Models and algorithms for managing quality of context and respect for privacy in the Internet of Things

Samer Machara Marquez

Thèse n° 2015SACLL005

Thèse de Doctorat
de l'Université Paris-Saclay,
préparée à Telecom SudParis

École Doctorale N° 580
Sciences et Technologies de l'Information et de la Communication

Spécialité : Informatique

Par

Samer MACHARA MARQUEZ

Models and Algorithms for Managing Quality of Context and Respect for Privacy in the Internet of Things

Thèse présentée et soutenue à Évry, le 17 Novembre 2015

Composition du jury :

M. Antoine Beugnard	Professeur à Télécom Bretagne	Rapporteur
Mme Michelle Sibilla	Professeur à l'Université Toulouse 3	Rapporteur
Mme Isabelle Demeure	Professeurà Télécom ParisTech	Examinateur
Mme Sophie Chabridon	Maître de Conférences (HDR) à Télécom SudParis	Encadrante
Mme Chantal Taconet	Maître de Conférences (HDR) à Télécom SudParis	Directrice de thèse

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

Acknowledgment

I would like to express my sincere gratitude to my advisors Prof. Chantal Taconet and Sophie Chabridon for the continuous support of my Ph.D study and research, for their patience, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Antoine Beugnard, Prof. Michelle Sibilla, and Dr. Isabelle Demeure, for their interest in my work.

I thank my fellow labmates in for the stimulating discussions, and for all the fun we have had in the last four years.

I thank my friends Troy, Ender, Carina, Jay, Gustavo, Ernesto, Willy, Sebastien, and Glenda. I am lucky to have met they here, and I thank them for their friendship, love, support and encouragement to finish my thesis.

Last but not the least, I would like to thank my family for their constant love and supporting me spiritually throughout writing this thesis and my life in general.

I dedicate this thesis to my wife, Francis Andrea,
who did more than her share around the house as I sat at the computer.
With her support and love, she gave me the force to ended my Ph.D.

Samuel Elias, this thesis is also for you.

Résumé

L'Internet des Objets (IdO) est un nouveau paradigme, dont l'idée de base est de connecter un grand nombre d'objets de façon ubiquitaire. Ces objets sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisins en partageant des données acquises directement en mesurant certains faits. Tout cela, afin d'atteindre des objectifs communs [Giusto et al., 2010]. Cette information ne représente pas seulement l'état des utilisateurs, mais aussi les processus dans lesquels ils sont impliqués, on appelle cela le *Contexte*. Le *Contexte* fournit des informations à la fois pour la reconnaissance des opérations nécessaires et leur traduction vers les services disponibles en fournissant une vue structurée et unifiée du monde dans lequel un système fonctionne [Coutaz et al., 2005].

Avec l'IdO, de nombreuses applications récupèrent des informations de contexte concernant les utilisateurs (propriétaires de contexte) tels que leurs habitudes, leurs comportements ou leur état émotionnel. Ceci offre alors beaucoup d'avantages aux utilisateurs, mais aux dépends de leur intimité. La problématique de recherche de cette thèse se situe dans la vision orientée sémantique de l'IdO proposée par [Atzori et al., 2010] considérant comment représenter, stocker, organiser et explorer les informations générées par l'IdO. Cette vision privilégie l'exploitation des solutions de modélisation pour intégrer la notion de sécurité de la vie privée dans l'IdO.

Les applications et services (consommateurs de contexte) qui sont sensibles au contexte, attendent des données de contexte correctes et fiables afin d'adapter leurs fonctionnalités [Filho, 2010]. Dans cette thèse, la qualité de Contexte (QoC) est une métadonnée qui est associée aux informations de contexte. Elle décrit une série de critères exprimant la qualité de l'information de contexte. Ces métadonnées peuvent être utilisées pour déterminer la valeur de l'information pour une application particulière dans une situation particulière. Nous explorons des solutions intergicielles pour intégrer la gestion de la vie privée et de la QoC dans l'IdO.

Cette thèse se distingue des autres recherches du domaine de la gestion de contexte en tenant compte du découplage entre les participants de l'IdO, à savoir les propriétaires des informations de contexte et les consommateurs de ces informations de contexte. De plus, nous considérons la QoC comme un facteur affectant la vie privée des individus.

Synthèse des contributions

Cette thèse fournit les contributions suivantes selon deux axes:

- **Axe 1** : Concevoir un méta-modèle de contrat de contexte pour définir la vie privée et la QoC pour exprimer les préoccupations des propriétaires et des consommateurs de contexte.

  Cette conception est basée sur deux points : Premièrement, nous considérons que la vie privée est la capacité des propriétaires de contexte à contrôler quoi, comment, quand, où et avec qui partager des informations. Par conséquent, nous identifions quatre dimensions de la vie privée (le but, la visibilité, la rétention, la QoC), pour les utiliser au moment de la définition des politiques et des

obligations d'accès. Deuxièmement, les consommateurs de contexte attendent un certain niveau de QoC afin d'accomplir leurs tâches.

Nous proposons un modèle de contrat conforme à la Loi des Etats-Unis de 1974 sur la protection de la vie privée et aux lois européennes concernant la protection des renseignements personnels [OECD, 1980, EU, 1995, EU, 2002]. Nous définissons deux types de contrats de contexte :

1. Les **contrats de production de contexte** définissent les clauses pour la production de données de contexte avec des exigences de confidentialité (indiquant les exigences des propriétaires de contexte avant d'accepter de fournir des données de contexte) et des garanties de QoC (établissant les garanties que le producteur de contexte est prêt à remplir en ce qui concerne la QoC).
2. Les **contrats de consommation de contexte** définissent les clauses de consommation de données avec des exigences de QoC (indiquant le niveau de QoC que le consommateur attend afin de permettre l'exécution d'une application), et des garanties de confidentialité (indiquant les garanties proposées par le consommateur afin de protéger la vie privée des propriétaires des informations de contexte).

Dans cette thèse, la « confiance » est une probabilité subjective par laquelle un individu A (fiduciant) attend qu'une autre personne B (fiduciaire) effectue une action déterminée de façon fiable et en toute sécurité dans une situation donnée avec un sentiment de sécurité relative, même si des conséquences négatives sont possibles.

La QoC et la vie privée sont liées à la notion de « confiance », mais sous deux angles différents. Du point de vue du producteur de contexte, la priorité doit être accordée au respect de la vie privée. Si un consommateur de contexte n'est pas digne de confiance, une plus stricte confidentialité est souhaitée. Du point de vue du consommateur de contexte, une qualité d'information élevée est attendue à partir de sources dignes de confiance. Dans le cas contraire, d'éventuelles erreurs peuvent se produire.

La solution présentée dans cette thèse décrit le méta-modèle de contrat pour les consommateurs de contexte et les producteurs de contexte que nous avons proposé pour le cadriciel MUCONTEXT défini dans le cadre du projet ANR INCOME[1]. Chaque type de contrat de contexte est créé sans connaître l'autre partie. Afin de faire correspondre ces deux types de contrats, il est important d'inclure le paramètre de « confiance » permettant à chaque partie d'assouplir ses exigences.

Ce méta-modèle définit les éléments de protection de la vie privée, les informations de QoC et le contexte lui-même impliqués lors de l'activation des règles d'accès aux données de contexte.

La notion de contrat augmente le sentiment de sécurité et développe la confiance entre les parties, parce que des limites claires sont établies entre ce qu'une partie attend et ce qu'elle est prête à faire pour l'autre partie. Notre méta-modèle est conçu pour être simple à utiliser tout en préservant l'expressivité nécessaire. Ceci est important car les utilisateurs peuvent comprendre et mettre en

[1] http://anr-income.fr

place facilement comment leurs informations sont utilisées par les applications de l'IdO. Ce méta-modèle de contrat de contexte peut être ajouté à des méta-modèles de gestion de contexte existants tels que [Taconet et al., 2009].

- **Axe 2** : Proposer un algorithme pour créer des accords entre les producteurs et les consommateurs de contexte en évaluant et en comparant les exigences et les garanties indiquées sur leurs contrats de contexte respectifs.

  Comme les deux parties en présence, producteurs et consommateurs d'informations issues de l'IdO, ont des contrats symétriques, quand un participant définit ses besoins, l'autre définit ses garanties. Le processus de correspondance de ces contrats de contexte vérifie si les exigences de l'une des parties sont couvertes dans les garanties offertes par l'autre partie. Par conséquent, prendre une décision basée sur cette compatibilité du point de vue du producteur, conduit à fournir ou non des données contextuelles. Du point de vue du consommateur, l'accès à des données de contexte est autorisé ou refusé. A partir de cette définition, nous avons conçu des algorithmes pour déterminer si la fourniture et la consommation sont autorisées ou non, selon la correspondance entre les contrats de contexte.

Afin de valider notre approche dans un cadre opérationnel, nous comparons la performance de notre solution par rapport à une solution basée sur le standard XACML pour la définition de politiques de contrôle d'accès. Nous définissons une procédure de traduction pour transformer les contrats de contexte au format XACML afin d'appliquer notre algorithme pour les implémentations existantes de XACML avec un minimum de modifications. Nous pouvons donc intégrer notre méta-modèle ainsi que l'algorithme de traduction de contrat dans les cadriciels de l'IdO existants ou futurs. Nous avons choisi XACML car il possède toutes les caractéristiques pour exprimer les éléments de nos contrats et aussi pour sa popularité. Les détails sur notre choix sont explicités dans la section 2.7. Afin de valider notre méta-modèle de contrat et l'algorithme d'appariement, nous avons réalisé une étude de cas et mis en œuvre l'entité correspondant au "Point de Décision de la Politique" de XACML (PDP) pour évaluer les contrats traduits.

L'étude de cas que nous proposons est divisée en mini-scénarios correspondant à chaque besoin considéré dans cette thèse (voir la section 4.3). Nous illustrons les raisons pour lesquelles la QoC est nécessaire pour atteindre les objectifs d'un système, comment la QoC affecte la vie privée et les cas où la confiance contribue à améliorer la QoC reçue. De cette étude de cas, nous extrayons quelques exemples précis où les objectifs de l'application ne peuvent pas être atteints en raison d'une QoC insuffisante. De plus, nous montrons comment une augmentation de la QoC demandée peut compromettre la vie privée.

Les méta-modèles de contrat proposés sont bénéfiques à la fois pour définir des contrats, et aussi pour automatiser la gestion de la livraison des informations de contexte avec la QoC appropriée et une protection de la vie privée satisfaisante. En ce qui concerne le premier avantage, les contrats fournissent un catalogue des contraintes et des modèles pertinents et utiles pour construire des applications sensibles au contexte, considérant que les propriétaires de contexte définissent et adaptent indépendamment des règles de confidentialité. Concernant le second avantage, les contrats de production sont disponibles à l'exécution et peuvent être modifiés à tout moment par les propriétaires de contexte. Ces modèles de contrats offrent ainsi une approche globale afin de correspondre dynamiquement aux exigences et aux

garanties de vie privée et de QoC. Chaque moitié de contrat évolue de manière autonome et est interprétée par un gestionnaire de contexte. La correspondance entre les deux parties d'un contrat prend en compte les effets dynamiques concernant la confiance actuelle entre les producteurs, les consommateurs, les données contextuelles provenant de l'environnement, et l'organisation des participants. De plus, concernant la protection de la vie privée, il devient possible de garantir le bon fonctionnement des applications en prenant en compte la gestion de la QoC et ainsi de favoriser la confiance entre les participants de l'IdO.

Plan du document

Après l'introduction du travail de cette thèse dans le premier chapitre, le présent document est divisé en deux parties et une conclusion. La première partie (chapitres 2 et 3) présente les concepts généraux, les fondamentaux et l'état de l'art liés à l'IdO, la QoC, la notion de confiance, et le respect de la vie privée. La deuxième partie (chapitres 4, 5, 6 et 7) de ce travail, présente nos contributions liées aux contrats de QoC et de vie privée entre producteurs et consommateurs de données de contexte dans le cadre de l'IdO. À cet effet, nous proposons un méta-modèle spécifique pour l'IdO et un middleware pour mettre en correspondondance les contrats de production et de consommation de contexte. Ce document est organisé en sept chapitres, en plus de l'introduction, comme décrit ci-dessous.

- Le **Chapitre 2** présente une étude et une analyse d'un ensemble de concepts clés (c.-à-d l'IdO, la QoC, la confiance et la vie privée), puis introduit les technologies (langages et cadriciels), les paradigmes (notion de contrat) et les lois relatifs au sujet de cette thèse. Ce chapitre présente également la liste des critères de protection de la vie privée que nous avons sélectionnés pour comparer les travaux connexes.
- Le **Chapitre 3** étudie plusieurs modèles et cadriciels existants pour la protection de la vie privée et la gestion de la QoC au sein de l'IdO. Nous étudions en particulier s'ils considèrent les questions suivantes identifiées pour l'IdO : le découplage entre les producteurs et les consommateurs, la gestion de la confiance et la manipulation d'événements dynamiques. Ce chapitre résume différentes solutions qui ont contribué à la définition de la vie privée et à l'identification de ses différentes dimensions. En outre, ce chapitre montre que certains travaux de recherche proposent d'inclure QoC et respect de la vie privée dans les systèmes sensibles au contexte. Enfin, le chapitre montre la conformité de chaque solution proposée avec la liste des critères de protection de la vie privée que nous avons introduite dans le chapitre 2.
- Le **Chapitre 4** présente un cas d'utilisation en guise d'illustration. De ce cas d'utilisation, nous extrayons les exigences concernant la gestion de la vie privée et de la QoC dans l'IdO. Nous présentons également l'architecture proposée pour la gestion des contrats de QoC et de respect de la vie privée.
- Le **Chapitre 5** présente notre contribution pour la modélisation des contrats dans le cadriciel MUCONTEXT. Nous avons suivi une approche dirigée par les modèles pour formaliser une représentation standard pour l'expression des exigences et garanties côté producteur et côté consommateur de contexte. Enfin, ce chapitre décrit l'évaluation des contrats utilisant MUCONTEXT par une approche qualitative.

- Le **Chapitre 6** détaille le processus d'appariement des contrats MUCONTEXT pour créer les accords nécessaires pour le partage de données de contexte entre les producteurs et les consommateurs.
- Le **Chaptitre 7** présente l'approche suivie pour traduire les contrats MUCONTEXT en politiques XACML. Grâce à cette approche, un contrat MUCONTEXT peut être manipulé par un standard bien connu concernant le contrôle d'accès et la protection de la vie privée.
- Le **Chapitre 8** conclut la thèse en synthétisant les contributions de notre travail et propose plusieurs perspectives de poursuite de ce travail.

Mots-clés: Internet des Objets (IdO), Vie privée , Qualité de contexte (QoC), Contrats, Ingénierie dirigée par les modèles (IDM).

Abstract

The Internet of Things (IoT) is a novel paradigm, which basic idea is the pervasive presence around us of a variety of things or objects that are able to interact with each other and cooperate with their neighbors by sharing data, directly acquired by measuring some facts, in order to reach common goals [Giusto et al., 2010]. This information not only represents the state of users but also the processes in which they are involved, this is called the *Context*. The *context* informs both the recognition and mapping of operations onto available services by providing a structured, unified view of the world in which a system operates [Coutaz et al., 2005].

With the IoT, many applications consume context information concerning users (context owners) such as, daily routines, behaviors, health or emotional state, offering lots of benefits to users, but compromising their privacy. The research problematic of this thesis lies within the "semantic-oriented" IoT vision proposed by [Atzori et al., 2010]) considering issues related to how to represent, store, organize, and search information generated by the IoT. This vision favors the use of appropriate modeling solutions for integrating privacy protection into the IoT.

Context-aware applications and services (context consumers) expect correct and reliable context data to adapt their functionalities [Filho, 2010]. In this thesis, the Quality of Context (QoC) corresponds to meta-data attached to context information describing a range of criteria that express context information quality. These meta-data can be used to determine the worth of the information for a particular application in a particular situation. We explore middleware and framework solutions to integrate the management of privacy and QoC in the IoT.

This thesis can be distinguished from other context management domain researches by bearing in mind the decoupling of the IoT participants, i.e., the owners of context information and the consumers of this context information. Moreover, we consider QoC as a factor affecting the privacy of individuals. This thesis provides the following contributions along two axis:

- **axis 1** Designing a Context Contract Meta-model to define privacy and QoC concerns of decoupled context owners and context consumers based on reciprocal trust. This meta-model has been proposed for the MUCONTEXT framework in the ANR INCOME[2] project.

  This design is based on two points. Firstly, we consider that privacy is the capacity of context owners to control what, how, when, where and with whom to share information. Therefore, we identify four privacy dimensions (purpose, visibility, retention, QoC), and use them in the definition of access policies and obligations. Secondly, context consumers expect a certain QoC level in order to perform their tasks. We then propose to define two kinds of context contract for the producer and the consumer sides as follows:

  1. A **Context producer contract** has clauses expressing the production of context data, privacy requirements, and QoC guarantees;
  2. A **Context consumer contract** has clauses expressing the consumption of context data, QoC requirements, and privacy guarantees.

[2] http://anr-income.fr

Each context contract is created without the knowledge of the other side contract. In order to match two contracts, it is important to include one parameter that allows each party to relax their requirements, which is the *Trust*. In this thesis, *trust* is a subjective probability by which an individual A (trustor), expects or believes that another individual B (trustee), performs a given action dependably, securely, and reliably in a given situation within a specified context with a feeling of relative security, even though negative consequences are possible.

Both QoC and privacy are related to the notion of *trust*, but from two different perspectives. From a context producer point of view, priority must be given to the respect of the privacy. If a context consumer is not trustworthy, a strict privacy is desired. From a context consumer point of view, a piece of context information of high quality is expected from trustworthy sources. If not, possible errors might occur.

- **axis 2** Proposing an algorithm to create agreements among context producers and context consumers by evaluating and compare requirements against guarantees, stated on their respective context contracts.

  As both IoT participants have symmetric contracts, when one participant defines its requirements, the other one defines its guarantees. The matching process of these context contracts verifies if the requirements of one party are included within the guarantees offered by the other party. Therefore, taking a decision based on this compatibility match from the producer point of view is to permit or deny the access to context data. Additionally, from a consumer point of view, the consumption of context data is permitted or denied. From this definition, we designed algorithms to determine whether access and consumption are authorized or not, according to the context contracts matching.

  In order to validate our approach in an operational setting, we evaluate the performance of our solution making a comparison with contracts based on the XACML standard for access control. We create a translation procedure to transform context contracts into the XACML format. This allows to apply our algorithm to existing implementations of XACML with minimal modifications.

Keywords: IoT, Privacy, Quality of Context (QoC), Contracts, MDE.

Contents

Part I Fundamentals and State of the Art on Privacy, QoC, Trust and contracts for the IoT

Conclusions and Perspectives

Appendices 161

List of Figures

Chapter 1

Introduction

Contents

1.1 Motivation and Problem Description

Day after day, the Internet is being extended with the interconnection of a big range of small devices (e.g., Sensors and smart objects) to build the Internet of Things (IoT) [Ma, 2011]. This enables everyday objects to share information among themselves and/or with other systems, thus providing events occurring in a world wide network to applications. As a consequence, mobile applications become context-aware taking into account events occurring in the environment. With the IoT, many applications can consume context data concerning user activities, behaviours, daily routines, health and welfare, which brings a lot of possible benefits to the user. However, as pointed out by Buckley [Buckley, 2006], the IoT raises critical issues concerning privacy. Agre and Rotenberg [Agre and Rotenberg, 1998] define privacy as the power someone has over a piece of information that belongs to him or her. Concretely, we define privacy as the capacity of control about what, how, when, where and with whom to share information.

