour les applications sensibles au contexte avec gestion de la QoC. La première contribution de cette thèse est d'intégrer la gestion de la qualité des informations de contexte au sein du gestionnaire de contexte COSMOS, développé par l'équipe MARGE de Télécom SudParis. L'intégration s'effectue par l'introduction d'un nouveau type de nœud de contexte permettant à la fois de préserver la gestion standard du contexte tout en préparant les éléments nécessaires à la construction, l'analyse et la gestion du contexte. Cette contribution inclut également une méthode pour la gestion fine de la qualité de contexte en intégrant différentes dimensions de la qualité pour construire une qualité résultante pouvant être une référence dans la comparaison des informations de contexte et ainsi la sélection du meilleur contexte.

Notre deuxième contribution consiste à prendre en compte la qualité de contexte dans le processus de modélisation pour permettre la génération des applications sensibles au contexte à l'aide du module cosmos2code. Ainsi, l'architecte se focalise essentiellement sur la partie spécifique à l'application et doit simplement fournir une description de la politique de contexte choisie pour permettre de générer le code d'implémentation.

6.2 Perspectives

6.2.1 Cohérence des informations de contexte

La gestion de contexte en environnement pervasif devient un élément clé pour le développement de nouvelles applications. Cependant, les propriétés des systèmes pervasifs ont un impact sur la cohérence des informations de contexte. La composition d'observations réparties provenant de différentes sources peut introduire des incohérences potentielles. Il est ainsi d'autant plus nécessaire de qualifier les informations de contexte et de déterminer des critères de comparaison. Des sources de contexte pourront également apparaître plus fiables que d'autres et des mécanismes permettant de sélectionner les sources de contexte à privilégier doivent être mis en place.

De plus, le contexte est un élément dynamique évoluant fréquemment et affectant en continu les applications sensibles au contexte. Les méthodes traditionnelles de gestion de cohérence ne sont donc pas toujours adaptées pour un système pervasif. Une perspective du travail présenté dans cette thèse serait de combiner deux approches complémentaires. D'une part, la prise en compte de la qualité des informations de contexte au sein de notre intergiciel de gestion de contexte COSMOS permettrait de traiter des conflits syntaxiques au sein de contextes de bas niveau. D'autre part, l'utilisation d'ontologies permettrait de déterminer l'équivalence sémantique de deux contextes pour lever des conflits de plus haut niveau.

6.2.2 Respect de la vie privée

La figure 1.1 du chapitre 1 montre l'évolution des technologies depuis l'informatique mobile vers l'Internet des objets qui fait suite au paradigme d'intelligence ambiante. L'Internet des objets donne une nouvelle dimension à l'intelligence ambiante en ne se limitant plus à un environnement proche d'un utilisateur particulier et pose des défis importants pour les services de gestion de contexte. Dans l'Internet des objets, le nombre d'objets connectés croît de manière exponentielle. Ces objets sont des sources (ou producteurs) de contexte qui fournissent des informations à transformer et acheminer vers un nombre important et changeant de consommateurs. L'acheminement et le traitement des informations sont réalisés par plusieurs intermédiaires situés à différentes échelles. Ainsi, avec l'Internet des objets, la gestion de contexte intervient à différentes échelles (réseau personnel, Internet, nuages) avec un découplage fort entre producteurs et consommateurs.

Comme le souligne l'UIT dans son rapport sur l'Internet des Objets [UIT, 2011] : « Des données seront échangées en permanence, de manière invisible à nos yeux, et sur une très grande échelle entre des objets et des personnes, ainsi qu'entre les objets eux-mêmes, à l'insu des « propriétaires » de ces données. Qui, en dernier ressort, sera maître des données collectées par tous les yeux et toutes les oreilles électroniques dans notre entourage ? » Afin de favoriser l'acceptabilité par les utilisateurs de nouvelles applications ambiantes, il est fondamental d'offrir des mécanismes permettant de garantir le respect de la vie privée des utilisateurs et la protection des données manipulées.

L'une des difficultés est de traiter les potentiels antagonismes entre, d'une part, les contraintes

impliquées par la demande d'un niveau de QoC satisfaisant pour les informations de contexte et, d'autre part, la nécessité de garantir la protection des données fournies par les utilisateurs : par exemple, le producteur est contraint par le respect de la vie privée qui implique de brouiller l'information de localisation, alors même que le consommateur demande un niveau de précision de localisation en contradiction avec ce qui peut être fourni par le producteur. Nous pensons qu'un traitement conjoint QoC et vie privée permettrait de résoudre ce type de conflit.