Figure 1.1 introduces the vocabulary used in this document. The *Context owner*, (i.e., a person, a group of persons, or an organization) is the entity having the capacity of decision and control about privacy rules over his/her/its context data. Software and hardware entities providing these context data are named *Context producers*. Entities that require context data, i.e., context-aware applications, are called *Context consumers*. Persons that access those applications are called *Context end-users*. The middleware between consumers and producers is the *Context manager*; it is in charge of allowing or denying access to context data autonomically to consumers with the appropriate level of Quality of Context (QoC), while preserving the privacy of the context owner.

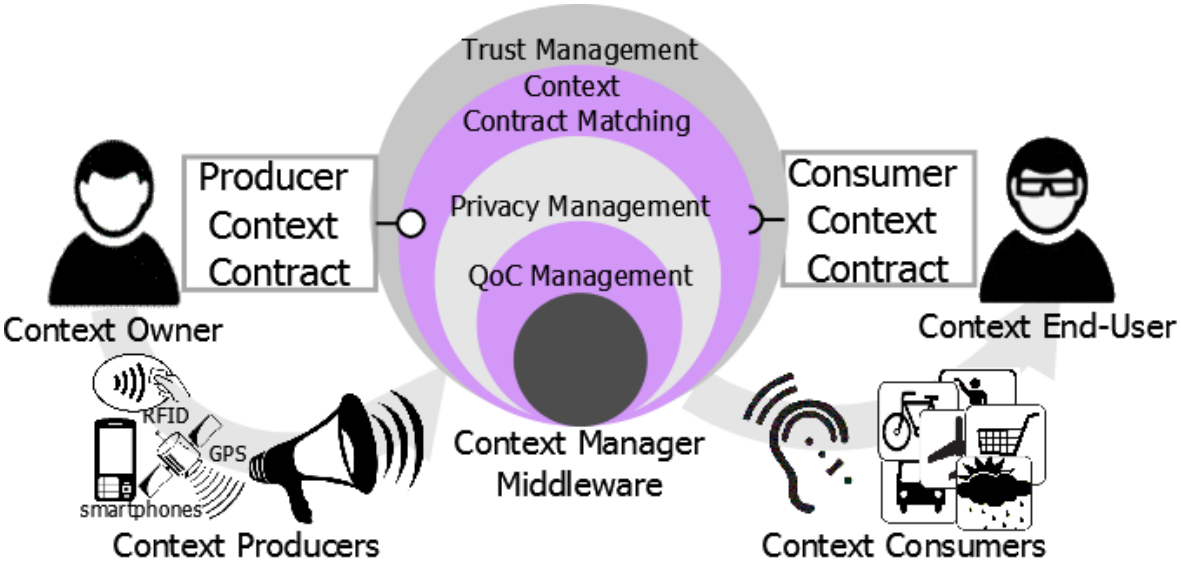


Figure 1.1: Distributed view of producers, consumers and contracts

Taking into account QoC is essential for IoT applications for the following reasons: *(i)* The disparity and instability of the producers, *(ii)* Different ways to manage the information, *(iii)* Difficulties to recover the information. QoC management must therefore provide mechanisms to reduce the impact that all these situations can produce over the ambient services.

QoC is defined by a set of measurable quality criteria such as precision, error probability or freshness [Buchholz et al., 2003]. Each context consumer has different requirements on each quality criteria according to its needs. Some context consumers require more precision, others more freshness, and so on. Through QoC, the worth of context data for a specific application is evaluated. For example, to take a decision or to guarantee an appropriate quality of service, a minimum level of QoC is required.

One of the main challenges of building trusted context-aware applications for the IoT is that context producers and context consumers are decoupled, i.e., they run on remote devices and are not aware of each other. Furthermore, in some cases, context owners want to remain anonymous to protect their privacy. It is therefore necessary to find a trade-off between the context owners'privacy requirements and the level of QoC required by applications. That is to say, how much QoC is enough to achieve the objective of an application without delivering the maximum available QoC so as to preserve privacy. However, limiting the QoC provided for privacy purpose is interesting, but not sufficient. To solve the trade-off between privacy and QoC, trust plays an important role. While tuning QoC for privacy, we are interested in determining if context owners trust context end-users, as well as, if context consumers trust the context data collected by context providers. The higher the level of trust between owners and consumers, the higher the level of QoC consumers will be allowed to get and the higher the quality and/or the performance of the applications.

A part of the problem is how to match producers' restrictions in terms of privacy with consumers' requirements in terms of QoC. Additionally, how to establish an agreement about which metrics must be

used to determine if privacy is respected and QoC is reached. It means that, at some point, users should be aware of how much privacy was actually respected and should be able to modify their requirements at runtime if necessary. Therefore, applications should promote their trustworthiness and should not ask for more than what they really need. How can users be sure that these applications are secure enough against privacy violations? How can they trust these applications? Transparency is the key; systems should provide clear mechanisms to watch what they are doing at runtime and an easy way to configure them. The issue is to find a mechanism where consumers and producers can reach an agreement about the amount of information needed to respect both privacy and usefulness of information.

We propose to answer this issue with contracts between consumers and producers that include clauses on privacy and QoC. The contract must define the conditions in which producers and consumers accept to share/receive data. From the producer point of view, the contract must help to define clearly the level of privacy protection required and, from the consumer point of view, the contract must express the QoC required for the proper functioning of applications. In operational terms, the consumer asks for a determined QoC in order to guarantee the system output and the producer decides to share its data or not depending on the trade-off in terms of privacy requirements. This type of contract is essential in the development of the IoT, since it enables producers to trust consumers and at the same time this will allow applications to be more efficient and produce assertive decisions based on data with a sufficient level of QoC.

1.2 Research Goal and Sub-Goals

This thesis has for ambition to provide meta-models to express the requirements in terms of QoC and privacy protection among the participants of the IoT as well as the matching algorithms to apply these meta-models through the different IoT context applications, for both the industry and the general public by addressing the main challenges mentioned earlier.

The main objective of this PhD thesis has been split into two sub-objectives, indicating the steps to be followed in this research:

1. **To express a meta-model to define requirements and their variability, particularly concerning QoC and Privacy for the decoupled participants, context producers and context consumers, of the IoT.**

   When designing context-aware systems, developers must be able to express their contractual requirements in terms of their extra-functional properties for context information. Thus, IoT systems will be able to better manage their adaptations. The requirements must be able to vary during execution under certain predefined conditions. Symmetrically, the context management system must identify the QoC information and privacy provided and / or used.

2. **To design an algorithm to evaluate and compare the stated requirements against guarantees defined by the IoT participants to create agreements to share context information.**

   The matching process of the meta-model instances verifies if the requirements of one party are included in the guarantees offered by the other party. Thus, determining whether access and consumption are authorized or not, according to the matching process.

This thesis studies solutions in terms of, on the one hand, models with the design and the execution of ambient systems for taking into account extra-functional constraints, and on the other hand, mechanisms to evaluate the expressed requirements.

1.3 Approach

We first conducted a survey on the concepts of IoT, QoC, Trust, Privacy, Contract Formalization and Privacy Policy Languages. These concepts are the backbone of our thesis.

We then performed a study on privacy taxonomies and privacy protection criteria to identify the main characteristics, requirements and concepts involved in respecting privacy taking into account the constraints imposed by the IoT to determine the information required to design privacy contracts. Additionally, based on these privacy protection criteria and some privacy protection mechanisms, we defined a list of criteria to compare previous works related to our investigation.

Furthermore, we identify existing suitable meta-models to be part of our meta-model solution, such as, QoC and trust models. Our final aim is to create a meta-model which includes all these elements together to define contracts to protect the privacy of IoT users.

To answer the thesis objectives presented in Section 1.2, we propose the following approach:

Model Driven Engineering

The model driven engineering paradigm allows one to define specific meta-models. These meta-models allow then to design and develop models that will act both statically (design life cycle) and dynamically (runtime life cycle). Statically, the model ensures that contracts can be checked by the context manager in order to be used to generate composite objects which audit contracts. For example, one can determine statically freshness constraints that should be checked at runtime. Dynamically, the model ensures that contracts can evolve depending on the context information in order to be used to generate compositions of objects at runtime. For example, the freshness of information can be checked at runtime to fit the needs of context-aware applications.

1.4 Contributions

We propose context contracts compliant with the U.S. Privacy Act of 1974 and European laws concerning privacy [OECD, 1980, EU, 1995, EU, 2002].

We define two kinds of context contracts. *Producer context contracts* define clauses for the production of context data with privacy requirements (indicating the context owner demands before accepting to provide context data) and QoC guarantees (establishing the guarantees the producer is ready to fulfill with respect to QoC). *Consumer context contracts* define clauses for the consumption of context data with QoC requirements (establishing the QoC the consumer is expecting for running an application), and privacy guarantees (indicating the guarantees the consumer agrees on in order to protect the privacy of the context owner).

The solution presented in this thesis describes privacy and QoC meta-models to create rules for combining them. We create the MUCONTEXT Contract meta-model for context consumers and context producers in the course of the ANR INCOME project. The notion of contract increases the feeling of security and improves trust because it establishes clear limits of what one part expects and what it is willing to do for the other part. Our language is intended to be less wordy than current privacy policy languages preserving the required expressiveness. This is important because users can understand and easily set up how their information is used by IoT applications. These meta-models can be added to an already existing context management meta-model [Taconet et al., 2009].

As producers and consumers are decoupled, we create a symmetric MUCONTEXT contract meta-model for both of them. Such contracts allow both producers and consumers to use the same concepts to define their guarantees and requirements. Based on this, we design an algorithm to match the guarantees of ones with the requirements of the others to create agreements to access context data.

We define a translation process of our MUCONTEXT Contracts into the XACML language. Therefore, we can integrate our MUCONTEXT Contract meta-model and matching algorithm into existing or future IoT frameworks. We choose XACML because it possesses all the features to express the elements of our MUCONTEXT Contracts and also for its popularity. More details on this decision can be found in Section 2.7.

In order to validate the MUCONTEXT contract meta-model and matching algorithm, we designed a case study and implemented an XACML Policy Decision Point (PDP) to evaluate the translated contracts.

The case study we propose, is split in mini-scenarios each corresponding to one of the identified requirements addressed by this thesis (See Section 4.3). We illustrate why being aware of QoC is necessary to achieve the objectives of a system. That is how QoC affects privacy and in which cases trust helps to improve QoC. From this case study, we extract some examples of where the objectives of the application could not be reached due to poor QoC. Likewise, we show how increasing QoC could compromise privacy.

The proposed MUCONTEXT contract meta-models are beneficial both: *(1)* for defining contracts, and *(2)* for automating the management of context delivery with appropriate QoC and privacy protection. Regarding the first benefit, contracts provide a guide to formulate and develop rules, constraints and models relevant and useful for building context-aware applications, whereas context owners independently define, and then adapt their privacy rules. Regarding the second benefit, the producer MUCONTEXT contract are available at runtime and may be modified at anytime by context owners. Such contract models offer a comprehensive approach to dynamically match QoC/privacy requirements and guarantees. Each half contract participates to an autonomic matching performed by a context manager. The matching takes into account dynamic facts concerning the current trust among producers, consumers, context data coming from the environment, and the organization of the participants. Considering privacy protection, guaranteeing the smooth operation of applications through QoC management will enable trust among IoT participants.

1.5 Layout of the document

After introducing the work of this thesis in this first chapter, this document is divided into two parts and a conclusion. The first part (Chapters 2 and 3) presents general concepts, fundamentals and state of the art related to IoT, QoC, Trust, Privacy. The second part (Chapters 4, 5, 6 and 7) of this work, presents our contributions related to QoC and privacy agreements between producers and consumers of data in the IoT. For this purpose, we propose a specific meta-model for the IoT and a middleware to match producer and consumer agreements.

This document is organized into 8 (eight) Chapters, including this introduction, as described below.

- **Chapter 2** presents a study and analysis of a set of key concepts (i.e., IoT, QoC, Trust and Privacy), technologies (languages and frameworks), paradigms (notion of contract) and laws concerning the subject of this thesis. This chapter also presents the list of protection criteria we have selected to study related works.
- **Chapter 3** studies several existing models and frameworks for privacy and QoC enforcement in the IoT. We specifically study if they consider the following issues identified for the IoT: the decoupling of producers and consumers, trust management and the handling of dynamic events. This chapter summarizes different works that have contributed to the definition of privacy and the identification of its different dimensions. As well, this chapter shows that some research works propose to include QoC and Privacy concerns into context-aware systems. Finally, the chapter shows the compliance of each proposed solution with the selected privacy protection criteria list introduced in Chapter 2.
- **Chapter 4** presents an illustrative use case. From this use case, we extract requirements concerning the management of privacy and QoC in the IoT. We also introduce the proposed architecture for managing QoC and privacy agreements.
- **Chapter 5** presents our contribution: MUCONTEXT contracts. We have followed a model-driven approach to formalize a standard representation to express the requirements and guarantees of both producers and consumers. Finally, this chapter describes the evaluation of the MUCONTEXT contracts following a qualitative approach.
- **Chapter 6** introduces the matching process of MUCONTEXT contracts to create agreements necessary for sharing context data between producers and consumers.
- **Chapter 7** presents the approach followed to translate MUCONTEXT contracts into XACML policies and requests. Through this approach, MUCONTEXT contracts may be handled using a well known standard for access control and privacy.
- **Chapter 8** concludes the thesis by synthetising the contributions of our work, as well as exposing perspectives.

Part I

Fundamentals and State of the Art on Privacy, QoC, Trust and contracts for the IoT

Chapter 2

Fundamentals and Models of Privacy, QoC, Trust and Contract for the IoT

Contents

2.1 Introduction

This chapter presents a study and analysis of a set of key concepts, technologies, paradigms and laws in order to understand the subject of this thesis. In this path, several relevant research works were analyzed from the management of personal, sensitive information to the study of the different European privacy laws. They give us a guideline, even if they were not exactly tackling the same issue. Moreover, many technologies were studied during the development of this thesis. Next are presented those selected as the most suitable proposals to solve the problem. This chapter is organized as follows:

Section 2.2 presents the concept of IoT used in this thesis and positions our work within the IoT paradigm vision. The next section (2.3) introduces the concept of QoC considered in this thesis. Besides, it presents the meta-model chosen to represent the QoC. Section 2.4 presents the concept of Trust used in this thesis. Furthermore, it shows the meta-model chosen to represent the Trust in our solution. Section 2.5 clarifies what is privacy in the frame of this thesis and describes the main dimensions that compose privacy. It also discusses some protection criteria that can be used to protect privacy. Section 2.6 explains the notion of contract by making an analogy with real life contracts. Furthermore, it presents the model-driven approach followed to formalize the contracts. Section 2.7 focuses on suitable languages and frameworks for the definition of privacy policies in order to choose a suitable language to represent context contracts.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects, such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, which are able to interact with each other and cooperate with their neighbors to reach common goals [Giusto et al., 2010].

In fact, IoT semantically means “a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols” [INFSO, 2008]. This implies that a huge number of (heterogeneous) objects are involved in the process.

Many are the domains and the environments in which new IoT applications would likely improve the quality of our lives, for example: transportation and logistics domain, health care domain, smart environment (home, office, plant) domain, personal and social domain. Figure 2.1 presents some of the potential applications among the inconceivable number of solutions that can provide the IoT.

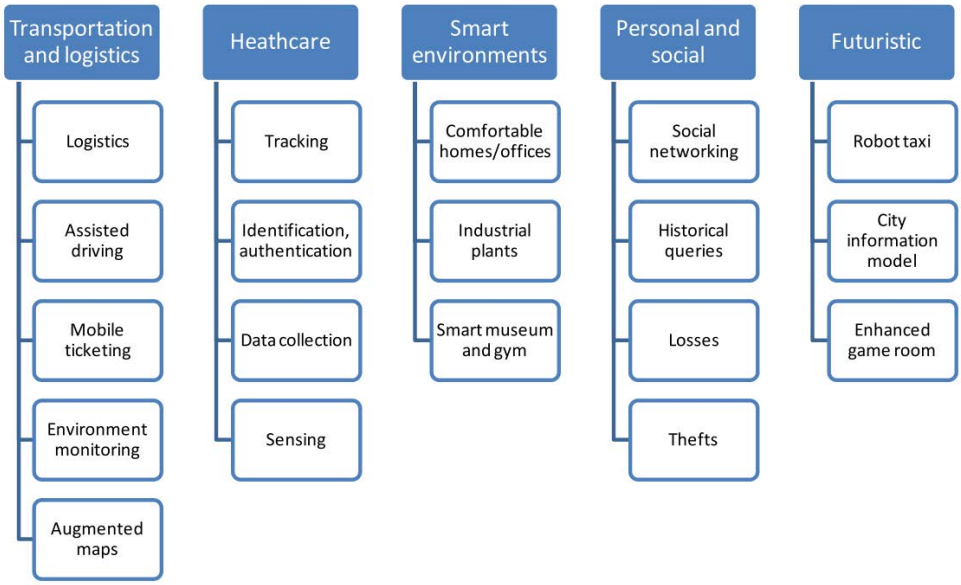


Figure 2.1: Applications domains and relevant major scenarios [Atzori et al., 2010]

[Atzori et al., 2010] conducted a survey on the IoT paradigm, which describes the different visions that the research community has about an IoT definition. They identify the technologies that enable the IoT, and describe the remaining open issues. In this section, we connect these aspects with the subject of our work.

2.2.1 IoT Paradigm Vision

The IoT paradigm is the result of the convergence of the three main visions "Things oriented", "Semantic oriented" and "Internet oriented", according to [Atzori et al., 2010]. Figure 2.2 shows these visions, highlighting and classifying with reference to the IoT the technologies and standards that they contribute to characterize.

The subject of this thesis lies within the "semantic-oriented" IoT vision. The idea behind this vision is that the number of items involved in the Future Internet is destined to become extremely high. Therefore, issues related to how to represent, store, interconnect, search, organize, and share information generated by the IoT will become very challenging [Atzori et al., 2010]. Another aspect to take into account is the security. Traditional security goals like confidentiality, availability, reliability, integrity, accountability, responsibility, do not cover all the needs and threats of the IoT. In this context, semantic technologies play a key role. In fact, they can exploit appropriate modeling solutions for integrating privacy and security into the IoT. The major implication of pervasive systems on security is the risk and negative consequences for the privacy of its users. For that, it is important to include privacy from the design of such systems. The privacy problem is the main concern of our contribution.

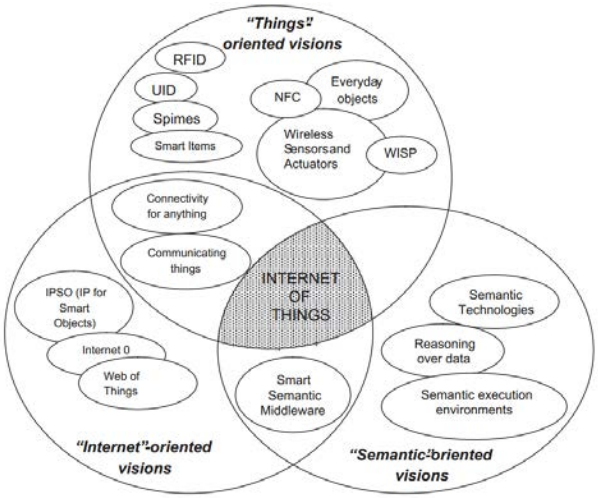


Figure 2.2: "IoT" paradigm as a result of the convergence of different visions. [Atzori et al., 2010]

2.2.2 Enabling Technologies

The most relevant technologies that make the IoT possible as identified by [Atzori et al., 2010] are "Identification, sensing and communication technologies" and "Middleware". In the context of this dissertation, we are interested in exploring middleware solutions to integrate the management of privacy and QoC in the IoT.

A middleware is a software application that provides core services like concurrency, transactions, messaging, security. The middleware can be composed of one layer or a set of sub-layers interposed between a technology and the application level. The middleware allows programmers to focus on their specific purpose, which is the development of a specific application enabled by the IoT infrastructure. The middleware does the heavy lifting by hiding the details of the different technologies implied in the IoT.