Bibliographie

[Abid and Chabridon, 2011] Abid, Z. and Chabridon, S. (2011). A Fine-grain Approach for Evaluating the Quality of Context. In *8th IEEE Workshop on Context Modeling and Reasoning (CoMoRea), PerCom'11 Workshop Proceedings*, Seattle, WA, USA. IEEE Computer Society. 5, 64, 82

[Abid et al., 2009a] Abid, Z., Chabridon, S., and Conan, D. (2009a). Cohérence et qualité des informations de contexte en environnement pervasif. In *3ème Workshop sur la Cohérence des Données en Univers Réparti - CDUR'09*, Toulouse, France. 6

[Abid et al., 2009b] Abid, Z., Chabridon, S., and Conan, D. (2009b). A framework for quality of context management. In *First Int. Workshop on Quality of Context (QuaCon), Stuttgart, Germany*, volume 5786 of *LNCS*. Springer. 6, 64, 73

[Abowd et al., 1998] Abowd, D., Dey, A., Orr, R., and Brotherton, J. (1998). Context-awareness in wearable and ubiquitous computing. *Virtual Reality*, 3(3) :200–211. 26

[Ashley et al., 1999] Ashley, J., Copeland, M., Grahn, J., and Wheeler, D. (1999). The GNU Privacy Handbook. *The Free Software Foundation*. 34

[Atrey et al., 2010] Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis : a survey. *Multimedia System, Springer Berlin-Heidelberg*, (April). 39

[Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A Survey on Context Aware Systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277. 24

[Baldwin, 1987] Baldwin, J. (1987). Evidential support logic programming. *Fuzzy sets and systems*, 24(1) :1–26. 42

[Bayes, 1763] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Soceity of London*, 53 :370–418. 37

[Beamon and Kumar, 2010] Beamon, B. and Kumar, M. (2010). Adaptive context reasoning in pervasive systems. In *Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware*, pages 10–17. ACM. 40

[Becker et al., 2010] Becker, S., Blessing, A., Dürr, F., Geiger, L., Großmann, M., Gutscher, A., Häussermann, K., Heesen, J., Käppeler, U., Lange, R., et al. (2010). Reference Model for the Quality of Context Information. Nexus Team. 13, 24, 35, 36, 41

[Bettini et al., 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing, Elsevier*, 6(2) :161 – 180. 46

[Bouzeghoub et al., 2010] Bouzeghoub, A., Taconet, C., Jarraya, A., Do, N. K., and Conan, D. (2010). Complementarity of Process-oriented and Ontology-based Context Managers to Identify Situations. In *4th IEEE Int. Workshop on Context Modeling and Management for Smart Environments in conjunction with ICDIM '10 : 5th Int. Conf. on Digital Information Management, Thunder Bay, Canada.* 55

[Brézillon, 2002] Brézillon, P. (2002). Hors du contexte point de salut. In *Objets communicants*, Autrans, France. 26

[Bringel Filho and Martin, 2008] Bringel Filho, J. and Martin, H. (2008). A quality-aware context-based access control model for ubiquitous applications. In *Enhancing interoperability between enterprise planning applications : an architectural framework.* 53

[Bringel Filho et al., 2010] Bringel Filho, J., Miron, A., Satoh, I., Gensel, J., and Martin, H. (2010). Modeling and Measuring Quality of Context Information in Pervasive Environments. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 690–697. IEEE. 11, 13, 53, 54, 57, 58, 59

[Brown, 1995] Brown, P. (1995). The stick-e document : a framework for creating context-aware applications. *ELECTRONIC PUBLISHING-CHICHESTER-*, 8 :259–272. 26, 123

[Buchholz et al., 2003] Buchholz, T., Kupper, A., and Schiffers, M. (2003). Quality of context information : What it is and why we need it. In *10th Int. Workshop of the HP OpenView University Association (HPOVUA)*, Geneva, Switzerland. 27, 30, 58, 90

[Budinsky et al., 2008] Budinsky, F., Merks, E., and Steinberg, D. (2008). *Eclipse Modeling Framework 2.0.* Eclipse. Addison Wesley Professional. 46

[Chaari et al., 2007] Chaari, T., Ejigu, D., Laforest, F., and Scuturici, V. (2007). A comprehensive approach to model and use context for adapting applications in pervasive environments. *Journal of Systems and Software*, 80(12) :1973–1992. 38

[Chabridon et al., 2011a] Chabridon, S., Abid, Z., Taconet, C., and Conan, D. (Dec. 2011a). A Model Driven Approach for the QoC-Awareness of Ubiquitous Applications. In *UCAMI'11 : Proceedings of the 5th International Symposium on Ubiquitous Computing and Ambient Intelligence, Riviera Maya, Cancun, México.* AIAm. 5, 85

[Chabridon et al., 2012b] Chabridon, S., Conan, D., Abid, Z., and Taconet, C. (2012b). Ingénierie dirigée par les modèles pour la construction d'applications ubiquitaires sensibles à la qualité du contexte. pages 167–176. 5, 85, 99

[Chabridon et al., 2012a] Chabridon, S., Conan, D., Abid, Z., and Taconet, C. (Online publication in September 2012a). Building ubiquitous QoC-aware Applications through Model-driven Software Engineering. *Science of Computer Programming.* 5, 85, 99

[Chabridon et al., 2010] Chabridon, S., Conan, D., Taconet, C., Nguyen, C., Ngo, C., Lim, L., and Abid, Z. (2010). MDE, DSL and Tooling for Effective Context Management in Ubiquitous

Computing. In *Proc. MobiCASE Workshop on Mobile Software Engineering*, Santa Clara, CA, USA. 6

[Chabridon et al., 2011b] Chabridon, S., Ngo, C.-C., Abid, Z., Conan, D., Taconet, C., and Ozanne, A. (2011b). Towards qoc-aware location-based services. In *DAIS'11 : Proceedings of the 11th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, pages 71–76, Berlin, Heidelberg. Springer-Verlag. 5, 99, 100

[Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College, Hanover, NH, (USA). 46

[Chen et al., 2003] Chen, H., Finin, T., and Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(03) :197–207. 55

[Conan et al., 2007a] Conan, D., Rouvoy, R., and Seinturier, L. (2007a). Scalable Processing of Context Information with COSMOS. In *Proc. 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, volume 4531 of *Lecture Notes in Computer Science*, pages 210–224, Cyprus. Springer-Verlag. 18, 56, 64, 88

[Conan et al., 2007b] Conan, D., Rouvoy, R., and Seinturier, L. (2007b). Scalable Processing of Context Information with COSMOS. In *Proc. 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, volume 4531 of *Lecture Notes in Computer Science*, pages 210–224, Cyprus. Springer-Verlag. 38, 64

[Conan et al., 2008] Conan, D., Rouvoy, R., and Seinturier, L. (2008). COSMOS : composition de nœuds de contexte. *Technique et Science Informatiques*. À paraître. 64

[Conti et al., 2012] Conti, M., Das, S., Bisdikian, C., Kumar, M., Ni, L., Passarella, A., Roussos, G., Tröster, G., Tsudik, G., and Zambonelli, F. (2012). Looking ahead in pervasive computing : Challenges and opportunities in the era of cyber-physical convergence. *Pervasive and Mobile Computing*, 8(1) :2–21. 24, 56

[Coutaz and Crowley, 2008] Coutaz, J. and Crowley, J. (2008). Plan « intelligence ambiante » : défis et opportunités. Rapport final du Groupe de Travail « Intelligence Ambiante » du Groupe de Concertation Sectoriel (GCS3) MSER DGRI A3. 25

[Coutaz et al., 2005] Coutaz, J., Crowley, J., Dobson, S., and Garlan, D. (2005). Context is Key. *Communications of the ACM*, 48(3) :49–53. 16, 65, 109

[Dargie and Hamann, 2006a] Dargie, W. and Hamann, T. (2006a). A Distributed Architecture for Reasoning about a Higher-Level Context. In *Proc. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 268–275. 24

[Dargie and Hamann, 2006b] Dargie, W. and Hamann, T. (2006b). A distributed architecture for reasoning about a higher-level context. In *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006)*, pages 268– 275. 46

[Dey, 2001] Dey, A. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1) :4–7. 25

[Dey and Abowd, 2000] Dey, A. and Abowd, G. (2000). Towards a better understanding of context and context-awareness. In *Chi 2000 workshop on the what, who, where, when, and how of context-awareness, Conference on Human Factors in Computer Systems*, pages 304–307. 38

[Dey et al., 2001a] Dey, A., Abowd, G., and Salber, D. (2001a). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2) :97–166. 50

[Dey et al., 2001b] Dey, A., Salber, D., and Abowd, G. (2001b). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Special issue on context-aware computing in the Human-Computer Interaction Journal*, 16(2–4) :97–166. 46, 65

[Dorn et al., 2006] Dorn, C., Schall, D., and Dustdar, S. (2006). Granular context in collaborative mobile environments. In *On the Move to Meaningful Internet Systems 2006 : OTM 2006 Workshops*, pages 1904–1913. Springer. 59

[Filho, 2010] Filho, J. B. (2010). *A Family of Context-Based Access Control Models for Pervasive Environments*. PhD thesis, MSTII Doctoral School, University of Grenoble Joseph Fourier, France. 11, 53, 54, 58

[Frank et al., 2010] Frank, K., Robertson, P., and Fortes Rodriguez (2010). Faster Bayesian context inference by using dynamic value ranges. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 50 – 55. 37

[Frederik Hermans, 2009] Frederik Hermans (2009). *Data Fusion Based on Distributed Quality Estimation in Wireless Sensor Networks*. PhD thesis, Fachbereich Mathematik und Informatik, Institut für Informatik, Berlin. Advisors : Prof. Dr.-Ing. Jochen Schiller Dipl.-Inform. Norman Dziengel. 13, 43

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns : elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 78

[Gutscher, 2007] Gutscher, A. (2007). A trust model for an open, decentralized reputation system. *Trust Management*, pages 285–300. 41

[Haghighi et al., 2008] Haghighi, P. D., Krishnaswamy, S., Zaslavsky, A., and Gaber, M. M. (2008). Reasoning about context in uncertain pervasive computing environments. In *3rd IEEE European Conf. on Smart Sensing and Context (EuroSSC)*, volume 5279 of *Lecture Notes in Computer Science (LNCS)*, pages 112–125. Springer. 38