The middleware is most commonly used as software that enables communication and management of data in distributed applications. Moreover, it is gaining more and more importance in the last years due to its major role in simplifying the development of new services and the integration of legacy technologies into new ones.

[Katasonov et al., 2008] describes a solution to meet the middleware needs for the domain of the IoT. They believe that the development of a new generation of middleware platform will allow the creation of self-managed complex systems, consisting of distributed, heterogeneous, shared and reusable components of different nature.

2.2.3 Open Issues

A large research effort is still required, even if the IoT becomes feasible thanks to the already existing enabling technologies. [Atzori et al., 2010] describes the problems related to standardization activity,

addressing and networking issues, security and privacy. We focus our research on privacy protection and intend to limit a potential performance degradation of applications accessing the IoT.

People will resist to the IoT as long as they have no strong confidence that it will not cause serious threats to their privacy. The main issue related to privacy comes from the ease with which a lot of private data about a person can be collected without the person being even aware of it. The control of the diffusion of such data is impossible with current techniques.

In the next sections, we first consider the notions of QoC and trust respectively in Section 2.3 and Section 2.4. We will then address the concept of privacy later in Section 2.5.

2.3 Quality of Context (QoC)

Prior to defining the notion of QoC, it is important to make a clarification regarding context data and context information concepts. In this manuscript, both terms are used with the same meaning as in the domain of information management where data are considered as plain facts. Only when it is relevant, a distinction between data and information will be made. Context data represent useful raw data that have been acquired by a context-aware application through sensing, observing, or measuring some facts. Context information is obtained from context data that have been processed, organized, or structured in order to provide meaningful information that applications are able to interpret [Arcangeli et al., 2012].

2.3.1 QoC Definition

QoC is related to any inherent information that describes context information and can be used to determine the worth of the information for a specific application [Buchholz et al., 2003]. QoC specializes the general notion of Quality of Information (QoI) for context information.

Context-aware applications and services expect correct and reliable context data to adapt their functionalities [Filho, 2010]. Karen Henrickson et al. have determined imperfection as a characteristic of context information: *"Information may be incorrect if it fails to reflect the true state of the world it models, inconsistent if it contains contradictory information, or incomplete if some aspects of the context are not known."*[Henricksen et al., 2002]. Later on, they consider that *...context models will need to specify a range of characteristics of context information. . .* [Indulska et al., 2003]

Therefore, several research teams ([Hönle et al., 2005], [Zimmer et al., 2006]) have proposed to attach metadata to context information representing its quality. These metadata allow the improvement of the operational value of the context.

Summarizing, in this thesis the QoC corresponds to meta-data attached to context data describing a range of criteria that express context quality. These meta-data can be used to determine the worth of context information for a particular application in a particular situation.

2.3.2 QoCIM : A new QoC Meta-model

[Marie et al., 2013b] has conducted a rigorous study of several works that address QoC modelling and management. Table 2.1 summarizes the models they studied to conclude that none of them can easily

provide without adaptation the three necessary properties that they identify for an information model for QoC namely expressiveness, genericity and computability.

Model→ / ↓Wished property	OGC	IoT-A	COSMOS	CIM	OMG
Expressiveness			✓		✓
Computability	✓			✓	✓
Genericity		✓		✓	

Table 2.1: Summary of the studied models [Marie et al., 2013b]

Additionally, [Marie et al., 2013b] analyzed and compared different proposals of QoC criteria lists defined by different authors. Their analysis explicitly demonstrates the existence of divergences and concludes on the difficulty to converge to a unique and exhaustive QoC criteria list. Table 2.2 highlights their conclusions showing that there is no consensus about which QoC criteria have to be used to measure the QoC of context information. Moreover, the table provides a way to compare different lists of QoC criteria. This makes possible to compare new specific lists among them. Indeed, with the development of context-aware applications, new high level criteria could appear and Table 2.2 provides a method to organize and combine lists of QoC criteria.

Facing this situation, Marie et al. [Marie et al., 2013b] propose the meta-model, QoCIM depicted in Figure 2.3. QoCIM is dedicated to exploit and to manipulate any QoC criterion within context managers and context-aware applications.

The three main advantages of QoCIM model according to its authors are: *(i)* It is not dependent on any QoC criterion. *(ii)* It offers a unified solution to model, at design time, heterogeneous QoC criteria. *(iii)* The models based on QoCIM could be used, at runtime, by both context managers and context-aware applications, for dynamic valuation of the QoC. Thanks to these three characteristics the QoCIM model fits perfectly our needs. Therefore, we decide to integrate QoCIM as part of our solution to model contracts.

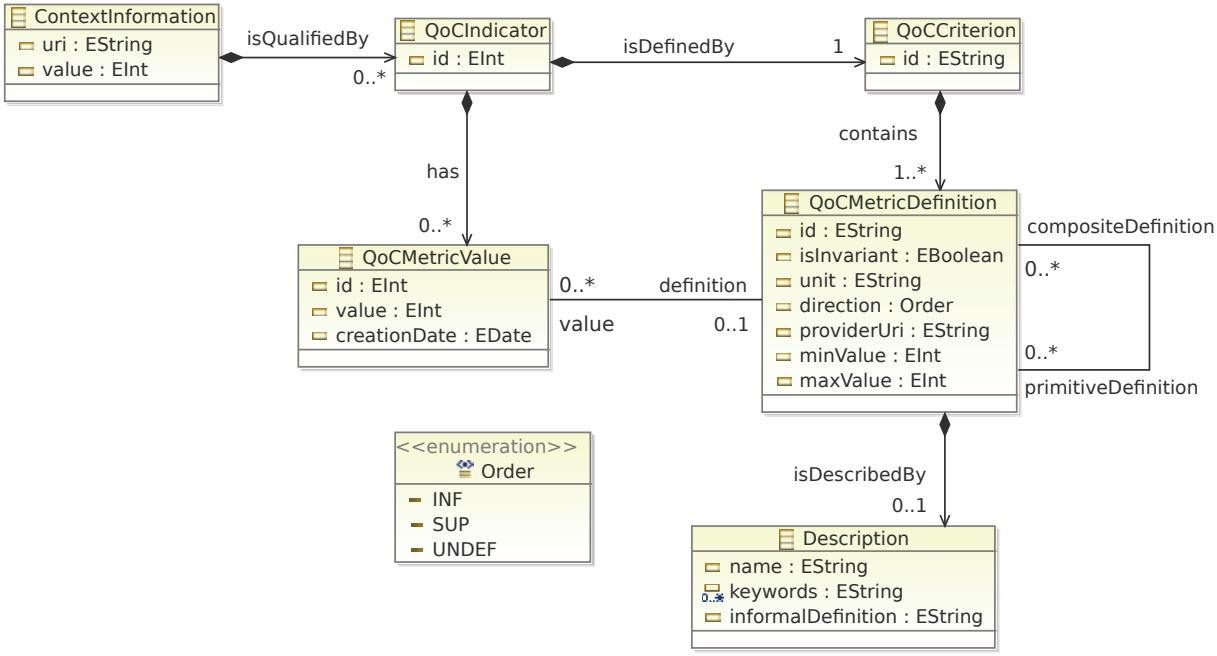


Figure 2.3: QoCIM : QoC Information Model [Marie et al., 2013b]

Chronological order

Abstraction order	Criterion \ Reference	[Buchholz et al., 2003]	[Kim and Lee, 2006]	[Sheikh et al., 2007]	[Filho, 2010]	[Manzoor et al., 2012]	[Neisse, 2012]
1	Probability context is correct, free of errors	**Correctness**	Accuracy		**Precision**	Accuracy	**Precision**
2	Max. distance for sensor to get context					*Sensor range*	
3	Location of the real world entity					*Entity location*	
4	Location of the sensor					*Sensor location*	
5	Period between two collections of context			*Temporal resolution*	✓	*Time period*	
6	Date of collection of context	✓	✓	✓	✓	*Measurement time*	*Timestamps*
7	Granularity location of context			*Spatial resolution*	**Resolution**		
8	Rate the confidence of the provider	*Trust worthiness*					
9	Critical value of context					*Significance*	
10	Granularity (detail level) of context	**Precision**		**Precision**	*Sensitiveness*	*Usability*	
11	Context consumer have access to context		✓			*Access right*	
12	Context transfers restricted and secured		Access security (11)		Access security		
13	Format coherence with consumer needs		Consistency			Consistency	
14	All aspects of entity are available	**Resolution**	Completeness		Completeness	Completeness	
15	Believe in the correctness of context			**Correctness**		*Reliability* (1, 2, 3, 4)	
16	Validity of context depending on freshness	**Up to dateness** (6)	**Up to dateness** (6)	*Freshness* (6)	**Up to dateness** (5, 6)	*Timeliness* (5, 6)	

Name	Criterion (name + meaning) only defined by one author	Meaning	Meaning defined by all authors
Name	Name defined by different authors for different meanings	*Name*	Name only defined by one author
Name	Name defined by different authors for the same meaning	Name (X)	The definition of this criterion depends on the X criterion
✓	Criterion not defined by author but another criterion depends on it		

Table 2.2: Comparison of different lists of QoC criteria [Marie et al., 2013b]

2.4 Trust

On the Internet in general, trust is an essential factor in the success of websites, systems and services. If an end-user feels confident in a system/service or in the intermediary through which he/she reaches that system/service, this indicates that it is likely that this person agrees to disclose personal or private information such as medical data, credit cards numbers. He/she will feel protected and knows that on the other side, everything is prepared to prevent any leak of information that he/she does not want to reveal.

Taking the example of the Amazon.com web site, though which most people buy to sellers they do not know. However, given the general trust in that website, they do these purchases relying on the assumption that Amazon.com will protect their personal data and will take appropriate measures if any problem happens.

In the IoT, trust must be even more present in systems and services. If a large number of objects around us are collecting information about everyone and people do not trust that system or service, this may cause feelings such as, persecution, being watched, privation of freedom. Such lack of trust may cause the repudiation of these sensors and systems by users.

2.4.1 Trust Definition

Different authors have defined Trust considering all the perspectives required to obtain a faithful computational trust value. The three most representative definitions which complement each other are [Gambetta, 2000], [McKnight and Chervany, 1996], [Grandison and Sloman, 2000]. Combining these definitions, we propose the following concept of trust:

Trust is a subjective probability by which an individual, A (trustor), expects or believes that another individual, B (trustee), performs a given action dependably, securely, and reliably in a given situation within a specified context with a feeling of relative security, even though negative consequences are possible.

Trust is a key underlying element of any transactional activity. It characterizes the "bond" and "comfort" that the transacting parties share amongst themselves and impacts the utility of their mutual activities. In pervasive applications, transactional activities will typically involve the exchange of information between parties.

[INCOME, 2013a] presents trust as the cornerstone between privacy and QoC. It concludes that, both QoC and privacy are related to the notion of trust, but from two different perspectives.

- From the *context producers point of view*, “priority must be given to the respect of the privacy of the context owners. If a context consumer is not trustworthy, a strict privacy is desired.”
- From the *context consumer point of view*, “a context information of high quality is expected to ease the decision making process from the point of view of the context consumer.”

Trust is critical in the large-scale, open distributed pervasive systems considered here, enabling interactions between parties in uncertain and constantly changing environments. The verb "believe" in the definition above will allow us in the future to exploit both probabilistic and logic-based techniques.

2.4.2 A Trust Meta-model

[Neisse, 2012] fomalizes trust as the measurement of the belief from a trusting party point of view (trustor) with respect to a trusted party (trustee) focused on a specific aspect that possibly implies a benefit or a risk. And, the term trustworthiness refers to the amount, measurement, or degree of trust in a trust relationship.[Neisse, 2012] did not find in the literature an appropriate trust management model that combines multiple trust aspects as required by the context-aware computing domain. All other existing solutions refer to a specific domain and are not easily portable to another domain.

Figure 2.4 depicts the trust meta model proposed by [Neisse, 2012] showing how the concept of "Trust Belief" is related to the social trust concepts of "System Trust", "Dispositional Trust" and "Situational Trust"

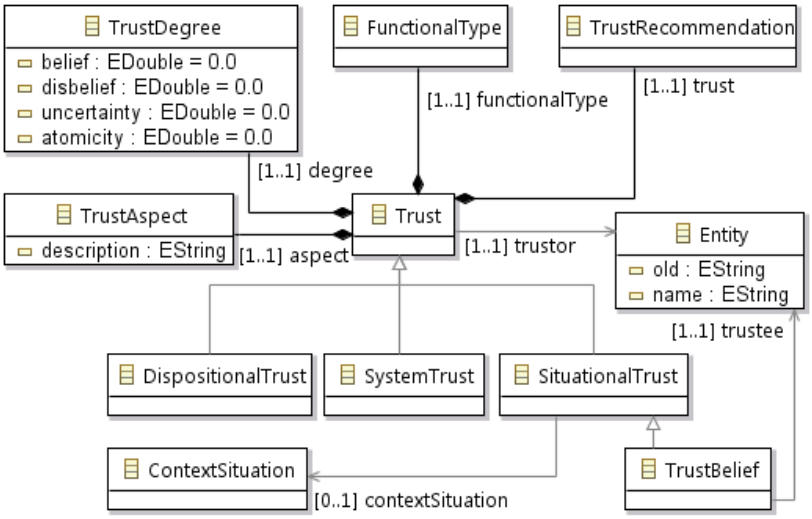


Figure 2.4: Trust Type Model [Neisse, 2012]

The concept of "Dispositional trust" is the intrinsic/inherited disposition of an entity to trust any other given entity in the absence of evidence or previous experiences. The concept of "System Trust" is the impersonal trust perception an entity has regarding the set of regulations and safeguards of the system as a whole.

The advantages that make us consider this trust model as part of our solution are the following: *(i)* It is extensible. It was designed to support *(ii)* context-aware services users and consumer services focusing on trust aspects related to identity provisioning, *(iii)* privacy enforcement, *(iv)* context information provisioning and *(v)* context-aware service provisioning. Therefore, we decide to integrate this trust model as part of our solution to model contracts to define the agreements between producers and consumers.

2.5 Privacy and how to Protect it

Privacy is one of the fundamentals pillars of this thesis, that is why instead of beginning by defining what is privacy, we decide to start by clarifying what it is not. After this clarification we present the definition we propose.

2.5.1 What Privacy Is Not

It is common to think of privacy when we learn about leaks of information, unauthorized modification of data, theft of credit card numbers or the violation of secure access to computers. But what is the border between privacy and security? Roger Clarke [Clarke, 2013] brings out a common misuse of the Term 'Privacy'. The term "privacy" is often employed by people to refer to the security of data against various risks, such as the risks of data being accessed, modified by unauthorized persons or more restrictively, to refer only to the security of data during transmission. These aspects are only a small fraction of the considerations within the field of "information privacy", which is the interest an individual has in controlling, or at least significantly influencing, the handling of data about themselves [Clarke, 2013]. More appropriate terms to use for those concepts are 'data security' and 'data transmission security'.

The term "confidentiality" is also sometimes used by computer scientists to refer to "data transmission security", risking confusion with obligations under the law of confidence.

2.5.2 Privacy Definition

Privacy *is the interest that individuals have in sustaining a "personal space", free from interference by other people and organizations* [Clarke, 2013].

The concept of privacy is profoundly embedded into our civilizations, is recognized in all legislations of civilized countries and the concerns about its protection have proven to be a significant barrier against the diffusion of the technologies involved in the IoT [Violino, 2003].

Roger Clarke [Clarke, 2013] has outlined a number of common privacy consideration that have gained wide acceptance, this list states the complexity of the privacy issue. They are as follows:

- Privacy of the person
- Privacy of personal behavior
- Privacy of personal communications
- Privacy of personal data
- Privacy of personal experience

Nevertheless, as can be observed in Figure 2.5, many of these aspects were ephemeral and none of them generated records. But on the other hand, the IoT will retain a massive consolidation of individuals' personal data, experiences and behaviors rendering them available for exploitation.

Privacy Protection *is a process of finding appropriate balances between privacy and multiple competing interests* [Clarke, 2013].

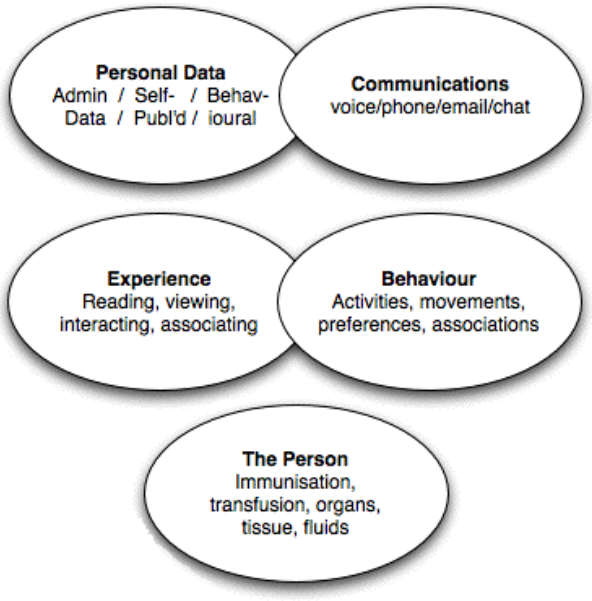


Figure 2.5: Privacy Multiple Dimensions Diagram by [Clarke, 2013]

[Agre and Rotenberg, 1998] define privacy as the power someone has over a piece of information that belongs to him or her. [Skinner et al., 2006a] propose a definition of Ideal Privacy that is aligned with the ambitions of our work "With Ideal Privacy, users at all times determine when, how and what personal information is revealed. Additionally, personal data owners decide to what extent others can utilize their information once access is granted. Ideal Privacy gives users complete control over their personal data and more generally all of their information privacy concerns."

"Privacy in general is very subjective and means different things to different people. Common among all interpretations is the perspective that it is a human right but is context and environmentally dependent" [Dey et al., 2001].

Concretely, we define privacy as:

The capacity of control about what, how, when, where and with whom to share information.

Broadly, when people think of "the Internet", they visualize going online by using a computer, or mobile device, such as tablets or smartphone. In traditional Internet users play an active role. Although, Internet users recognize the privacy problems. Every day, they give away personal information about themselves, often without even realizing it. In many instances, is impossible to control how this information is used by others.

In IoT scenarios, privacy problems arise even for people who are not using any IoT service. Numerous everyday devices can also access the Internet and transmit various types of data. In fact, almost any item (even an article of clothing with a special tag [Violino, 2003]) can be connected to the Internet.

The privacy danger is the vast amounts of data that these smart objects continuously broadcast over

the Internet [Clearinghouse, 2013]. In effect, the Internet is rapidly evolving into an "always on" tool of surveillance. Some examples of these smart objects include:

- Medical devices that continuously monitor blood pressure and other vital signs
- Home security devices (including surveillance cameras)
- Thermostat controls
- Wearable devices with the ability to record events
- Smart grid appliances, which are services that rely on access information about a consumer's use of energy.
- Motor vehicle "black boxes"
- Athletic shoes and Fitness Band Trackers that monitor your exercise routine.

While these smart devices have the potential to help consumers save time and energy and improve their health and safety, the benefits come with significant privacy and security risks. The data collection and potential sharing of that data with others from these devices are a significant concern. It's important to realize that even tiny amounts of collected data can, in the aggregate, reveal a great deal about our personal lives. For example, medical information could be mined and used inappropriately, pricing might be adjusted in a discriminatory manner, and advertisements might be targeted based upon analysis of collected data.

People concerns about privacy are indeed well justified. In fact, the ways in which data collection, mining, and provisioning will be accomplished in the IoT are completely different from those that we now know and there will be an amazing number of occasions for personal data to be collected. Therefore, for individuals it will be impossible to personally control the disclosure of their personal information.

Accordingly, privacy should be protected by ensuring that individuals can control which of their personal data is being collected, who is collecting such data, and when this is happening. Furthermore, the personal data collected should be used only in the aim of supporting authorized services by authorized service providers; and, finally, the above data should be stored only until it is strictly needed.

To handle the data collection process, appropriate solutions are needed in all the different subsystems interacting with human beings in the IoT.

Without an effective regulation and control of the IoT, a significant number of individuals are going to suffer privacy violations which can imply lost job opportunities, ruined reputations, discrimination, etc.

2.5.3 Privacy Taxonomy Background

Privacy has been defined legally and legislatively in particular by the U.S. Privacy Act of 1974 and by the European Union [OECD, 1980, EU, 1995, EU, 2002]. Greenleaf [Greenleaf, 2012] has isolated ten global elements that are common to international directives for the protection of privacy and that are now universally accepted as part of a full data privacy law. These ten privacy concerns are given in

Table 2.3. Each privacy concern may be preserved through a protection dimension. For identifying the relevant protection dimensions, we follow the taxonomy proposed by Barker et al. [Barker et al., 2009]. This taxonomy describes how to handle data privacy in practice in the domain of database systems. It includes four privacy dimensions in which data repository privacy can be achieved that are the following:

1. *Purpose*: Defines for what goal the data is used;
2. *Visibility*: Indicates who is authorized to access data;
3. *Retention*: Specifies how long the data is retained;
4. *Granularity*: Determines the level of detail at which the data is delivered.

For more details about these dimensions, the reader can refer to [Ghazinour et al., 2009a].

The privacy concerns as stated by [Greenleaf, 2012] and the protection dimensions that can enforce them are summarized in Table 2.3.

Data Privacy Law Concern [Greenleaf, 2012]	**Privacy Dimension [Barker et al., 2009]**
Collection: limited, lawful and by fair means; with consent or knowledge	Granularity (QoC)
Data quality: relevant, accurate, up-to-date	Granularity (QoC)
Purpose specification at time of collection	Purpose, Retention
Notice of purpose and rights at time of collection	Visibility, Purpose
Limited use (including disclosure) to specified or compatible purposes	Visibility, Purpose
Security through reasonable safeguards	Visibility, Retention
Openness of personal data practices	Visibility
Access: individual right of access	Visibility
Correction: individual right of correction	Visibility, Granularity (QoC)
Accountable: data controllers accountable for implementation	Visibility, Purpose, Granularity, Retention

Table 2.3: Global privacy concerns and their associated protection dimensions

We have correlated these ten precepts with the dimensions suggested by Barker et al. [Barker et al., 2009] (See Table 2.3). We concluded that each privacy concern is preserved through one or more privacy dimensions. This confirms the validity of Barker's approach. For context data, we extend the granularity dimension to the more general QoC concept which may actually subsume it. The granularity is considered as one QoC criterion among others.

Figure 2.6 shows examples of the purpose, visibility and granularity dimensions proposed in [Barker et al., 2009]. On each axis, some levels are defined. A specific privacy policy may be associated to each level or to each combination with one level taken from each axis.

The four protection dimensions of Barker's taxonomy are described below in more detail and we relate them to context data.

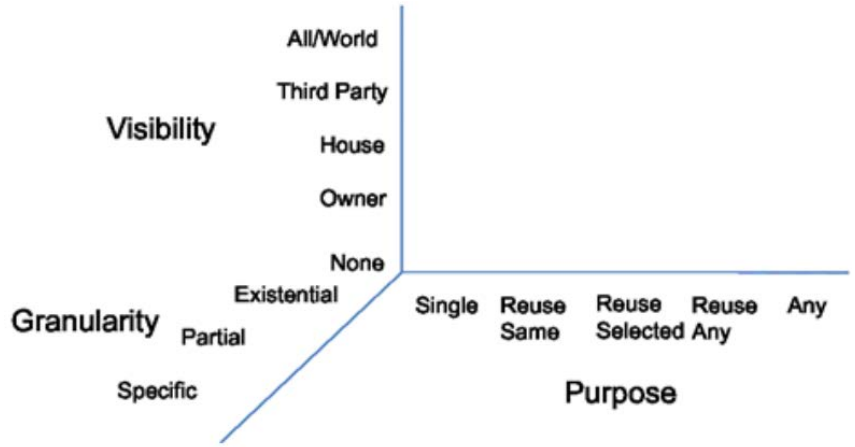


Figure 2.6: Example of Data Privacy dimension levels [Barker et al., 2009]

2.5.3.1 Purpose

Purpose is a fundamental privacy principle ensuring that context consumers declare their legitimate intentions on their use of context data. A privacy-aware system should therefore explicitly record and track the purpose for which a data item is collected and should then verify that the given reasons match the declared purpose. For example, a patient provides a healthcare provider with personal information in order to receive appropriate medical care. Thus, the patient accepts to release sensitive information but only for a very specific purpose [Ghazinour et al., 2009a].

Purposes may be organized into purpose categories. As shown on the purpose axis in Figure 2.6, purpose categories Single means that information can only be used for one specific task and at one given time. Purpose Reuse same indicates that information may be reused but only for the same task. Reuse Selected is intended to capture the case where the data collected can be used for purposes related to the primary purpose for which the data was provided. Reuse Any data is used for un-foreseeable related purposes such as a research study or to monitor efficiency. Finally, purpose category Any is the least privacy protective category and probably should be discouraged as a matter of normal data collection. However, there are scenarios that could be envisioned where the data is either of such low value or it already exists in the public domain, where this might be desirable.

2.5.3.2 Visibility

For the visibility concern, the context owner defines who is allowed to access or use the context data. For instance, patients may want their medical information to be visible to their doctor (for treatment purposes), but not to a third-party such as a potential future insurance company. In other words, visibility constrains the set of consumers who can access context data with respect to an operation and a purpose [Ghazinour et al., 2009a].

In Figure 2.6, the visibility scale is divided into four levels in addition to the None case where the

data is not visible at all. Concerning visibility Owner, [Barker et al., 2009] argues that the ownership of data should be retained by the original provider. With respect to context data, visibility Owner indicates that only the context owner has access to the information. Visibility House was introduced by [Barker et al., 2009] to indicate that the data repository housing the data can view the data provided. In our case related to context data, visibility House would correspond to a visibility restricted to the context manager (which may use the data for transformation purpose). The Third party level offers a special visibility over the information to a third party which is required to conform to some kind of an agreement before access is granted. In our case, we define several circles of visibility for third parties (see section 5.4.2). Finally, visibility All/World states that the data are provided to everyone.

2.5.3.3 Retention

Privacy principles require that data that have been collected for a particular purpose should be removed after they have been used for this intended purpose. They can also be removed after a given delay. Retention models may specify an expiry condition (expressed on time, period, number of accesses, etc) after which the data should not be accessible even for authorized users and purposes. A privacy model must therefore explicitly indicates for how long collected data can be used and under which circumstances. Data that have passed their retention limit must be deleted by the context manager and by the context consumers.

A long period of retention may be required when working with historical information for calculating tendencies over time for instance. But this kind of history management can infringe on the privacy of the individuals. Therefore, it is important to consider their agreement to use their information for a limited period of time during which the application can perform the necessary operations. In general, a retention period is proposed by context consumers, which may then be accepted, refused, or modified by context owners.

2.5.3.4 Granularity

With the granularity dimension, context owners may specify how detailed is the provided information will be. For example, a medical care-giver could require very specific information about a particular patient's condition, while an insurance company may only need a summary statement of the costs of the patient's care [Barker et al., 2009].

On Figure 2.6, the first level of the granularity scale, named Existential, corresponds to the case where privacy could be compromised by undertaking a set of existence tests. Knowing the existence of data may already impact privacy. The Partial level indicates that only partial information is revealed. This implies to use some techniques including obfuscation, aggregation or summarization to alter the information in some non-destructive way. For instance, the granularity specifies whether an exact age is shown or only an age range, such as child, teenager, or adult [Ghazinour et al., 2009a]. Finally, the lower privacy protection, called Specific, occurs when specific data are released. This is the most common scenario where the data are operated in an unobfuscated way.

2.5.4 Privacy Protection Criteria

More and more people do care about their privacy and the risk of using a service or system which can reveal part of their personal information. This situation alarms people very much and might create a rejection of these technologies. That is why, it is fundamental to find ways to ensure privacy and generate a safe feeling. Especially, for applications or systems that people are not using directly but which are collecting information about everyone around. Given the importance of preserving privacy, several proposals have been developed throughout the different topics of privacy.

In order to provide a solution to model the privacy, we are interested in dealing with existing work proposing lists of principles or techniques to protect privacy. We first present the work of some relevant authors who have proposed principles or techniques to preserve privacy. Then, we analyze the meaning, differences and contrast among these principles and techniques and their applicability to the IoT. The next section then shows the results of our comparative study. Finally, we elaborate a list of privacy protection criteria applicable to the IoT with the aim to check whether existing solutions already apply these principles and techniques.

Agrawal, 2002

In 2002, [Agrawal et al., 2002] proposed what they call Hippocratic Databases Systems, standing that the databases should include privacy as a central concern. They enunciate the first list of ten founding principles to protect privacy: “Purpose specification”, “Consent”, “Limited collection”, “Limited use”, “Limited disclosure”, “Limited retention”, “Safety”, “Accuracy”, “Openness”, “Compliance”.

These principles are based on the current privacy legislation and guidelines around the world. Countries such as Canada, United States, France, Australia, and Japan have their own list of privacy principles, but in general they remain very close. Some of them are exhaustive and cover several kinds of subjects as companies are more focused on very sensitive data like medical records. The legislation in this matter has evolved allowing the establishment of legal codes that ensure the privacy protection in any field.

Even if the scope of [Agrawal et al., 2002] research is different from ours, the objective of identifying the key components in privacy protection remains the same. Based on the fact that the IoT allows the collection, storage and access to a large amount of personal information, we can consider the IoT as a huge database. It indeed presents two properties considered as essential for databases by [Ullman, 1988]: first, the ability to manage the persistence of collected data and second, the ability to access this huge amount of data. In conclusion, these ten principles are applicable to the IoT with some constraints that we discuss later in Section 2.5.5.

Skinner, 2006

[Skinner, 2006] proposed in 2006 a new list of principles to protect privacy. This list was constructed based on the Hippocratic Database principles proposed by [Agrawal et al., 2002]. In the same spirit, but in the context of developing information systems, [Skinner, 2006] affirms that, “...like the Hippocratic Database principles that govern the design and implementation of the databases ... the rest of the systems should be designed through the guidance of the Hippocratic policies”. Skinner describes seven Hippocratic policy principles: “Anonymity”, “Limited collection and use”, “Limited disclosure and retention”,

"Security and sensitive information", "Openness, access, and integrity", "Third party and transborder uses", "Identifiers".

Roger Clarke, 2006

In 2006, [Clarke, 2006] provides definitions of privacy and related terms. His approach is more general and is not oriented to information systems or databases. He does not propose a list of principles, instead he discusses some actions affecting privacy from the point of view of the identification of individuals. These actions are: "Authentication", "Anonymity", "Identification", "Pseudonymity".

INCOME Project

The INCOME project has the ambition to provide multi-scale context management solutions for the IoT. [INCOME, 2013a] establishes a relationship among "access control", "trust", and "QoC" to protect privacy.

The privacy concern then becomes a matter of trust for deciding whom to share data with. Thus, people will have to control the access to their information in a complex and moving environment.

Additionnaly, QoC can be seen as a means to protect the user's privacy through the use of obfuscation techniques associated to the level of QoC of the context information provided.

2.5.5 Principles and Actions

The table 2.4 groups the privacy principles and actions exposed previously and shows the convergence among all these actions and principles. The first row shows the domain for which each list was created. Then, the next rows associate the principles and actions by their affinities to solve similar problems. Finally, last row shows if the list corresponds to a principle or an action to protect privacy.

This section discusses the different principles and actions regarding the IoT constraints studied in this thesis.

2.5.5.1 Consent

[Agrawal et al., 2002] establishes the principle of "consent", in which users must agree with the purpose for which their personal information is captured. However, the consent cannot be archived in the IoT. Information is captured by everyday objects without people realizing it and in many cases they are unable to stop it. That is why a different paradigm to design IoT applications that protect the privacy of people is required. Anonymity, access control, obfuscation and trust control are strategies that combined altogether allow some privacy protection.

2.5.5.2 Access Control and Authentication vs Anonymity

[Skinner, 2006] considers that everyone should have an identifier conferred by a responsible entity and should never be revealed. However, this goes against anonymity and pseudonymity, which means that,

Domain →	**General Domain**	**Database Systems**	**Information Systems**	
Authors → **↓ Issue**	[Clarke, 2013]	[Agrawal et al., 2002]	[Skinner, 2006]	[INCOME, 2013a]
Privacy	Authentication, Anonymity, Identification, Pseudonymity	1. Purpose Specification 2. Consent 3. Limited collection 4. Limited use 5. Limited disclosure 6. Limit Retention	1. Anonymity 2. Identifiers 3. Limited collection and use 4. Limited disclosure and retention	Access Control
Security		7. Safety	5. Security and sensitive information 6. Third party and transborder uses	Trust management
Openness		8. Openness	7. Openness,	
QoS		9. Accuracy	access, and integrity	QoC (Obfuscation)
Compliance		10. Compliance		
	Privacy Principles			**Privacy Actions**

Table 2.4: Privacy Protection Actions Comparison by Author

whenever possible, personal information collected or stored should not be associated directly or indirectly to any particular individual [Clarke, 2013].

There are many reasons why a person wants to remain anonymous. For example, to escape from some responsibilities, to avoid physical harm, to avoid unwanted and unjustified public exposure, or to keep personal data out of the hands of intrusive marketers and governments.

Whenever and wherever possible the collection of personal information in the IoT should be done in a way that supports anonymity and pseudonymity of active or passive owners. Although, in many cases the anonymity is possible, there are some others (such as medical information) in which it is impossible to apply this protection technique. That is why, another mechanism is required to protect privacy.

[Clarke, 2013] defines authentication as, "*the process whereby a degree of confidence is established about the truth of an assertion*". Thanks to the authentication process, access control can be archived, where users can setup their privacy preferences, limit the access to their personal information, and establish obligations to use their personal data. Thus, authentication can increase the trust among the participants of IoT.

The INCOME project [INCOME, 2013a] has analyzed different forms of access control. Concluding that authorization systems should provide both a very expressive policy language and an adaptable enforcement architecture such as XACML [OASIS, 2012, Cheaito et al., 2011] or P3P [W3C, 2011]. However, the complexity of the system to control combined to the complexity of the language makes this solution not conceivable. An authorization policy language should be more human readable, and control systems should also consider the following constraints:

- Users cannot spend too much time configuring their devices before using it.
- Privacy mechanisms should rather consider users' preferences to limit interactions with end-users.

Therefore, it is required to develop automatic mechanisms to translate human readable privacy requirements into access policies.

2.5.5.3 Limiting Collection, Purpose, Use, Retention, and Disclosure

[Agrawal et al., 2002] and Skinner [Skinner, 2006] agree that the key to success for protecting privacy is to limit the actions that threaten privacy. This requires to first limit the collection of personal information to the minimum necessary for accomplishing a specified purpose.Personal information shall then be retained only as long as necessary for the fulfillment of the purpose. Finally, personal information shall not be communicated outside of the system to third partieswithout the consent of its owner.

To achieve all these principles, the use of access control policies that are able to compose all these elements is required. As it is shown here, attribute-based access control policies and systems fit perfectly to this aim. However, simplifying the writing of this policies is still an open issue.

2.5.5.4 Security and Sensitive Information

For [Skinner, 2006], 'The information system must take all reasonable steps to protect the personal information it holds from misuse and loss and from unauthorized access, modification or disclosure." This thesis does not deal with the security aspects but we recognize its importance and agree that sensitive information must always be protected by 'stronger' security safeguards. However, it is a separate component that can be connected independently.

As we made clear at the beginning in Section 2.5.1, one should not confuse the term privacy with data security. The privacy protection does not only depend on security safeguards, it also depends on what usage is made of that information.

2.5.5.5 "Openness, Accuracy and Integrity" vs. Obfuscation

[Agrawal et al., 2002] propose the "Openness and Accuracy" principles to protect privacy. [Skinner, 2006] adds the principle of "Integrity". The principle of "openness" says that, an individual must be able to retrieve, modify or delete the information about himself easily. "Accuracy" means that stored information should be accurate and up-to-date. And the "integrity" objective is to ensure data are recorded exactly as intended and, upon later retrieval, to ensure the data are the same as it were when originally recorded. In short, data integrity aims to prevent unintentional changes to information.

The "openness" property allows individuals to erase the traces of their activities or to change some facts that may jeopardize their privacy. Indeed, incorrect information may cause serious perils like attempts against the security and privacy of individuals.

However, individuals have the right of modifying their information and render it less accurate to protect their privacy. If an individual is unable to make these modifications, then the information system must take all reasonable steps to ensure the integrity or obfuscation of the personal information it stores.

The IoT should allow users to protect their privacy by reducing the accuracy of their information automatically. The principle of "Openness" as described by Agrawal corresponds more to direct user interaction through graphical interfaces. Our solution pretends to describe the parameter of this reduction of accuracy by modifying the QoC of collected context information.

In computing, obfuscation refers to the process by which information is deliberately altered to influence consciously the set of inferences that could be made to protect sensitive information while allowing users to still derive (hopefully) value from the information they receive [Bisdikian et al., 2012]. The obfuscation serves as the mechanism for protecting against inappropriate use of shared information.

To conclude, authentication and access control play an important role in the implementation of these principles, providing safeguards against theft and other misappropriations.

2.5.6 Proposed Privacy Protection Criteria List

We compose a new list of criteria combining the principles proposed by [Agrawal et al., 2002] and [Skinner, 2006] and the mechanisms to protect privacy proposed by the INCOME project [INCOME, 2013a]. Here is the list of the privacy protection criteria that will be discussed in more details in chapter 3:

- Context-Aware Collection Control
- Purpose Access Control
- Retention Access Control
- Disclosure Access Control
- QoC (Obfuscation)
- Trust management

2.6 The Notion of Contract

[Meyer, 1992] proposed the term “Design by Contract” (DbC) as an approach for designing software. Meyer explains with a clear metaphor of the conditions and obligations of business contracts, that contracts enable developers to specify constraints to be satisfied by context, so that, the application behaves as expected. However a contract here is quite different from the concept of a software contract.

Meyer’s metaphor inspires us to follow a similar approach in which two parties (producers and consumers) expect some benefits from the contract and are prepared to incur some obligations to obtain them. Concluding that, what is an obligation for one party is usually a benefit for the other.

These benefits and obligations should be documented to protect both sides. As a result, the producer is entitled to receive a specific privacy protection. The consumer must receive a particular level of QoC. And both producer and consumer are not liable for failing to carry out tasks outside of the specified scope.

However, to force some obligations to consumers or producers on the IoT is difficult. Therefore, it is preferrable to use the term "Guarantees", that is “a promise that something will be done or will happen”[Press, 2012]. This concept relies on the trust between the parties to ensure compliance with the responsibilities acquired. Additionally, in this thesis, the expected benefits are expressed as requirements concerning the privacy protection for the producer and a proper level of QoC for the consumer.

This symmetry between guarantees and requirements is suitable for IoT agreement solutions, as both producers and consumers have requirements and should fulfill certain guarantees if they want that their needs are met. The contracts will describe the expected behavior of each IoT participant and based on those expectations an agreement will be established to create a relationship among them.

Meyer also stands that, if the contract is exhaustive, every "obligation" (in our case guarantee) entry also in a certain sense describes a "benefit" (in our case requirement) by stating that the constraints given are the only relevant ones. In this thesis, these constraints correspond to the clauses of the contract specifying the context situations in which the requirement or guarantee will be applied.

2.6.1 Contract Definition

Acoording to The Cambridge Dictionary [Press, 2012], a contract is "a legal document that states and explains a formal agreement between two different people or groups, or the agreement itself".

An interpretation of this definition can be:

The written form of such an agreement that may be enforceable by law between two or more parties for doing or not doing something specified.