[Hall and McMullen, 2004] Hall, D. and McMullen, S. (2004). *Mathematical techniques in multisensor data fusion*. Artech House Publishers. 39

[Henricksen, 2003] Henricksen, K. (2003). *A framework for context-aware pervasive computing applications*. PhD thesis, School of Information Technology and Electrical Engineering, The University of Queensland, Australia. 24, 55, 56

[Henricksen and Indulska, 2006] Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications : Models and approach. *Pervasive and Mobile Computing*, 2(1) :37–64. 46, 86

[ISO, 2004] ISO (2004). International vocabulary of basic and general terms in metrology (vim). *International Organization for Standardization*, 2004 :09–14. 27, 32

[Jøsang, 1997] Jøsang, A. (1997). Artificial reasoning with subjective logic. In *Proceedings of the Second Australian Workshop on Commonsense Reasoning*, volume 48. 34, 35, 42, 43

[Khare and Rifkin, 1998] Khare, R. and Rifkin, A. (1998). Trust management on the world wide web. *Computer networks and ISDN Systems*, 30(1) :651–653. 34

[Kim and Lee, 2006a] Kim, Y. and Lee, K. (2006a). A quality measurement method of context information in ubiquitous environments. 2 :576–581. 13, 27, 28, 29, 30, 31, 32, 54, 58

[Kim and Lee, 2006b] Kim, Y. and Lee, K. (2006b). A quality measurement method of context information in ubiquitous environments. *International Conference on Hybrid Information Technology*, 2 :576–581. 24

[Klyne and al., 2007] Klyne, G. and al. (2007). Composite Capability/Preference Profile (CC/PP) : Structure and vocabularies 2.0. W3C recommendation. 46, 86

[Korpipaa et al., 2003] Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., and Malm, E. (2003). Managing context information in mobile devices. *IEEE pervasive computing*, 2(3) :42–51. 38

[Krause and Hochstatter, 2005] Krause, M. and Hochstatter, I. (2005). Challenges in Modelling and Using Quality of Context (QoC). In et al., T. M., editor, *Mobility Aware Technologies and Applications (MATA)*, volume 3744 of *Lecture Notes in Computer Science*, pages 324–333. Springer-Verlag. 27, 58

[Kristian Ellebæk Kjær, 2007] Kristian Ellebæk Kjær (2007). A survey of context-aware middleware. In *SE'07 : Proceedings of the 25th conference on IASTED International Multi-Conference*, pages 148–155, Anaheim, CA, USA. ACTA Press. 24

[Krumm, 2009] Krumm, J. (2009). Realistic driving trips for location privacy. In Tokuda, H. et al., editor, *7th International Conference on Pervasive Computing*, volume LNCS 5538 of *Lecture Notes in Computer Science*, pages 25–41. Springer. 103

[Leclercq et al., 2007] Leclercq, M., Ozcan, A. E., Quema, V., and Stefani, J.-B. (2007). Supporting Heterogeneous Architecture Descriptions in an Extensible Toolset. In *ICSE '07 : Proceedings of the 29th Int. Conf. Software Engineering*, pages 209–219, Washington, DC, USA. IEEE Computer Society. 66

[Lieberman and Selker, 2000] Lieberman, H. and Selker, T. (2000). Out of context : Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3.4) :617–632. 26

[Manzoor, 2010] Manzoor, A. (2010). *Quality of Context in Pervasive Systems : Models, Techniques, and Applications*. PhD thesis, School of Computer Science, Wien Technical University, Austria. 24, 28, 58

[Manzoor et al., 2008] Manzoor, A., Truong, H., and Dustdar, S. (2008). On the Evaluation of Quality of Context. In *IEEE EuroSCC, 3d European Conference on Smart Sensing and Context*, volume LNCS 5279. Springer. 11, 13, 17, 28, 30, 31, 32, 47, 48, 53, 54, 58, 59

[Manzoor et al., 2012] Manzoor, A., Truong, H.-L., and S, D. (2012). Quality of Context : Models and Applications for Context-aware Systems in Pervasive Environments. *The Knowledge Engineering Review Special Issue on Web and Mobile Information Services*, To appear. 28

[Marsh and Science, 1994] Marsh, S. and Science, U. (1994). *Formalising trust as a computational concept*. Citeseer. 34

[McKeever, 2011] McKeever, S. (2011). *Recognising Situations Using Extended Dempster-Shafer Theory*. PhD thesis, School of Computer Science and Informatics, National University of Ireland, Dublin. 40, 44

[McKeever et al., 2009a] McKeever, S., Ye, J., Coyle, L., and Dobson, S. (2009a). A Context Quality Model to Support Transparent Reasoning with Uncertain Context. In *First Int. Workshop on Quality of Context, QuaCon 2009, Stuttgart, Germany*, volume 5786 of *LNCS*. Springer. 13, 47, 48

[McKeever et al., 2009b] McKeever, S., Ye, J., Coyle, L., and Dobson, S. (2009b). Using Dempster-Shafer Theory of Evidence for Situation Inference. *Smart Sensing and Context*, pages 149–162. 13, 40, 44, 45, 46, 59