In this thesis, contracts allow that two strangers (producer and consumer) offer agreements between each other. A standard support should therefore clearly specify the requirements and guarantees to which each party will be submitted. For this reason, it is important to know what are the components of real life contracts to design such a standard support.

The elements of a contract according to U.S. Legal Inc. [U.S. Legal Inc., 2013] are: "offer" and "acceptance" by "competent party" having capacity to exchange "guarantees" to create "mutuality of obligation". An offer is a promise to act or refrain from acting, which is made in exchange of a return promise to do the same. The acceptance of an offer is the expression of the assent to its terms. A guarantee is the exchange of values in which each contract party must provide something of value that induces the other to enter the agreement. And, with the mutuality of obligation, both parties are bound to perform their obligations otherwise the agreement will be canceled.

The U.S. Legal Inc. [U.S. Legal Inc., 2013] concept is the starting point to create the proposed model solution of this thesis.

2.6.2 Contract Formalization

As previously mentioned, the requirements and guarantees should be documented to protect both sides. In order to formalize the contracts, we have chosen to follow a model-driven approach. The Model-driven Architecture (MDA) is a software development methodology which focuses on creating and exploiting domain models. Models can be processed in many ways. They can be validated, transformed, translated into code, or interpreted [Völter, 2009]. With these models, we can create dynamic contracts which play a significant role in the automation and the adaptation process of context applications on the IoT considering privacy and QoC concerns at all stages of the system lifecycle.

MDA is based on a four-layer meta-modeling architecture and several complementary OMG standards, as we can see in Figure 2.7. These standards are Meta-Object Facility (MOF) [OMG, 2011a], Unified Modeling Language (UML) [OMG, 2005] and XML Meta-data Interchange (XMI) [OMG, 2011b].

The layers are: meta-meta-model layer (M3), meta-model layer (M2), model layer (M1), and the real world or instance layer (M0). The purpose of these four layers with a common meta-meta-model is to support multiple meta-models and models and their scaling – to enable their extensibility, integration and generic model and meta-model management.

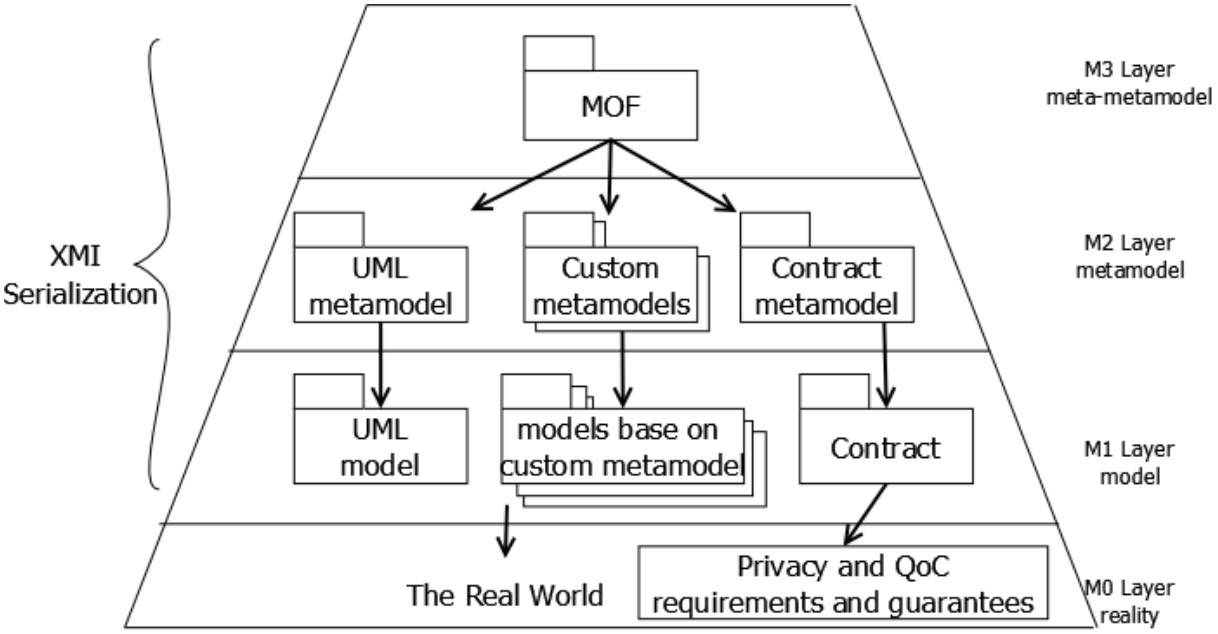


Figure 2.7: Meta-model Layers

The M3 layer defines an abstract language and framework for specifying, constructing and managing meta-models. It is the foundation for defining any modeling language, such as UML or even MOF itself. MOF also defines a framework for implementing repositories that store meta-data (e.g. models) described by meta-models [OMG, 2011a]. Layer M2 manages all the meta-models defined by MOF. One of these is UML, a graphical modeling language for specifying, visualizing and documenting software systems. With UML profiles, we can extend basic UML concepts (Class, Association, etc.) with new concepts (by the use of stereotypes) and adapt them to specific modeling needs. The M1 layer contains the models of the real world, represented by concepts defined in the corresponding meta-model at the M2 layer (e.g. UML meta-model). Finally, at the M0 layer are things from the real world. Newer MDA approaches consider that models are "snapshots" of reality. Whereas, the "traditional" MDA approach sees the real-world things as instances of model elements [Atkinson and Kühne, 2003].

The last basic standard of this architecture is XMI, which defines mapping from MOF-defined meta-models to XML documents and schemes. XML, which has good software tools support, enables meta-meta-model, meta-model and model sharing through XMI.

Applying MDA approach to the solution, the M2 layer corresponds to the contract meta-model, which defines elements of the contract, for example, depending on the point of view, requirements and guarantees on privacy and QoC. At this level is defined a specific language to describe the privacy and

QoC, and how are they related. The contracts are at the M1 layer, as instances of the contract meta-model. A model at this level will apply to a specific domain solution, for example, to define the levels of QoC required for a medical context application and the actions taken to guarantee not to disclose the collected information to any unauthorized third party. Finally, the materialization of M0 is represented by the translation of the contract terms, for example into access policies, with the enforcement of the context application configuration, even to set up the behavior of some sensors.

2.7 Privacy Policy Languages

On the Internet, a privacy policy of a service provider condenses the usage of collected personal user data [Kosta and Dumortier, 2008]. For the IoT, a privacy policy should compile the management of collected context data not only for the context consumer side, but also the actions of the context producer side. This section focuses on suitable languages and frameworks for the definition of privacy and QoC policies in order to define context contracts.

There are three features that a policy language must satisfy to set privacy policies on dynamic environments as is the IoT. Firstly, a language should provide a high degree of *functionality*, in order to cover all aspects regarding the IoT, like taking into account the dynamism of context information, negotiation of the privacy and QoC levels and trust among the actors. Secondly, the language should be *expressive*, which guarantees the definition of all mandatory parts of privacy and QoC into a policy. And the final criterion is the provision of *policy enforcement* on both the producer and consumer sides.

The next sections discuss existing policy languages: P3P, EPAL, SAML, XACML, in order to decide which of these languages is a good candidate for our solution implementation. The language selection is based on the three properties mentioned in the previous paragraph. However, it does not mean that no other language is capable of performing the same functions. We are looking for a language associated with a framework, which should allow us to make an implementation with a minimum of modifications.

2.7.1 P3P - Platform for Privacy Preferences

The P3P current release 1.1 was published by the W3C Working Group in 2006 and designed to give users more control on their personal information when browsing the Web. P3P is a standard machine-readable language allowing websites to declare their intended use of the information they collect about web browser users.

P3P works in the following way: An internet user defines locally a set of privacy preferences, then when this user visits a web site, a dedicated privacy agent tool retrieves from the Web Server the P3P policy and compares it with the user's pre-defined privacy preferences. If the two do not match, the privacy agent will inform what information the server wants to get and ask if he/she is willing to provide to the site, and risk giving up more personal information. As an example, a user may store in the browser preferences what information about their browsing habits should not be collected. If the policy of a Website states that a cookie is used for this purpose, the browser automatically rejects the cookie.

Focusing on functionality, the P3P standard is designed to communicate privacy practices to the user. P3P offers web users to either accept or reject a published privacy policy. It does not consider any context

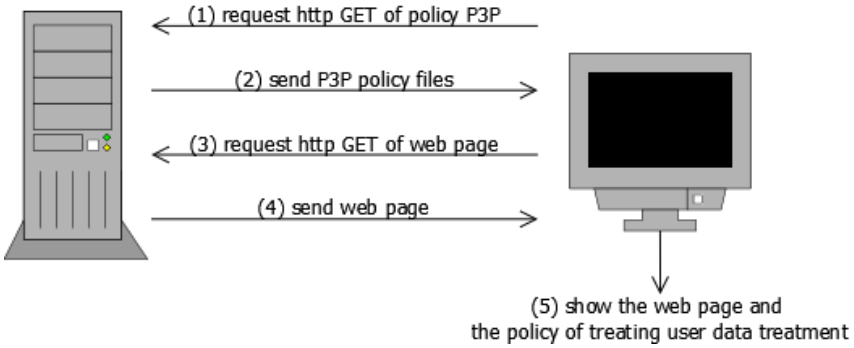


Figure 2.8: HTTP transaction with P3P

information to suggest or adapt a decision. The standard does not offer the possibility to negotiate privacy policies with service providers.

In terms of expressiveness, the main content of a privacy policy is the following: Which information the server stores, such as, IP address, email address, name. The use of the collected information, for example, regular navigation, tracking, personalization, telemarketing. Who will receive this information, for example, only the current company, third party. The permanence and visibility, e.i., for how long information is stored; whether and how the user can access the stored information. In general, it seems that P3P offers a vocabulary capable of defining any privacy practices of service providers. However, aspects like QoC and trust are not taken into account.

Even though not claimed by the standard, the non-enforcement of privacy policies represents a further weakness of P3P. While a published P3P policy asserts privacy practices of the service provider, users are not able to control the usage of their personal data [Sackmann et al., 2006]. This lacking enforcement requires users to trust the accuracy and truth of the service provider's privacy statements.

2.7.2 EPAL - Enterprise Privacy Authorization Language

IBM designed the Enterprise Privacy Authorization Language (EPAL) in 2003 as an interoperability language for exchanging privacy policy in a structured format between applications or enterprises. EPAL [IBM, 2003] is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems according to fine-grained positive and negative authorization rights. An EPAL policy categorizes the data an enterprise holds and the rules which govern the usage of data of each category. Since EPAL is designed to capture privacy policies in many areas of responsibility, the language cannot predefine the elements of a privacy policy. Therefore, EPAL provides a mechanism for defining the elements which are then used to build the policy.

An EPAL policy defines lists of hierarchies of data-categories, user-categories, and purposes, and sets of (privacy) actions, obligations, and conditions. User-categories are the entities (users/groups) that use collected data (e.g., travel expense department or tax auditor). Data-categories define different categories of collected data that are handled differently from a privacy perspective (e.g., medical-record vs. contact-data). Purposes model the intended service for which data is used (e.g., processing a travel expense

reimbursement or auditing purposes). These elements are then used to formulate privacy authorization rules that allow or deny actions on data-categories by user-categories for certain purposes under certain conditions while mandating certain obligations. In order to allow for general rules and exceptions, EPAL rules are sorted by descending precedence. E.g., a rule about a particular employee can be inserted before the rule about the department in order to implement an exception.

In general, the main objectives of this language are to: enable organizations to be demonstrably compliant with their stated policies; reduce overhead and the cost of configuring and enforcing data handling policies; and leverage existing standards and technologies. Therefore this language is mostly used for internal purposes and it is more fine-grained than the web privacy policy languages.

With regard to expressiveness, EPAL offers all necessary policy elements to define internal privacy policies. As the specification was designed for the internal use in enterprises, it does not offer an element for the definition of disclosures to third parties. Furthermore, the lack of rule-combining algorithms complicates the definition of a foreseeable evaluation behavior.

2.7.3 SAML - Security Assertion Markup Language

SAML (Security Assertion Markup Language) is an Extensible Markup Language (XML) standard that allows a user to log on once for affiliated but separate Web sites. SAML 2.0 became an OASIS Standard in March 2005. SAML specifies three components: assertions, protocol, and binding, and three assertions: authentication, attribute, and authorization. An authentication assertion validates the user's identity. Attribute assertion contains specific information about the user. And an authorization assertion identifies what the user is entitled to do. Figure 2.9 shows the main concepts of SAML.

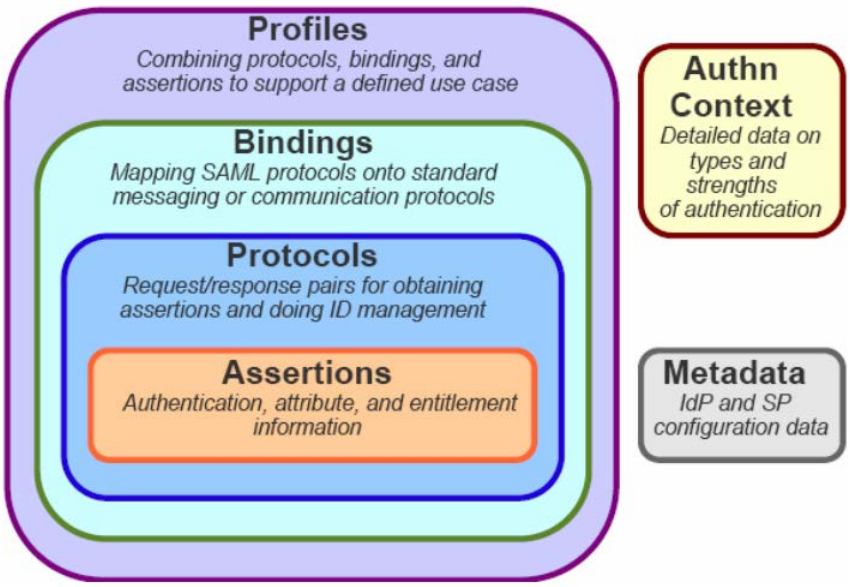


Figure 2.9: The relationship between basic SAML Concepts [Ragouzis et al., 2008]

The SAML specification defines three roles: the principal (typically a user), the identity provider (IdP), and the service provider (SP). In the use case addressed by SAML, the principal requests a service from the service provider. The service provider requests and obtains an identity assertion from the identity provider. On the basis of this assertion, the service provider can make an access control decision.

The protocol defines how SAML asks for and receives assertions. Binding defines how SAML message exchanges are mapped to Simple Object Access Protocol (SOAP) exchanges.

SAML includes the ability to make statements about the attributes and authorizations of authenticated entities. There are many common situations in which the information carried in these statements is something that one or more of the parties to a communication would desire to keep access to a set of entities as restricted as possible. Statements of medical or financial attributes are simple examples of such cases.

Parties making statements, issuing assertions, conveying assertions, and consuming assertions must be aware of these potential privacy concerns and should attempt to address them in their implementations of SAML-aware systems [Standards, 2005].

Focusing on functionality, SAML is designed for business-to-business (B2B) and business-to-consumer (B2C) transactions. In terms of expressiveness, it does not define any of the attributes required to describe privacy. SAML 2.0 provides a number of extensibility points by which new requirements, unforeseen at original drafting, can be accommodated in an interoperable manner. Finally, no support was provided for conveying obligation in SAML, however, with the addition of XACMLAuthzDecisionStatementTypes in v2.0, it becomes possible to forward the entire XACML response, including obligations, to the Policy Enforcement Point (PEP).

2.7.4 XACML - eXtensible Access Control Markup Language

The latest 3.0 version of XACML (eXtensible Access Control Markup Language) was standardized in January 2013 [XACMLv3, 2013]. The standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate access requests according to the rules defined in policies. The XACML standard defines the format of the policies, the request and the response between the PEP and the PDP. As the default representation of XACML is XML and is backed by a schema, the request and response are typically expressed as XML elements or documents. XACML is rising in popularity and acceptance. There are many new APIs to handle this technology that make it easy to use it.

An XACML request contains information necessary to take authorization decision. Basically it contains attribute names and attributes values. When evaluating the policy, attribute names and attribute values are compared according to criteria defined in the policy [XACMLv3, 2013]. Figure 2.10 depicts the general structure of the XACML language.

Focusing on functionality, the architecture proposed by XACML includes a Policy Information Point (PIP), which is an entity that acts as a source of attribute values (e.g. a resource, subject, environment). This feature provides the dynamism required by the IoT to assess the environment at runtime in order to determine whether the policy is applicable and to take an access decision. The PIP can also be used to search for information on the trust levels among IoT actors. In terms of expressiveness, the fact that XACML can be extended by adding new Categories, Attributes, Function, Obligation, Policy Combin-

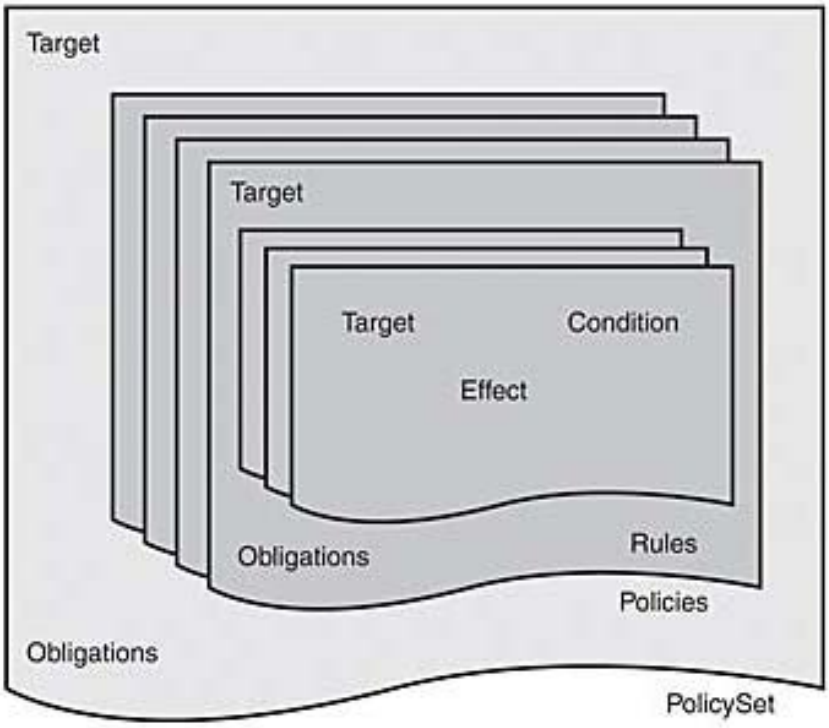


Figure 2.10: Core XACML constructs and their interrelationships. [Rosenberg and Remy, 2004]

ing Algorithms, allows one to implement any model to describe the privacy and QoC, and any context information attribute to make the evaluations. Finally, the enforcement of obligations is included in both the language and architecture of XACML, which is suitable for the enforcement of actions to protect the privacy or to guarantee a correct QoC level.

The Appendix E section presents a brief introduction of the main elements of XACML.

2.7.5 Conclusion

Among the studied languages, only one of them can easily provide without adaptation the three aspects of functionality, expressiveness and policy enforcement that we have identified as necessary.

The P3P language is neither functional nor allows policy enforcement. EPAL and SAML languages combines the expressiveness and enforcement aspects but do not bring the functionality aspect. And finally, the closest model to our needs is the XACML language and framework. Table 2.5 summarizes the conclusions of our study of these languages.

As a consequence, we chose the XACML [XACMLv3, 2013] authorization solution for the following reasons:

It is a sophisticated access control language aimed at both users and enterprises [Kumaraguru et al., 2007]. It provides fine-grained authorization with a high level of abstraction by means of policies and rules. It

language → ↓ feature	**P3P** [W3C, 2011]	**XACML** [XACMLv3, 2013]	**EPAL** [IBM, 2003]	**SAML** [Ragouzis et al., 2008]
functionality		✓		
expressiveness	✓	✓	✓	✓
enforcement		✓	✓	✓

Table 2.5: Privacy Policy Languages Comparison

supports Attribute-Based Access Control (ABAC), and dynamic evaluation of policies can be done with context data retrieved from the environments by using Policy Information Points (PIP).

XACML reference architecture externalizes the authorization system as a service in the infrastructure. It enables placing the authorization service as a new layer, which means that authorization algorithms can be removed from the application logic. Finally, XACML is a standard ratified by the OASIS standards organization.

The XACML architecture promotes a loose coupling between the component that enforce decisions, the policy enforcement point (PEP), and the component that takes decisions based on XACML policies, the policy decision point (PDP).

2.8 Conclusion

This chapter defines the terms of IoT, QoC, Trust, Privacy, and the notion of contract. We identify that the subject of this thesis lies within the "semantic-oriented" IoT vision. We present and decide to integrate, as part of our solution for modeling contracts, the QoCIM meta-model and the trust meta-model proposed by [Marie et al., 2013b] and [Neisse, 2012] respectively. We present differing recognitions and interpretations of privacy, and we follow the taxonomy proposed by [Barker et al., 2009] to identify the dimensions that describe privacy (Purpose, Visibility, Retention, Granularity) appropriately for the IoT. We also compare different principles and actions for protecting privacy proposed by several authors to obtain our list of protection criteria that will use to verify the conformity of the related works. We then present a model-driven approach to formalize the proposed contracts. We continue with the analysis and evaluation of the capabilities of several privacy policy languages concluding that the eXtensible Access Control Markup Language (XACML) is a feasible alternative to implement the model of contracts. This analysis includes the reasons why we have chosen XACML to represent context contracts, as well as, the main concepts of this technology. In the next chapter, we present some related works, and we analyze them according to the privacy protection criteria proposed here.

Chapter 3

State of the Art of Models and Frameworks for Privacy and QoC Management in the IoT

Contents

3.1 Introduction

One of our objectives is to provide a generic and expressive modeling solution for expressing the requirements of active or passive participants of the IoT concerning privacy and QoC. The passive participants are those users who are not registered in the application, however information about them is captured by sensor. Our approach follows the MDE (Model Driven Engineering) principles to create contracts that not only use the existent models or taxonomies to describe privacy but also, to define rules to use QoC to protect privacy. Our model also should guarantee that context consumer applications receive context information with the appropriate QoC level they need.

We decided to study several existing models and frameworks that have proposed mechanisms for privacy protection, even if they address only partly our objective, in order to identify any possibility of reusing some of the concepts or modeling patterns they propose. Additionally, we are interested to know

which of the proposed privacy protection criteria (Section 2.5.6) were implemented for these solutions. Our interest is therefore to establish the differences between our work and the existent solutions by identifying whether they treated the problems of the decoupling of participants, trust management and how they handled dynamic events imposed by pervasive environments such as the IoT. In the next sections, we discuss the main works related with our research.

3.2 Obligation of Trust Protocol (OoT Protocol)

[Mbanaso et al., 2007, Mbanaso et al., 2009] introduce a new protocol called Obligation of Trust (OoT). This protocol allows to express requirements and capabilities in terms of privacy between a user and a remote Service Provider (SP) using XACML (eXtensible Access Control Markup Language).

According to them, privacy must be considered from both points of view, SPs and users, considering that the interaction and interchange of information come from both parts. It is well known that users normally wants preserve privacy, but even the SPs have the necessity of protecting it as well, taking into account the sensitive information they handle. In fact, they expose that if SPs can ensure privacy, users will trust more in them and will accept to share information. [Mbanaso et al., 2007, Mbanaso et al., 2009] propose that "this can be achieved with a balanced approach based on a negotiation between the two parties on their requirements and willingness to accept constraints imposed by the other side".

In order to be able to establish this negotiation, each party must specify its constraints on the other party (requirements) and what it is willing to do for the other party if all the requirements are fulfilled. In OoT, these two sets are expressed using the WS-XACML (Web Services Profile of XACML) Assertion Type (Figure 3.1) and are described using P3P policy contents. Each XACMLPrivacyAssertion is included in an SAML (Security Assertion Markup Language) assertion that provides a validity period, the signature of the asserter and forms a difficult to repudiate contract. Figures 3.2 and 3.3 show examples of XACMLPrivacy assertions for the SP and the user side.

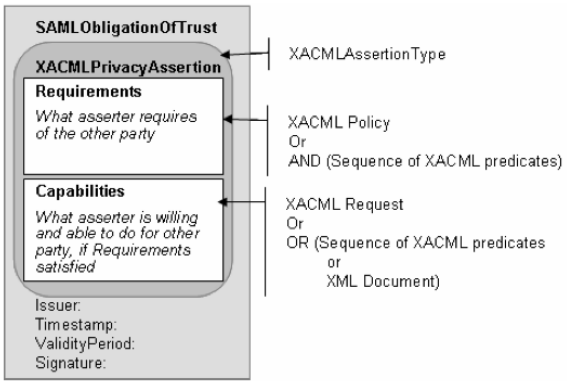


Figure 3.1: SAML Obligation of Trust Model [Mbanaso et al., 2009]

The OoT protocol offers interaction between parties to allow them to communicate what they can offer and what their constraints are. It is symmetric and has two kinds of messages that can be exchanged:

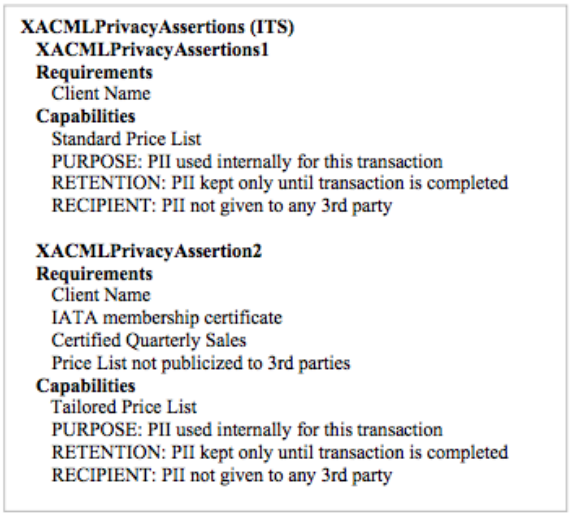


Figure 3.2: Example of a Service Provider (ITS) XACMLPrivacyAssertion [Mbanaso et al., 2009]

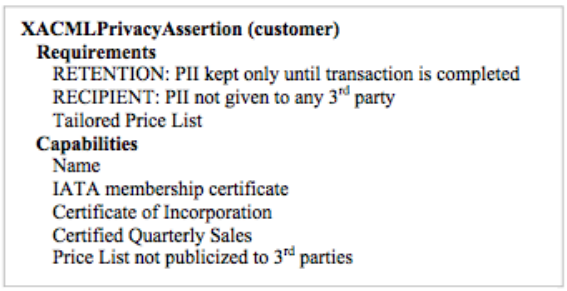


Figure 3.3: Example of a Customer XACMLPrivacyAssertion [Mbanaso et al., 2009]

Notification of Obligation (NOB) and Signed Acceptance of Obligation (SAO). When a party wants to request information or data from another party, it starts by sending a NOB containing constraints concerning what is wanted (requirements) and also what it is willing to offer (capabilities). The other party evaluates this NOB. If they can offer what is required and if they are satisfied by the capabilities, they send back a SAO. If they cannot offer some of what is required or if they have requirements that are not listed in the requestor capabilities, they send another NOB with what they can offer and their capabilities. This negotiation continues until a SAO is sent and accepted or if one party terminates the session (Figure 3.4).

When the negotiation between the two parties is over, *i.e.* one party can fulfill all the requirements received in the last NOB and is satisfied with all capabilities or a subset of them, this party sends back a SAO in which the requirements are a subset or all of the capabilities of the last NOB and the capabilities are equivalent to the requirements of the last NOB. The party who receives a SAO has only two options, to accept it and send back its own SAO (reversing requirements and capabilities) or to terminate the

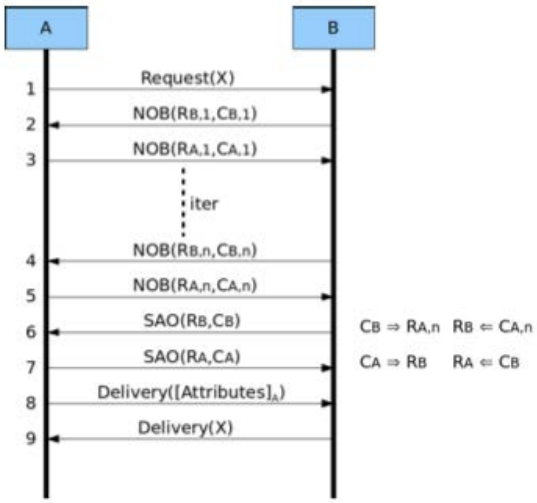


Figure 3.4: The OoT Protocol Sketch [Mbanaso et al., 2009]

session; one can not send back a NOB after receiving a SAO.

The scope of this research belongs to the web services domain. Its main issue is to provide privacy in distributed transactions where technically "difficult-to-repudiate" services are vital. [Mbanaso et al., 2009] describe one concrete approach to enhancing privacy assurance, by permitting the bilateral exchange of privacy requirements and the capabilities to satisfy them. The approach has not been proven on the IoT, which presents a set of different challenges. For example, the authors have recognized a limitation that affects directly the decoupling participants problem. This framework assumes that both parties exist as legal entities that can be used if violations occur. Requiring either a robust PKI (Public Key Infrastructure) system to exist or some other TTP (Trusted Third Party) mechanism to establish whether the subject of a certificate is a legal entity, and will put meaningful identifying information in the issued certificate.

[Mbanaso et al., 2007] have described how XACML can be used to address negotiations involving the gradual and incremental exchange of information in such a way that the level of trust that is established can determine what other information (policy/attributes) is released at any phase. This finding encourages us to use the XACML technology for the development of our solution.

3.3 PrimeLife policy engine (PPL Engine)

The privacy policy language (PPL) [PrimeLife, 2010] has been designed by the European project PrimeLife to protect privacy in emerging Internet applications such as collaborative scenarios and virtual communities. The scenario that has been considered is a user that provides personal information to an online shop; this information can be reused by a third company for advertising (cf. Figure 3.5). A Data Subject wants to access a resource hosted by a Data Controller, but has to reveal some personal data in order to access the resource. Furthermore, the Data Controller may want to further forward the Data Subject's personal data to a Downstream Data Controller. On one hand, PPL allows the Data Controller to express

which personal data it needs from the Data Subject and how it will treat this data. On the other hand, PPL allows the Data Subject to express to whom she is willing to release her personal data and how she wants her data to be treated.

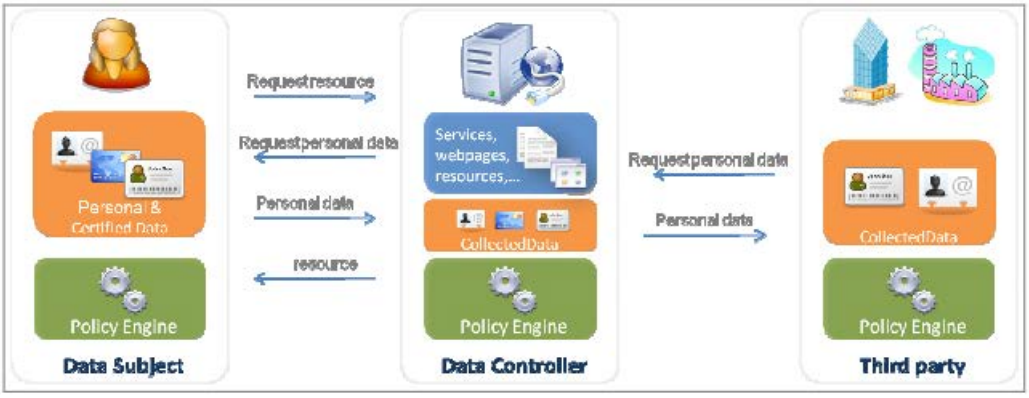


Figure 3.5: PPL model of interaction [PrimeLife, 2010]

PPL extends XACML [XACMLv3, 2013] in order to specify privacy requirements and obligations. The goal of the language is to be used by:

- the Data Controller to specify:
  - access restrictions to resource it offers,
  - how "implicitly" collected information (e.g log information such as IP address, connection time, etc) are treated,
- the Data Subject to specify:
  - access restrictions to personal information and how this information can be treated by the Data Controller,
  - how the Data Subject wants the implicit information to be treated.

The PPL schema reuses the traditional XACML entities (Target and Condition) and extends classes PolicySet, Policy and Rule to implement privacy requirements (Cf Figure 3.6). The CredentialRequirements describe the credentials that need to be presented in order to be granted access to a resource. The ProvisionalActions class is used to specify provisional actions that the requestor must perform before being granted access to a resource (e.g revealing attributes to the Data Controller or to a 3rd party). The DataHandlingPolicy class indicates what happens to the information about the Data Subject that is collected during an access request. The ProvisionalActions may contain an optional reference to the applicable DataHandlingPolicy. For example, if the provisional action consists in revealing a particular attribute, an associated data handling policy can specify how this attribute will be used. The DataHandlingPreferences state how the information obtained from the resource protected by the Data Subject

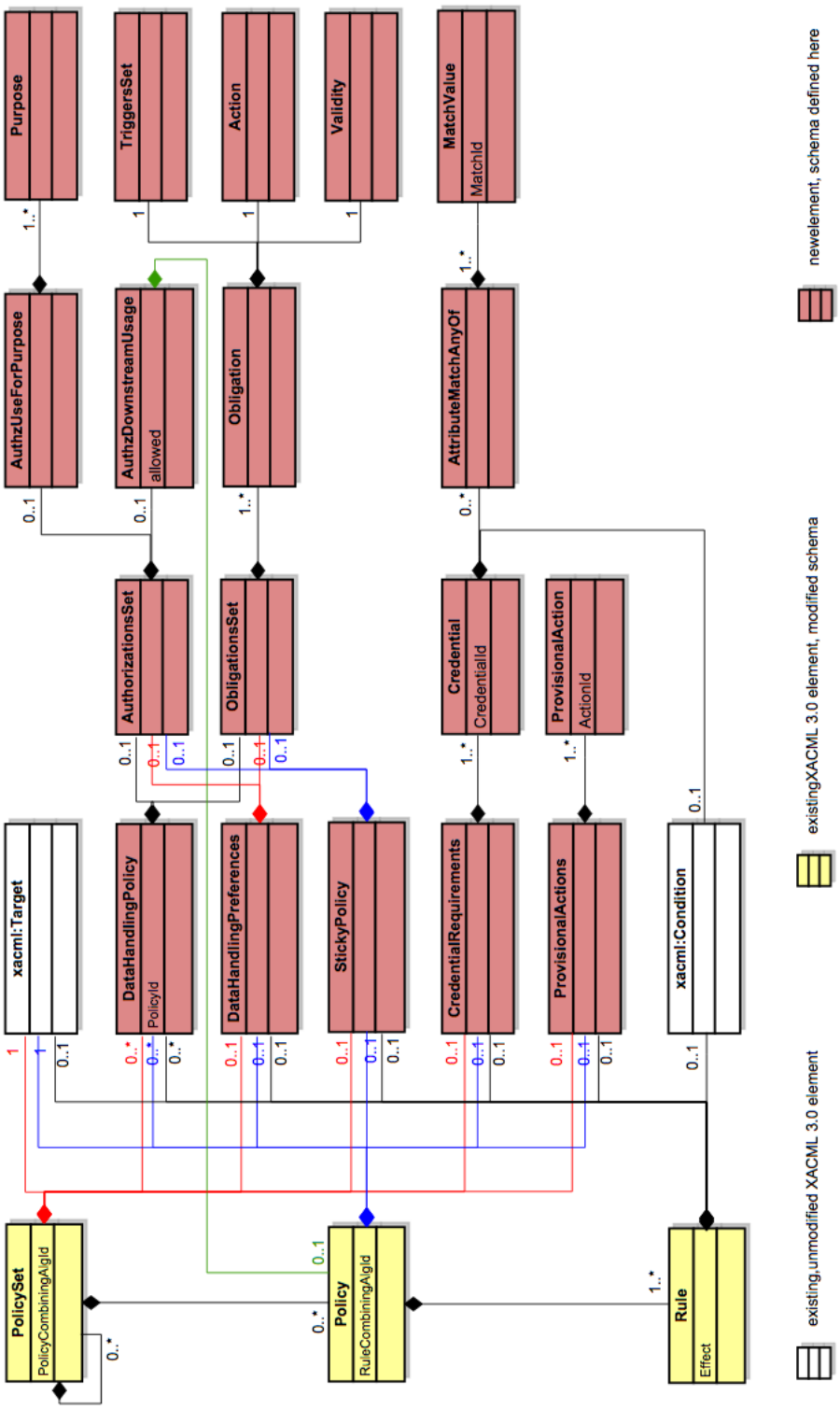


Figure 3.6: PPL Schema [PrimeLife, 2010]

is to be treated after access is granted. Preferences are expressed by means of a set of Authorizations and Obligations, just like DataHandlingPolicy. An important difference between DataHandlingPreferences and DataHandlingPolicy is the resource that they pertain to: DataHandlingPreferences describe how the resource protected by the Data Subject itself has to be treated after being collected; while a DataHandlingPolicy defines the information that a requester has to reveal in order to be granted the access to the resource. The StickyPolicy class is associated to a resource, corresponding to the agreed-upon sets of granted authorizations and promised obligations with respect to a resource. The StickyPolicy is usually the result of an automated matching procedure between the Data Subject's DataHandlingPreference and the Data Controller's DataHandlingPolicy. The data handling policies, preferences, and sticky policies contain a set of Obligations. An Obligation is defined as: "A promise made by a Data Controller to a Data Subject in relation to the handling of her Personally Identifiable Information (PII). The Data Controller is expected to fulfill the promise by executing and/or preventing a specific action after a particular event, e.g time, and optionally under certain conditions" [PrimeLife, 2010]. In addition to obligations that specify actions that the Data Controller is required to perform on the transmitted information, authorizations specify actions that it is allowed to perform. It is a generic, user-extensible structure for authorizations so that new, possibly industry-specific authorization vocabularies can be added later on. Two extensions of a basic authorization vocabulary for using data for certain purposes and for downstream access control (about forwarding the information to third parties) are specified.

The matching process in PPL [PrimeLife, 2009] consists in comparing the Data Controller's DataHandlingPolicy with the Data Subject's DataHandlingPreferences. The DataHandlingPolicy must be less permissive than the DataHandlingPreferences. Intuitively, this means that the DataHandlingPolicy provides less authorization than the DataHandlingPreferences and the DataHandlingPolicy defines more obligations than the DataHandlingPreferences. The matching can be done either by the Data Subject or by the Data Controller. The same mechanism is used when Data Controller shares personal data with third parties. However, the process does not describe any negotiation process. If the matching process fails, there is no mechanisms for reducing privacy requirements.

3.4 The User-Centric Privacy Framework (UCPF)

Smart Home environments are typically equipped with different kinds of sensors and tracking devices for context-aware service provisioning. The User-centric Privacy Framework aims at providing a generic framework for developing privacy aware systems [Bagüés et al., 2010]. The Sentry module, which is the core element of the framework, has been designed to be an integral part of the user's home environment to become seamlessly embedded into the Smart Home software infrastructure, and to act as a privacy proxy for a tracked individual.

The framework distinguishes four roles: i) target is the tracked individual, ii) the context provider acts as an intermediate entity responsible for collecting, caching, managing and disseminating context information, iii) the context-aware mobile service (CAMS) is the context aware-application (it compiles context information for its users), iv) and the recipient is the user of the service. The sentry is a privacy proxy between the context provider and the CAMS that controls accesses to privacy-relevant data of its user.