[Nakamura et al., 2007] Nakamura, E. F., Loureiro, A. A. F., and Frery, A. C. (2007). Information fusion for wireless sensor networks : Methods, models, and classifications. *ACM Computing Surveys*, 39(3). 37, 38

[Neisse, 2012] Neisse, R. (2012). *Trust and Privacy Management Support for Context-Aware Service Platforms*. PhD thesis, CTIT School, University of Twente, Enschede, The Netherlands. 48, 58

[Neisse et al., 2008] Neisse, R., Wegdam, M., and van Sinderen, M. (2008). Trustworthiness and quality of context information. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, pages 1925–1931. IEEE. 13, 27, 33, 34, 48, 49, 50, 58, 59

[Nzekwa et al., 2010] Nzekwa, R., Rouvoy, R., and Seinturier, L. (2010). A Flexible Context Stabilization Approach for Self-Adaptive Application. In *7th IEEE Workshop on Context Modeling and Reasoning (CoMoRea), PerCom'10 Workshop Proceedings*. 40, 44

[Object Management Group, 2008] Object Management Group (2008). Software & Systems Process Engineering Metamodel (SPEM) v2.0. formal/2008-04-01. 88

[OMG, 2006] OMG (2006). Meta Object Facility (MOF) Core Specification Version 2.0. OMG document formal/06-01-01. 46

[OMG, 2003] OMG (August 2003). UML 2.0 Superstructure Specification v2.0. OMG document formal/05-07-04. 46

[Pascoe, 1998] Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99. IEEE. 25

[Preuveneers and Berbers, 2006] Preuveneers, D. and Berbers, Y. (2006). Quality Extensions and Uncertainty Handling for Context Ontologies. In Shvaiko, P., Euzenat, J., Léger, A., McGuinness, D. L., and Wache, H., editors, *Proceedings of Context and Ontologies : Theory*

Practice and Applications (C&O 2006), pages 62–64, Riva del Garda, Italy. 13, 50, 51, 52, 53, 59

[Preuveneers and Berbers, 2007a] Preuveneers, D. and Berbers, Y. (2007a). Architectural back-propagation support for managing ambiguous context in smart environments. In *Proc. 4th Universal Access in Human-Computer Interaction, Ambient Interaction, UAHCI 2007, Part of HCI'2007*, volume 4555 of *Lecture Notes in Computer Science (LNCS)*, pages 178–187. Springer-Verlag. 24, 31, 33

[Preuveneers and Berbers, 2007b] Preuveneers, D. and Berbers, Y. (2007b). Towards context-aware and resource-driven self-adaption for mobile handheld applications. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 1165–1170, New York, NY, USA. ACM Press. 55

[Preuveneers and Berbers, 2008] Preuveneers, D. and Berbers, Y. (2008). Mobile phones assisting with health self-care : a diabetes case study. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 177–186. ACM. 38

[Preuveneers et al., 2004] Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., and De Bosschere, K. (2004). Towards an extensible context ontology for ambient intelligence. *Lecture Notes in Computer Science*, pages 148–159. 55

[Ranganathan et al., 2004a] Ranganathan, A., Al-Muhtadi, J., and Campbell, R. (2004a). Reasoning About Uncertain Contexts in Pervasive Computing Environments. *IEEE Pervasive Computing*, 3(2) :10–18. 24, 46

[Ranganathan et al., 2004b] Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R., and Mickunas, M. (2004b). MiddleWhere : A Middleware for Location Awareness in Ubiquitous Computing Applications. In Jacobsen, H.-A., editor, *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 3231 of *Lecture Notes in Computer Science*, pages 397–416, Toronto, Canada. Springer-Verlag. 24

[Razzaque et al., 2005] Razzaque, M., Dobson, S., and Nixon, P. (2005). Categorisation and modelling of quality in context information. In Roy Sterrit, W., Dobson, S., and Smirnov, M., editors, *Proceedings of the International Joint Conference on Artificial Intelligence, Workshop on AI and Autonomic Communications*. 24

[Ricquebourg et al., 2008] Ricquebourg, V., Durand, D., Menga, D., Delahoche, L., Marhic, B., Logé, C., and Jolly-Desodt, A. (2008). La fusion multi-capteurs dans l'habitat communicant : une approche non-probabiliste. In *Proceedings of the 4th French-speaking conference on Mobility and ubiquity computing*, pages 9–16. ACM. 40, 44

[Rouvoy et al., 2008] Rouvoy, R., Conan, D., and Seinturier, L. (2008). Software Architecture Patterns for a Context Processing Middleware Framework. *IEEE Distributed Systems Online*, 9(6). 38, 64

[Roy et al., 2009] Roy, N., Gu, T., and Das, S. (2009). Supporting pervasive computing ap-

plications with active context fusion and semantic context delivery. *Pervasive and Mobile Computing*. 37

[Sadri, 2011] Sadri, F. (2011). Ambient Intelligence : A Survey. *ACM Computing Surveys*, 43(4) :36:1–36:66. 15