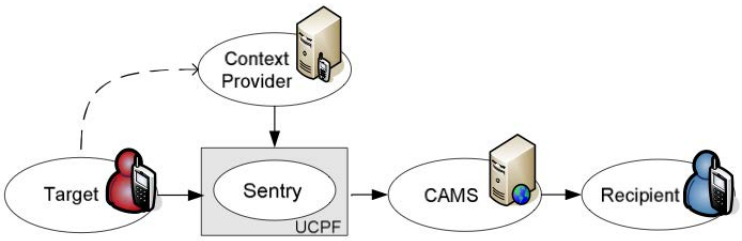


Figure 3.7: UCPF model of interaction [Bagüés et al., 2010]

The Sentry language is built on top of the Web ontology and the semantic web rules language [Bagüés et al., 2007]. A policy in the Sentry language (Figure 3.8) represents the privacy criteria of an entity identified by property onTarget. The language distinguishes two subclasses PersonalPolicy and OrganizationalPolicy. Organizational policies focus on the need of organizations to manage specified context information of a group of predefined individuals (e.g employees, clients). Personal policies allow individuals to control the use of their personal information. A policy is composed of a set of rules that consists in a condition (based on properties onSubject, onService, onAction and onResource) and an effect (property hasEffet). An effect can be a Negative Authorization Rule (NAR) to deny the disclosure of the requested data or a Positive Authorization Rule (PAR) to allow it. A PAR is one out of four positive permissions: the Positive Authorization Permission, the Positive Transformation Permission, the Positive Obligation Permission or the Positive Transformation Obligation Permission. Obligations are used to control the information disclosed among users of a particular CAMS, authorized purposes, number of accesses and retention time of data. Transformation is defined "as any process that the tracked user may define on a specific piece of context information to limit the maximum accuracy to be revealed, e.g coordinates accuracy max 500m or filter calendar items labelled as private" [Bagüés et al., 2007].

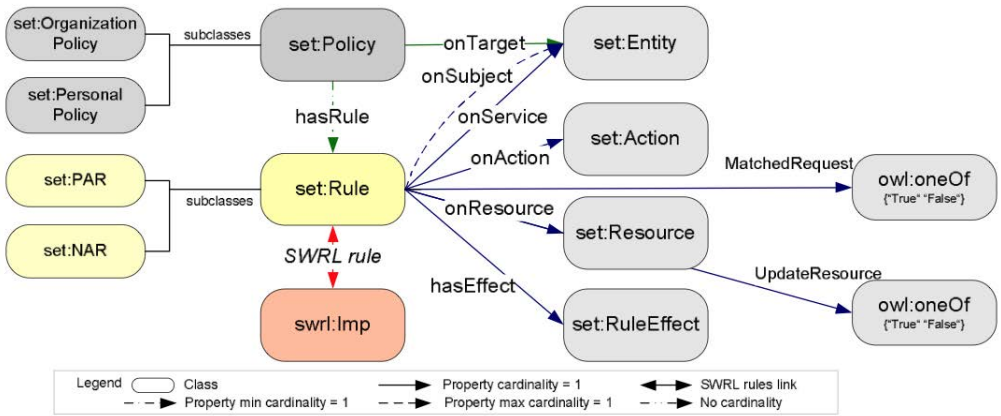


Figure 3.8: UCPF policy language [Bagüés et al., 2010]

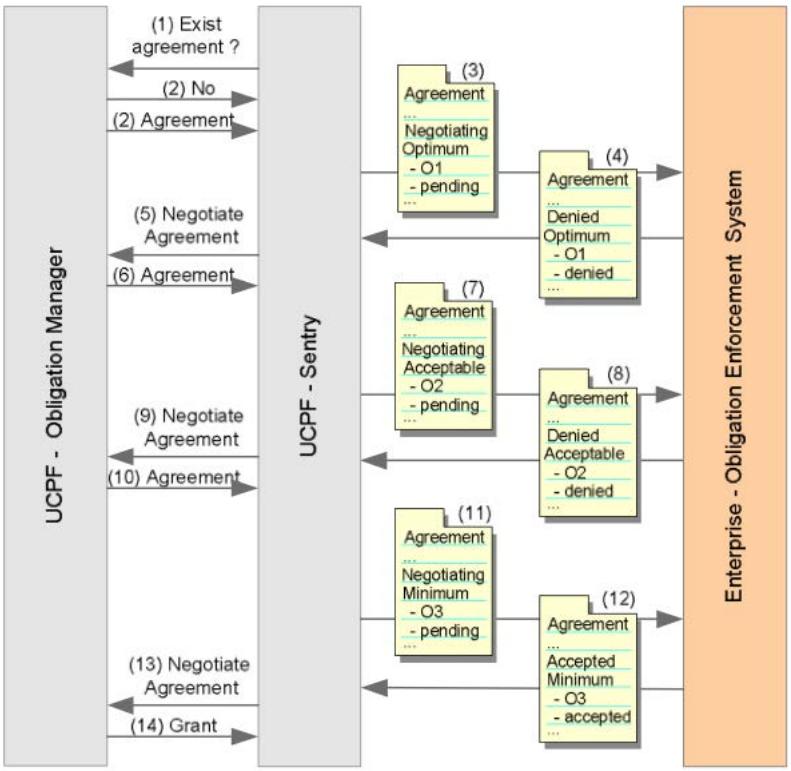


Figure 3.9: UCPF Agreement negotiation [Bagüés et al., 2010]

Sentry aims at allowing non-expert users to negotiate the desired level of privacy instead of making a "take or leave" approach [Bagüés et al., 2010]. Then, when a policy evaluation of a service request concludes on an obligation effect, Sentry queries the obligation manager module of the UCPF for an agreement over the pending obligations, step 1 in Figure 3.9.

If an agreement has been already traded, the obligation manager returns it to Sentry. If there is not a previous agreement, the obligation manager returns the agreement to be negotiated, step 2. For each negotiable obligation, the target can specify three levels corresponding to what is optimal, acceptable and minimal. For example, a negotiable obligation about data retention can be data must not be stored for the optimal level, delete data after 20 days for the acceptable level and delete data after 60 days for the minimal level. These three obligation levels are used by Sentry to launch the negotiation, steps 3 to 14. Sentry proposes each obligation level to the organization that owns the CAMS and is responsible to enforce obligation. If a specific level is rejected by the organization of the CAMS, Sentry proposes the next lower level until an agreement is reached or the negotiation fails after three rounds.

3.5 Family of Context-Based Access Control Models

[Filho, 2010] has proposed a family of reference models called CxtBAC for Context-Based Access Control, which takes into account the requirements of pervasive environments. Filho does not impose a particular implementation mechanism. His aim is to propose an access control model that mimics the natural way of access authorization in the physical world. He argues that: "Like in real life, in spontaneous pervasive environments the short-lived interactions between people and resources often occur in a dynamic, distributed, and transparent manner." Thus, less intrusive and more flexible access control solutions are required.

From his point of view, "smart usage of context information provides a powerful approach for controlling access to resources that, in many situations, is more suitable than conventional identity-based or role-based access control solutions."

In CxtBAC, the access control policies are entirely based and dependent on context information. CxtBAC augments or replaces traditional user attributes such as username/password for the purpose of authentication and access control by making them less intrusive and adaptable to situational or contextual changes. For instance, instead of assigning permissions directly to the users/roles, resource owners/administrators may define for each protected resource the context conditions that enable someone to access it. Therefore, when a request on a protected resource is made, the access control mechanism should identify the current context in order to enforce the associated set of access control policies.

CxtBAC can handle subject-based access control if needed, considering user identities and roles as specific types of context information. CxtBAC includes capabilities to establish relationships between context situations and permissions, as well as between users and context situation." Moreover, CxtBAC supports two well-known security principles: *least privilege* ("Least privilege is supported because access control system implementing CxtBAC should be configured for granting only the set of permissions required for the current situation of users" [Filho, 2010]) and *data abstraction* ("Data abstraction is supported by means of abstract permission, such as a shared multimedia object (e.g., photo, video) rather than read, write, execute, and delete permission typically provided by operating systems" [Filho, 2010]).

This model family (see Figure 3.10) consists in a core model CxtBAC_0, which defines the basic entities of their models and extensions with extra features such as context hierarchy (CxtBAC_1), constraints on context information (CxtBAC_2), constraints on QoC (Q-CxtBAC) or constraints for social relationships among users (S-CxtBAC) or constraints for privacy purpose (P-CxtBAC).

- Q-CxtBAC - Quality-Aware CxtBAC:

  According to Filho, QoC-awareness can reduce the probability of making a faulty decision. If the context information is incorrect or inaccurate, context-based access decisions made using this information might grant permission to unauthorized users resulting in security gaps. Therefore, Q-CxtBAC extends the $CxtBAC_3$ model by adding a set of QoC constraints that determine whether or not QoC associated with context and access context are acceptable. Therefore, only acceptable values will be permitted.

- S-CxtBAC - Social-Aware CxtBAC:

  Filho proposes the S-CxtBAC (Social-Aware CxtBAC), which extends CxtBAC_3, in order to support the definition of social relationships among users. For example, we can classify the known

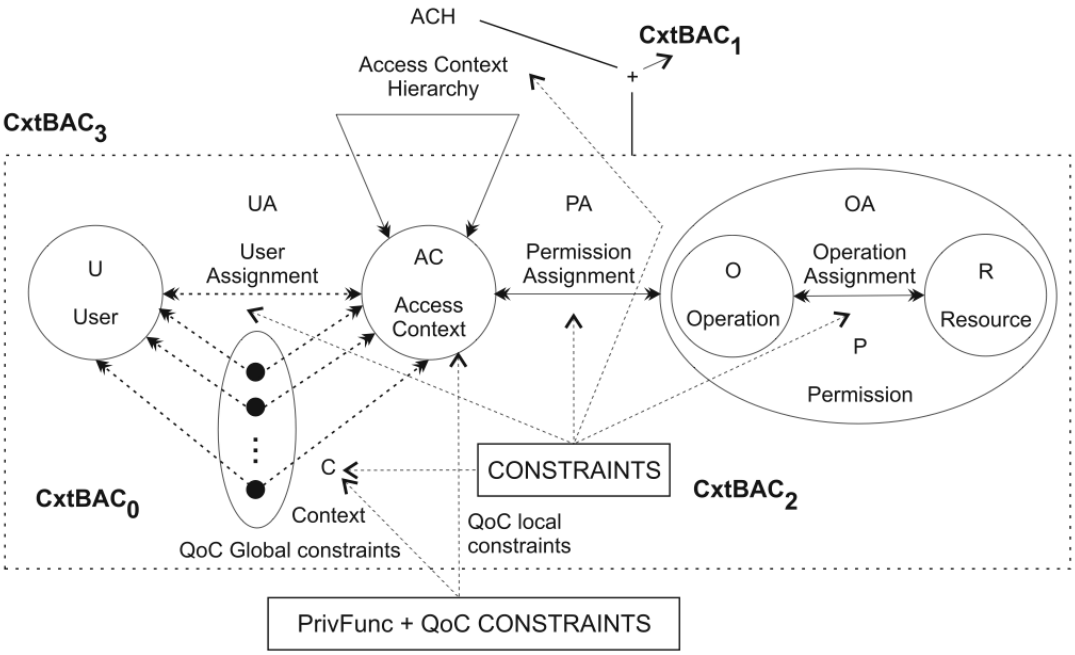


Figure 3.10: The family of Context-Based Access Control Models proposed by [Filho, 2010]

persons as friend, family, best friend, coworker, etc. This social classification can be used to grant different set of permission. Social relationships can be combined with contextual information for improving the context-based access control policies. In this case, the support to social relationships for defining access policies strengthens the constraints associated with access contexts.

- P-CxtBAC - Privacy-Aware CxtBAC:

  Regarding privacy, Filho argues that, "Context information can describe private information about the current situation of users". In this case, users might not want to disclose this information used for making access control decisions. Moreover, if users disclose his/her context information for a CxtBAC-based system, they might want that the enforcement of policies made using his/her context information preserves his/her privacy. The main idea behind this model is to increase the confidence of users in the system. Therefore, he/she describes P-CxtBAC that extends from the CxtBAC_3 model by defining functional aspects with regard to the support to privacy requirements of users, such as anonymous enforcement of access control policies. P-CxtBAC supports the following functions defined on context and access context entities: anonymity, selection, and obfuscation. The "selection function" indicates that systems should use only the set of context information necessary for making access control decisions.

- QP-CxtBAC - Quality and Privacy-Aware CxtBAC:

  Finally, The QP-CxtBAC is a model derived from the union of Q-CxtBAC and P-CxtBAC. Therefore, QP-CxtBAC enforces access control policies taking into account the quality and privacy

requirements of context information.

The management of access control systems that implements the CxtBAC core model (CxtBAC_3), consist of performing the following set of activities:

1. Identifying the context information that the system is able to gather for characterizing the situation;
2. Defining access context and its hierarchies;
3. Defining the constraints;
4. Defining context-based access context policies;
5. Reviewing policies, access context, access context hierarchies and constraints during the entire life cycle of the system.

If an access control system uses as basis S-CxtBAC, P-CxtBAC, Q-CxtBAC, and PQ-CxtBAC, other administration operations should be performed by the user/administrator. Such operations are described in the following:

S-CxtBAC: users should annotate the persons in their social network in order to be able of defining social-aware context-based access control policies;

P-CxtBAC: users should define policies with regard to their privacy requirements for protecting their context information. Moreover, the user/administrator should configure the PrivFunc to be executed on context, access context, and policies;

Q-CxtBAC : user/administrator should define QoC global threshold and QoC local threshold on context and access context, respectively. When defining a policy, user/administrator is able to use QoC local threshold for verifying the QoC information;

QP-CxtBAC: It should perform the operations described previously for the P-CxtBAC and Q-CxtBAC, following this order.

The solution proposed by Filho provides an alternative way to set access permissions to resources taking into account context information for the process of decision making for access control.

When Filho refers to the privacy concern when the access control system is using context information for making access decision. He provides some privacy functions, such as anonymization and obfuscation to protect the privacy avoiding any disclosure during the access decision process. However, he does not take into account the fact that the resource could also be context information. Hence, his model does not provide a way to force obligations such as purpose, retention, disclosure, use, that must be applied over the resource by the context consumer.

The first difference between Filho and our work is that the trust is considered as a QoC criterion, i.e., it only considers the point of view of the quality expected by the access control system to take accurate access decisions.

The second difference is that CxtBAC does not propose a solution for the decoupling of the participants, it assumes that all users and resources are registered in advance. Thus, participants are known beforehand. In CxtBAC, resources correspond to equipments or services. However, in the IoT, persons'

context information represents the resources, services or equipments.In addition, it is impossible to know in advance with what objects of the IoT a person (the resource) will interact.

The last identified difference is that QoC is used only to make access decisions. The author does not consider the QoC of the resource if the resource is context information. According to Filho ,"it is very important for ensuring the correctness of access decisions to take into account also the QoC information used for enforcing context-based access control policies."

3.6 SensorSafe and Obfuscation Frameworks

Chakraborty et al. [Chakraborty et al., 2011, Choi et al., 2011, Chakraborty et al., 2012] address the concept of behavioral privacy as opposed to traditional identity privacy and propose solutions to prevent the disclosure of some of these unintended inferences on user information. They rely on trust as a mechanism to establish a relationship between context producers and context consumers. They use a trust graph to identify the possible collusion between receivers and use it to determine how to adjust the quality of the shared data. Trust has different implications for data source and data receiver. They consider the context information as a resource not only to take access control decision but also to be used by context applications. The authors propose the use of obfuscation to reduce the risk of disclosure of valuable information.

One of the problems they try to solve is "How much context information to share?". Confirming that, the amount of data shared depends on multiple factors such as the information quality requested by the consumer, the trust that the producer has on the consumer, as well as the trust network of the consumer.

They express the above tradeoff between QoC and privacy as a linear problem, using a trust graph to quantify the risk of information leakage leading to privacy invasion. The solution relies on a combination of fine-grained access control and data obfuscation mechanisms to provide privacy-aware sharing.

The approach proposed by the authors is promising and has been implemented in the SensorSafe framework [Choi et al., 2011] and the obfuscation framework that are described below:

- **SensorSafe Architecture [Choi et al., 2011]**

  SensorSafe is a framework for privacy-preserving sharing of sensory information. SensorSafe provides rule-based sharing and basic sensor data obfuscation along with the detection of conflicting rules.

  The design of SensorSafe considers the following features. Firstly, SensorSafe provides fine-grained access control primitives for the specification of user's privacy rules based on temporal and spatial constraints, the identity of the data requester, the type of sensors, and sensor values. Secondly, the architecture, as depicted in Figure 3.11, uses distributed storage for sensor data which supports multiple institutional servers as well as personal data storage that users can trust. Thirdly, SensorSafe analyzes privacy rules associated with data to evaluate their utility for a particular query. That is, differences in individual preferences (access control and data perturbation) imply that the utility varies from one data source to the other. Therefore, to achieve a certain QoC is need to match data utility to application requirement. Finally, the architecture provides a library of data perturbation algorithms that can be suitably applied to protect sensitive inferences on the output data before sharing.

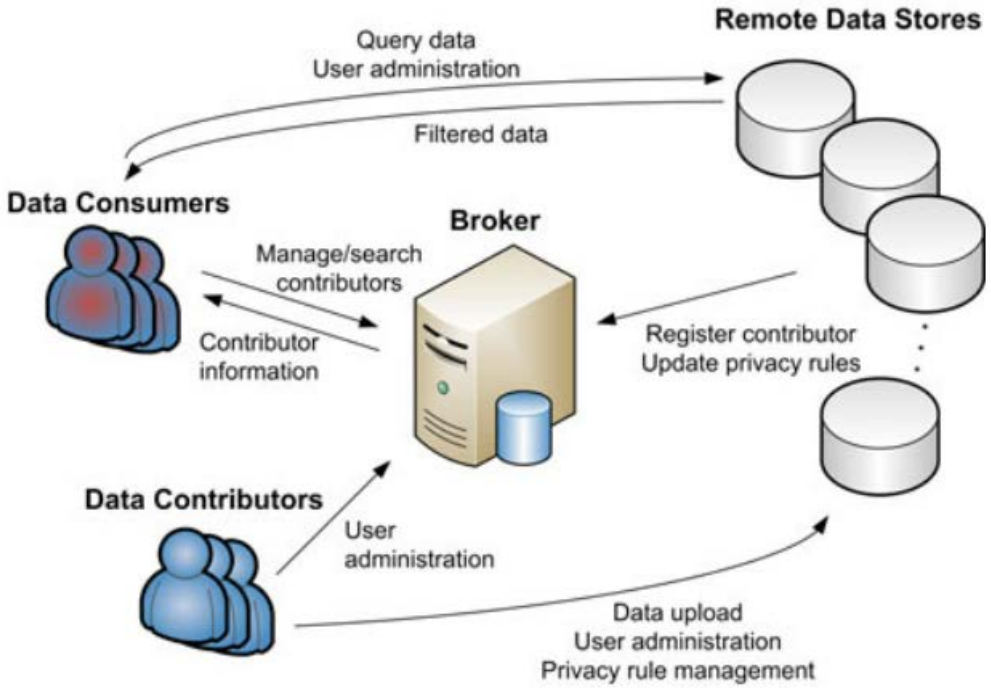


Figure 3.11: SensorSafe Architecture [Chakraborty et al., 2011]

- **Obfuscation Framework [Chakraborty et al., 2012]**

  In this work, the authors' main motivation is to control the information so as to preserve privacy. They focus on the use of obfuscation to control the possible inferences from shared sensory data to achieve a balance between the utility gained by the information consumers and risk experienced by the information producers.