[Salber et al., 1998] Salber, D., Dey, A., and Abowd, G. (1998). Ubiquitous computing : Defining an hci research agenda for an emerging interaction paradigm. 26

[Samama, 2008] Samama, N. (2008). *Global Positioning : Technologies and Performance*. Wiley Survival Guides in Engineering and Science. Wiley-VCH. 11, 102, 103

[Satyanarayanan, 2001] Satyanarayanan, M. (2001). Pervasive Computing : Vison and Challenges. *IEEE Personal Communications*, 8(4) :10–17. 25

[Schilit and Theimer, 1994] Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5) :22–32. 25, 26

[Shafer, 1976] Shafer, G. (1976). *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, NJ. 34, 35, 40, 42, 43

[Sheng and Benatallah, 2005] Sheng, Q. and Benatallah, B. (2005). ContextUML : A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services. In *Proc. 4th IEEE ICMB*, pages 206–212, Sydney, Australia. 86

[Taconet et al., 2009] Taconet, C., Kazi-Aoul, Z., Zaier, M., and Conan, D. (2009). CA3M : A Runtime Model and a Middleware for Dynamic Context Management. In Meersman, R., Dillon, T., and Herrero, P., editors, *Proc. 11th International Symposium on Distributed Objects and Applications*, volume 5870 of *LNCS*, pages 513–530, Vilamoura, Algarve, Portugal. Springer-Verlag. 87, 89, 92

[Tang et al., 2007a] Tang, S., Yang, J., and Wu, Z. (2007a). A context quality model for ubiquitous applications. 13, 52, 53, 59

[Tang et al., 2007b] Tang, S., Yang, J., and Wu, Z. (2007b). A context quality model for ubiquitous applications. In *IFIP International Conference on Network and Parallel Computing Workshops*, pages 282–287, Los Alamitos, CA, USA. IEEE Computer Society. 24

[UIT, 2011] UIT (2011). Internet of things. Union Internationale des TÃ©lÃ©communications http ://www.itu.int/. 110

[Uschold and Grüninger, 1996] Uschold, M. and Grüninger, M. (1996). Ontologies : principles, methods, and applications. *Knowledge Engineering Review*, 11(2) :93–155. 49

[Wang et al., 2004] Wang, X. H., Zhang, D. Q., Gu, T., and Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL. In *Proc. 2nd IEEE International Conference on Pervasive Computing and Communications*, pages 18–22, Orlando, FL, USA. 46, 86

[Weiser, 1991] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, pages 94–100. 15

[Wu, 2004] Wu, H. (2004). *Sensor data fusion for context-aware computing using dempster-shafer theory*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. Chair-Siegel, Mel W. 40

[Yager, 1987] Yager, R. (1987). On the dempster-shafer framework and new combination rules 1. *Information sciences*, 41(2) :93–137. 43

[Yasar et al., 2011] Yasar, A.-U.-H., Paridel, K., Preuveneers, D., and Berbers, Y. (2011). When Efficiency Matters : Towards Quality of Context-aware Peers for Adaptive Communication in VANETs. In *Intelligent Vehicles Symposium*, pages 1006–1012, Baden-Baden, Germany. 24

[Yau et al., 2002] Yau, S., Karim, F., Wang, Y., Wang, B., and Gupta, S. (2002). Reconfigurable Context-Sensitive Middleware for Pervasive Computing. *IEEE Pervasive Computing*, 1(3) :33–40. 65

[Ye et al., 2008] Ye, J., McKeever, S., Coyle, L., Neely, S., and Dobson, S. (2008). Resolving Uncertainty in Context Integration and Abstraction. In *ICPS08 : 5th Int. Conf. on Pervasive Services*, pages 131–140, New York, NY, USA. ACM. 47

[Zadeh, 1984] Zadeh, L. (1984). Review of a mathematical theory of evidence. *AI Magazine*, 5(3) :81. 44

[Zimmermann, 2007] Zimmermann, A. (2007). *Context Management and Personalization : A Tool Suite for Context- and User-Aware Computing*. PhD thesis, Westphalie Technical University - Fraunhofer Institute for Applied Information Technology, Germany. 24

Annexe A

Glossaire

Nous résumons dans ce glossaire un certain nombre de termes utilisés dans notre travail.

« Un *Système sensible au contexte* est un système dont le comportement ou la structure peut varier en fonction de l'état de l'espace des informations de contexte ». Le système est un terme général qui désigne une application, un service, l'intergiciel ou le système d'exploitation. Une des premières utilisations de ce terme est [Brown, 1995] qui le définit alors comme un système dont le comportement seul peut varier en fonction du contexte.

« Une *Entité observable* est un élément représentant un phénomène physique ou logique (personne, concept, etc.) qui peut être traité comme une unité indépendante ou un membre d'une catégorie particulière, et auquel des « observables » peuvent être associés. » [15]

« Un *Observable* est une abstraction qui définit un type d'informations à observer. Pour un système donné, un observable est rattaché à une entité observable. » Par exemple, à partir de l'entité observable *terminal*, sont associés les observables suivants : *mémoire vive disponible*, *liste des dispositifs d'interaction avec l'utilisateur*, *liste des connexions réseaux*, *disponibilité d'une connexion réseau*. « Chaque observable est associé à une ou plusieurs *Observation*, chaque observation définit un état de l'observable. » « Un *Observable Composite* est un observable dont les observations sont obtenues par une fonction prenant en entrée les observations d'un ou plusieurs observables. »« Un *Observable immuable* est un observable dont les observations ont la même valeur pendant toute la durée de vie du système. »

« L'*Espace des informations de contexte* d'un système est caractérisé par l'état des différents éléments le caractérisant : les entités observables, les observables et l'état des observations attachées aux observables. »

« Une *Situation d'adaptation* est un observable qui permet de repérer un changement d'état dans l'espace des informations de contexte. Ce changement d'état significatif pour le système nécessite une réaction dans le système. Cette réaction est appelée adaptation. » Une situation d'adaptation est un observable dont les changements d'état génèrent une modification dans le système sensible au contexte.

15. Cette définition est inspirée de la définition des entités du domaine base de données trouvée dans le Grand dictionnaire terminologique (Link :http://www.granddictionnaire.com)

Index

Annexe B

Composants de Gestion de la Qualité de Contexte

B.1 ContextAwareOperator

```
<?xml version="1.0" encoding="ISO-8859-1" ?> <!DOCTYPE definition PUBLIC
"-//objectweb.org//DTD Fractal ADL 2.0//EN"
"classpath://org/objectweb/fractal/adl/xml/standard.dtd">

<!--
COSMOS: COntext entitieS coMpositiOn and Sharing
Copyright: Copyright (C) 2008-2009
Contact: cosmos@picoforge.int-evry.fr

This library is free software; you can redistribute it and/or modify it under he
terms of the GNU Lesser General Public License as published by the FreeSoftware
Foundation; either version 2.1 of the License, or any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU *Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along
with this library; if not, write to the Free Software Foundation, Inc., 59
Temple Place, Suite 330, Boston, MA  02111-1307 USA

*Initial developer(s): Denis Conan.
*Contributor(s): Zied ABID
*
-->

<definition name="cosmos.qoc.QocAwareContextOperator"
        arguments="nodename,activitymanager,periodnotify,periodobserve">

<interface
```

```
name="pull-obs-out" contingency="mandatory" role="server"
cardinality="singleton" signature="cosmos.core.Pull"/>

<interface
name="push-notif-in" contingency="mandatory" role="server"
cardinality="singleton" signature="cosmos.core.Push"/>

<interface
name="push-notif-out" contingency="mandatory" role="client"
cardinality="collection" signature="cosmos.core.Push"/>

<interface
name="pull-obs-in" contingency="mandatory" role="client"
cardinality="collection" signature="cosmos.core.Pull"/>

<interface
name="message-manager" contingency="mandatory" role="client"
cardinality="singleton"
signature="org.objectweb.dream.message.MessageManagerType"/>

<interface
name="task-manager" contingency="mandatory" role="client"
cardinality="singleton"
signature="org.objectweb.dream.control.activity.manager.TaskManager"/>

<component name="message-manager"
        definition="org.objectweb.dream.message.MessageManagerPooling"/>

<component name="activity-manager" definition="${activitymanager}"/>

<component name="parentfacade"
        definition="cosmos.qoc.ParentFacade(nodename=>parent)">
  <attributes>
      <attribute name="periodnotify" value="${periodnotify}"/>
      <attribute name="periodobserve" value="${periodobserve}"/>
  </attributes>
</component>

<component name="qocoperator"
        definition="cosmos.qoc.QocOperator(nodename=>qoc)"/>

<component name="myoperator"
        definition="cosmos.operator.translate.ForwardUnaryCO(nodename=>MyOperator)
            "/>

<component name="childfacade"
        definition="cosmos.qoc.ChildFacade(nodename=>child)"/>

    <binding client="this.pull-obs-out"
             server="parentfacade.pull-obs-out"/>

    <binding server="this.push-notif-out"
```

```
82              client="parentfacade.push-notif-out"/>

84      <binding client="parentfacade.pull-obs-in"
                server="myoperator.pull-obs-out"/>
86
        <binding client="myoperator.push-notif-out"
88              server="parentfacade.push-notif-in" />

90      <binding client="parentfacade.compute-qoc"
                server="qocoperator.compute-qoc"/>
92
        <binding client="parentfacade.message-manager"
94              server="message-manager.message-manager"/>
        <binding
96              client="parentfacade.task-manager"
                server="activity-manager.task-manager"/>
98
        <binding client="myoperator.pull-obs-in"
100             server="childfacade.pull-obs-out"/>

102     <binding client="myoperator.message-manager"
                server="message-manager.message-manager"/>
104
        <binding client="myoperator.task-manager"
106             server="activity-manager.task-manager"/>

108     <binding client="qocoperator.qoc-data"
                server="childfacade.qoc-data"/>
110
        <binding client="qocoperator.message-manager"
112             server="message-manager.message-manager"/>

114     <binding client="childfacade.pull-obs-in"
                server="this.pull-obs-in"/>
116
        <binding client="childfacade.push-notif-out"
118             server="myoperator.push-notif-in"/>

120     <binding client="this.push-notif-in"
                server="childfacade.push-notif-in"/>
122
        <binding client="childfacade.message-manager"
124             server="this.message-manager"/>

126     <binding client="childfacade.task-manager"
                server="this.task-manager"/>
128
        <controller desc="composite"/>
130 </definition>
```

"Exemple complet de construction d'un nœud de qualité de contexte"

B.2 Arborescence d'un projet COSMOS avec QoC

```
.
├── pom.xml
├── README.txt
├── run.bat
├── run.sh
├── src
    ├── main
        ├── config
        │   ├── adlList.txt
        │   ├── logging.properties
        │   ├── spoon.cfg.xml
        ├── java
        │   ├── cosmos
        │       ├── operator
        │       │   ├── util
        │       │       ├── WrapperToSEorME.java
        │       ├── qoc
        │           ├── chunk
        │           │   ├── LocationChunk.java
        │           │   ├── PrecisionChunk.java
        │           │   ├── QoCChunk.java
        │           ├── collector
        │           │   ├── CollectorOfLocation.java
        │           ├── info
        │           │   ├── Locationinfo.java
        │           │   ├── PrecisionInfo.java
        │           │   ├── QoCinfo.java
        │           ├── operator
        │           │   ├── parameters
        │           │   │   ├── Completeness.java
        │           │   │   ├── Precision.java
        │           │   │   ├── UpToDateness.java
        │           │   ├── QoCInterface.java
        │           │   ├── translate
        │           │       ├── ForwardUnaryCOWithQoC.java
        │           │       ├── MergeNAryCOWithQoC.java
        │           ├── simulation
        │               ├── yourpackage
        │                   ├── MainContextNode.java
        │                   ├── Main.java
        │                   ├── MainWithoutCosmos.java
        ├── resources
            ├── julia-dream-activities-bundled.cfg
            ├── julia-dream-bundled-forCosmos.cfg
            ├── monolog-cosmos.properties
            ├── Siafu-1.0.4
            │   ├── Lib
            │   │   ├── linux
            │   │   │   ├── swt
```

```
50 |   |   |   |       +-- about_files
   |   |   |   |       |   +-- about_cairo.html
52 |   |   |   |       |   +-- IJG_README
   |   |   |   |       |   +-- lgpl-v21.txt
54 |   |   |   |       |   +-- mpl-v11.txt
   |   |   |   |       |   +- pixman-licenses.txt
56 |   |   |   |       +-- about.html
   |   |   |   |       +-- swt-debug.jar.back
58 |   |   |   |       +-- swt.jar
   |   |   |   |       +- swt.jar.back
60 |   |   |   +-- macosx
   |   |   |   +- swt
62 |   |   |   |       +-- about_files
   |   |   |   |       |   +-- IJG_README
64 |   |   |   |       |   +- mpl-v11.txt
   |   |   |   |       +-- about.html
66 |   |   |   |       +-- swt-debug.jar
   |   |   |   |       +- swt.jar
68 |   |   |   +-- osIndependent
   |   |   |   |   +-- commons-collections-3.2.jar
70 |   |   |   |   +-- commons-configuration-1.3.jar
   |   |   |   |   +-- commons-lang-2.2.jar
72 |   |   |   |   +- commons-logging-1.1.jar
   |   |   |   +-- siafu-context-remotecontrol-access-ws.jar
74 |   |   |   +-- siafu-context-remotecontrol-access-ws-src.jar
   |   |   |   +-- Siafu.jar
76 |   |   |   +- windows
   |   |   |       +- swt
78 |   |   |           +-- about_files
   |   |   |           |   +-- IJG_README
80 |   |   |           |   +- mpl-v11.txt
   |   |   |           +-- about.html
82 |   |   |           +-- swt-debug.jar
   |   |   |           +- swt.jar
84 |   |   +-- run_siafu.bat
   |   |   +-- run_siafu.sh
86 |   |   +-- run_siafu_ws.bat
   |   |   +-- run_siafu_ws.sh
88 |   |   +-- SimulationModels
   |   |   |   +-- Glasgow.jar
90 |   |   |   +-- Leimen.jar
   |   |   |   +-- Office.jar
92 |   |   |   +-- TestLand.jar
   |   |   |   +- Valencia.jar
94 |   |   +-- War
   |   |   |   +- SiafuWS.war
96 |   |   +- WSDL
   |   |       +-- SiafuContextWS.wsdl
98 |   |       +- SiafuRemoteWS.wsdl
   |   +- yourpackage
100|       +-- ContextNode.fractal
   |       +- YourComponent.fractal
```

```
102
33 directories, 67 files
```

"Exemple complet de construction d'un nœud de qualité de contexte"