  Additionally, they provide results from an activity tracking scenario to illustrate the use of the framework. Based on the following example:

  "Privacy and security measures available today mandate that applications declare what sensors they require during installation (or use) and a system framework provides a policing mechanism to ensure that undeclared resources are not accessed. However, suppose that the app developer maliciously included algorithms that also infer the type of a user's activity (sitting, sleeping, walking, running) from the same accelerometer data. Since the system provides the app with raw accelerometer data, both system and user are oblivious to the true behavior of this application. In today's mobile platforms, users must either fully trust the application or choose to not use it at all."

  Proposing that, mobile platforms should incorporate finer-grained access to sensor data by exploiting knowledge of common features that could be extracted from the data. Enabling better presentation of intent, in the sense that when applications or consumers specify the set of useful

inferences, they indirectly point to a set of features that need to be shared for those inferences to be made successfully. Hence, the producer provided with a white and black lists of inference functions, which are expected to be personalized, recipient-specific and context-dependent, can systematically design an obfuscation mechanism and evaluate whether sharing some data can be potentially divulging of private inferences incurring risk while simultaneously meeting the utility objective.

In the Obfuscation Framework, three interactions and possible ways are defined in which data flow can occur between the producers and the consumers. The authors illustrate their work through a mobile platform demarcated by the dotted box in the Figure 3.12. They assume that all the software components forming the mobile platform are trusted whereas the application layer (with third-party apps) is not trusted.

Depending on the type of interaction, the placement of the obfuscation framework may vary. In part (a), the framework needs to be placed between the mobile platform and the applications. Data streams passing through the framework layer are adequately transformed before they are handed over to the untrusted application. In part (b), the framework can be included into the client implementation in the application layer. In part (c), the obfuscation framework can be pushed to the trusted broker. This would eliminate the need for changes at both the application client the system level.

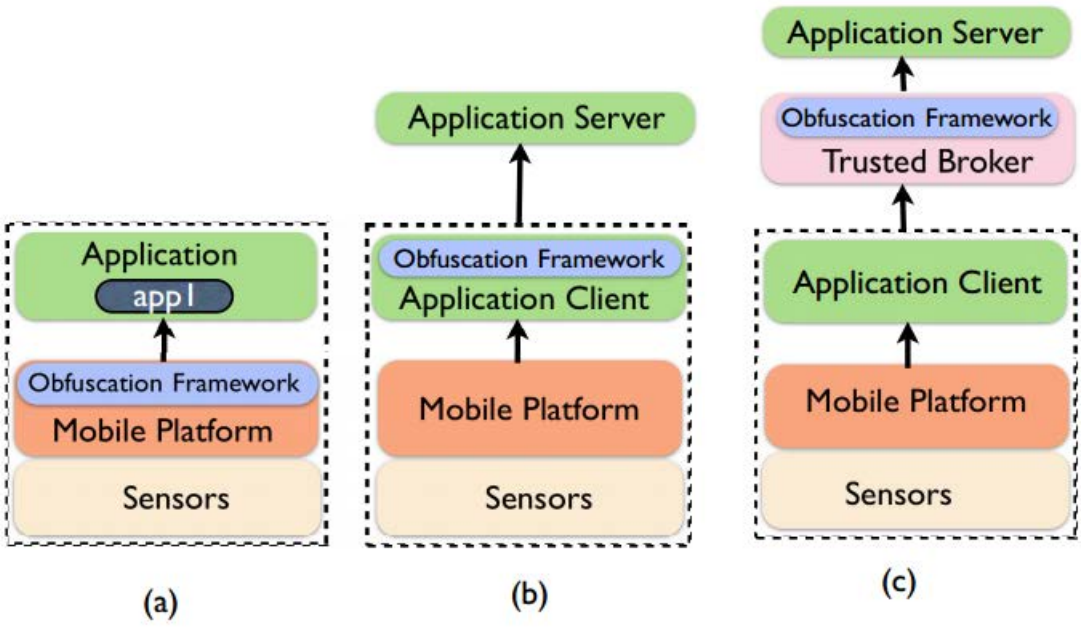


Figure 3.12: Different data flow paths from provider to consumer. Data shared with (a) locally running app (b) directly with cloud-hosted app (c) via trusted broker with cloud-hosted app. [Chakraborty et al., 2012]

[Choi et al., 2011] have a point in common with our research in that they take into consideration user context information (e.g., Medical sensors in healthcare systems) as a resource to be protected, to which applies obfuscation. These works inspired us to take into account trust as a mechanism to balance the producers' privacy requirements with the QoC expected by consumers. The notion of trust is therefore seen as a precursor to privacy and appears as a unifying paradigm describing both the desired obfuscation at the source and the expected QoC on the receiver side. Using the trust among participants in both directions, producers trusting consumers will delivering information with better quality. And consumers can trust that information will allow them to benefit from a good quality of context.

Although [Chakraborty et al., 2011] propose to use trust, it appears that trust management is not used to its full potential. The trust graph is static and should be defined before runtime. Also it does not allow to consider trust relationships depending on the current context situation.

The IoT requires a solution that considers dynamic connection and disconnection of producers and consumers. Therefore, trust between participants must be regularly evaluated as it varies with time and is context dependent.

The decoupling of participants is not supported by this solution. The use case of the Obfuscation Framework provides a mechanism to protect the context information collected by sensors in a mobile device belonging to the user. All sensors and potential consumers are known beforehand. However, in the IoT, sensors are not only embedded in the producers' devices, they are also scattered in the environment without the producer being necessarily aware of their presence. Therefore, producers do not know who are the potential consumers of their information. Similarly, consumers are not in the full capacity of knowing who are the potential producers.

3.7 Context-Based Trust and Privacy Management

The University of Twente [Neisse et al., 2008, Neisse, 2012] proposed a mechanism helping service providers to select context information providers taking into account their trustworthiness to provide context information at a given quality level.

Figure 3.13 illustrates the five roles distinguished by Neisse [Neisse, 2012] in a context-aware service platform. The arrows indicate the basic interactions between the roles when a user accesses a service provider. The box with a dotted line that surrounds the Context Owner represents sensors in the environment that collect context information about this entity. First, the Service User is authenticated thanks to the Identity Provider and receives an identity token (1). After the authentication is performed, the Service User requests access to a service provided by the Service Provider (2), which will verifies the identity token of the user in order to grant access to the service (3). To be able to adapt the service to the relevant context, the Service Provider requests context information about the Context Owner from the Context Provider (4). This context information is retrieved by the Context Provider from sensors in the physical environment of the Context Owner (5) which may be, for instance, the current activity or location of the Context Owner [Neisse, 2012].

As context-aware service providers depend on context information to adapt the delivered service reliably, this is possible only if the context provider can be trusted for reliably measuring and transmitting the context information with the required QoC characteristics. For Neisse et al., there is a direct rela-

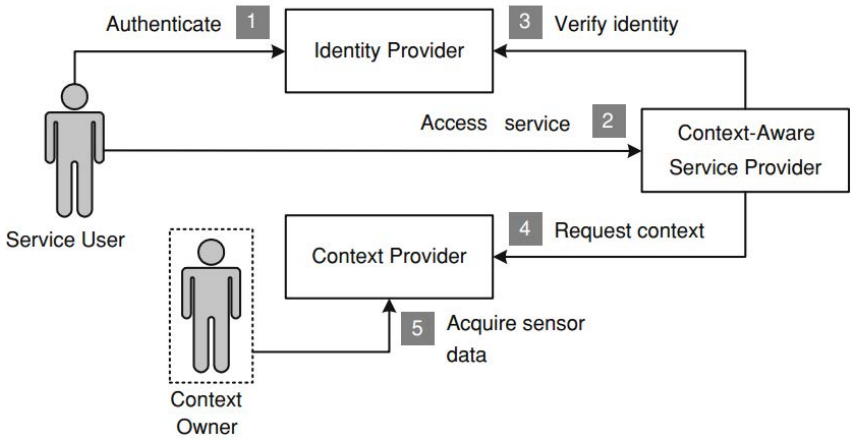


Figure 3.13: Roles In Context-Aware Platform [Neisse, 2012]

tionship between trust and QoC. Trustworthy context information providers capable of providing context information at high QoC levels contribute to the reliability of context-based service adaptation. However, this also increases the risks of privacy violation as more precise and reliable information is exposed. QoC can be seen as a means to protect the user's privacy through the use of obfuscation techniques associated to the level of QoC of the context information provided. Neisse et al. consider that trustworthiness is a property of the context provider from the context consumer's point of view, and that it should not be related to the context information itself and is therefore not part of the QoC model.

Neisse's work is very similar to what we want to achieve. For example, it proposes to use QoC as a mechanism to protect the privacy. However, in this proposal, there is no definition of the privacy elements to create privacy preferences. For example, there is no enforcement of obligations on purpose, visibility or retention. The privacy is a consequence of a good management of QoC and trust. Also, the service providers are the ones interested in using the context information. In the IoT, the context consumers are interested as well to access the context information provided by context owners. The final difference stems from the coupling of the parties. Both producers and consumers are known actors of the same system and the trust is used by consumers to select providers according to the QoC offered, which means that trust is a criterion of QoC.

3.8 Conclusion

With the IoT, new variables enter in the equation that are the decoupling of participants and the handling of the dynamic events occurring in pervasive environments. For instance, when someone enters a space equipped with sensors, one cannot avoid that these sensors get information that can be private, and the worst part is that we do not know who is consuming this information.

In this chapter, we analyze some related works and identified the differences with our work. We studied how they treated the aspects of privacy and QoC awareness in context-aware applications and

how we can adapt their work to the IoT characteristics. In Table 3.1, we summarize the works that have contributed to the definition of privacy and the identification of its different dimensions and how the various research works have evolved to include privacy as well as QoC concerns in context-aware systems. Additionally, this table classifies the different works according to the research domain (context-aware applications, QoC, privacy), the type of solution proposed (taxonomy, model, architecture) and if the problematics present in the IoT (trust management, decoupled participants) are taken into account.

	Research Domain			**Suggested Solution**			**IoT Issue**	
	Context Aware Applications	**QoC**	**Privacy**	**Taxonomie**	**Model**	**Architecture**	**Trust Management**	**Decoupled Participants**
[Dey et al., 2001]	✓					✓		
[Skinner et al., 2006b] [Skinner et al., 2006a]			✓	✓		✓		
[Duta and Barker, 2008]			✓	✓	✓			
[Barker et al., 2009]			✓	✓				
[Ghazinour et al., 2009a] [Ghazinour et al., 2009b]			✓	✓ (barker)	✓			
[Abid and Chabridon, 2011]	✓	✓	✓					
[Filho, 2010]	✓	✓	✓		Access Control	✓	QoC as Parameter	
[Neisse, 2012]	✓	✓	✓		Trust	✓	Trust and QoC to select providers	
Our Contribution	✓	✓	✓		Contracts Privacy, QoC, Trust	✓	Relationships in both directions	✓

Table 3.1: Summary of State of the Art

At the beginning, research works on context-awareness and privacy followed separate ways. The research about privacy was concentrated on describing privacy, whereas works on context-awareness proposed architectures to deploy context-aware applications. Then, the problem of handling QoC appears. Moreover, not every pieces of context information are good for all applications and, each application has its own requirements of quality. Filho [Filho, 2010] and Neisse [Neisse, 2012] were among the first ones to develop models and architectures taking into account both concerns in the development of context-aware applications. A difference we want to establish is the way we use the trust to build relationships among producers and consumers. The trust has been used as a QoC criterion by [Filho, 2010, Neisse, 2012]. We use the trust from two different perspectives. From a context producer point of view, priority must be given to the respect of the privacy. If a context consumer is not trustworthy, a strict privacy is desired. In contrast, for a context consumer, context information of high quality is expected from trustworthy sources. If not, possible errors might occur. We also use the trust to measure the compliance with privacy agreements from consumers, and QoC fulfillment from producer sources.

A last principle proposed by [Agrawal et al., 2002] aims to ensure that all principles to protect privacy will be applied. Therefore, the context owner must be able to verify the fulfillment of these principles.

We are interested in verifying the compliance of the proposed solutions with our proposed list of privacy protection criteria. Table 3.2 shows by author the compliance of each proposed solution with these criteria.

Model →	OoTProtocol	PPLEngine	UCPF	Context-Based Access Control	Sensor Safe and Obfuscation Framework	Context-Based Trust and Privacy
↓ Protection Criteria	[Mbanaso et al., 2009]	[PrimeLife, 2010]	[Bagüés et al., 2010]	[Filho, 2010]	[Chakraborty et al., 2012]	[Neisse, 2012]
Context-Aware Collection Control	-	✓	✓	-	-	-
Purpose Access Control	✓	✓	✓	-	✓	-
Retention Access Control	✓	-	✓	-	-	-
Disclosure Access Control	✓	✓	✓	-	-	-
QoC (Obfuscation)	-	-	-	✓ Only for access control decision	✓	✓
Trust management	✓	-	-	✓ as QoC criteria in access control decision	✓	✓

Table 3.2: Studied Solutions conformance to Proposed Privacy Protection Criteria List

In the state of the art, several research works consider mainly the consumer side to express data requests and check access rights. We argue that for contract management, it is also needed to express producer guarantees (what is called obligation in OoT [Mbanaso et al., 2009]). That is why we consider to manage both consumer and producer sides as an approach to safeguard privacy and QoC in the IoT.

Part II

Contributions - MUCONTEXT Contracts

Chapter 4

Motivations, Requirements and Architecture to Manage Privacy and QoC in the IoT

Contents

4.1 Introduction

People have subjective expectations concerning privacy, they have different perceptions of in what location or in what situation their context data should be private. This concern varies greatly from person to person. We develop in Section 4.2 an applicative case showing how privacy affects the personal life of a user. From this use case, we extract the requirements concerning the management of privacy and QoC in the IoT. We classify and describe in Section 4.3 these requirements through the concerns of context

situation identification, privacy and QoC. Section 4.4 proposes a software architecture to enforce privacy and QoC contracts. Finally, a conclusion of this chapter is presented in Section 4.5.

4.2 Motivating Scenario

The application scenario we propose takes place in a European city, which has an antipollution program that consists in encouraging the use of bikes. For that, the city hall has put around the city and suburbs bike points where citizens can take a bike for free during 2 hours. Additionally, each bike is connected to the "bike4all" system, which is the system that centralizes the bike control. Moreover, To make "bike4all" system attractive to the public, it is proposed to riders to optimize their time and enjoy as much as possible the riding experience according to each one preferences, including in protecting their privacy.

This system offers a pack of pervasive applications exploiting all the newest sensing and communication technologies: The city is equipped with many sensors for detecting environmental conditions, the presence of bike riders, and traffic accidents. Because, most of the citizens possess a smartphone with GPS positioning and other sensors the system uses these capabilities to communicate with users and get context data. Each traffic light has sensors that detect when a bike passes by and sends this information to the "bike4all" system. In this way, the system can calculate the volume of traffic on a street. Furthermore, the buses and trains have sensors as well to record if users are using the bikes on their complete paths or if they mix with other transports.

We illustrate this scenario with a personalized use case. David has subscribed to the "bike4all" system to enjoy all its advantages. We point out as the story unfolds, which of the identified requirements are concerned and exemplified by the use case. For each scene of the story, we want : *(i)* To identify which requirements are in play; *(ii)*To determine if we can create a contract to express both producers and consumers needs; *(iii)* To determine how easy it is to modify or adapt the contract when necessary.

"Meet Friends on the road" Scenario

> *David is riding back home. He decides to reveal his exact location to his acquaintances if they are close to him. He would appreciate crossing them on his path. Using David's location and knowing the location of his acquaintances, the system calculates and offers a new path, where friends can join and share their trip, or on the contrary a new path to avoid to cross them. To know David's acquaintances the system uses his contact list. For privacy reasons, David does not want to meet all the members of his contact list. Thus, he classified them in different groups and he requires different levels of QoC to his location for each group. For instance, his closest friends receive his exact location, some of his office mates receive a low quality location (e.g., +-10km error), and other contacts do not receive his location at all.*

"Other services on the road" Scenario

> *David receives notifications of applications that want to use his location, some of them want to provide him with free services, some with clear profitable objectives (e.g.,to show*

advertisements of stores around him) and some others non lucrative (e.g., send an ambulance in case of accident proposed by the fireman department). David rejects or accepts the services according to his convenience, taking into account that he can change at any time the QoC and conditions under which his information may be collected.

4.3 Requirements Analysis

The main goal of this thesis is to tackle the issue of the respect of privacy in the IoT. Therefore, we design the MUCONTEXT contract meta-model[3] (described in Section 5.3) that describes the concerns of privacy and QoC to set up users' preferences and context aware application behavior.

For the development of our solution, we consider creating agreements between context owners that do not know who will utilize their context data, and context consumers that do not know who will produce the context data. Moreover, this happens in an environment where context is always evolving for both owners and consumers.

This solution considers as well QoC concerns to take decisions about how to disclose private context data. Moreover, we pretend that the level of intrusion in private life can be configured changing the level of quality of the collected context information.

There are different methods for gathering requirements: document analysis, observation, throw-away prototypes, use cases and static and dynamic views with users. As for the INCOME project, we have chosen the analysis of requirements of [Bass et al., 2003]: process called Attribute-Driven Design (ADD). ADD is a top-down process driven by extra-functional requirements, each extra-functional requirement is expressed into a mini-scenario. Those requirements are then used to make adequate design decisions.

Because there is an interdependence among some requirements and to establish a priority of implementation, we classified the MUCONTEXT contract meta-model and agreement features into three types of requirements: 4.3.1 context situation requirements, 4.3.2 QoC concern requirements and 4.3.3 privacy concern requirements.

4.3.1 Context Situation Requirements

The context situation requirements refer to all the features related with the general behavior of the context management service that support the MUCONTEXT contract adaptation at run time. These requirements MUST respond to:

$R1$. Handle constantly evolving context data

Be able to handle context contract agreements changes based on the constant evolution of context data and take a decision of agreement adaptation due to a change in the user context.

Applicative case:

The context contracts should offer to context owners a way to indicate in which situation they agree to share their context data. In other words, the meta-model should provide the description of context

[3] MUCONTEXT stands for *multiscale context management framework* and is the name given in the ANR INCOME project to the framework dealing with context modeling and management.

situations in which the rules of privacy and QoC will be applied. Meanwhile, the middleware should continuously monitor the environment to allow one to evaluate the context predicates given in the context contracts.

In this applicative scenario, the middleware continuously monitors the environment to determine David's present situation to choose which rule to apply. For instance, David only shares his location with his acquittances when he goes home on his bike, after work. However, if the middleware detects that David is in an emergency situation, his location will be delivered to the ambulances and trustworthy people close to him.

4.3.2 QoC Concern Requirements

Modulate the QoC allows one to obfuscate context information. As a result, we can protect privacy. For this reason, we take into account the concern of QoC to support privacy requirements. At least, the context management service MUST:

$R2$. Handle evolutive requirements on QoC.

Offer a mechanism to enable the consumer applications to dynamically express some variations of either the QoC criteria that they are interested in, and the quality level they expect for each criteria.

Applicative case:

The meta-model should provide a way to describe the context consumers QoC requirements to look for context providers that correspond with their needs or adapt their process to the QoC level obtained. For its part, the middleware will deploy the new consumer context contracts to get the suitable QoC level.

For instance, in the aplicative case, David accepts to use the city EMS system. This service needs to know David's exact location to assist him if he has an accident. For that purpose, at the beginning, the system has been configured to request the best possible level of QoC for his location all the time. So that, an ambulance can reach him if an accident happens.

4.3.3 Privacy Concern Requirements

Finally, the context management service's requirements concerning privacy are the following:

$R3$. Use a Privacy Policy Language that supports QoC definitions.

Allow users to express authorization and usage control constraints based on QoC criteria.

Applicative case:

The meta-model provides both producers and consumers with a common language to express their requirements in terms of privacy and QoC. As each party does not know the other one, the meta-model offers a language to a define the guarantees on the same terms of privacy and QoC. The middleware should be able to understand this language to setup its behavior and create agreements between producers and consumers and modulate QoC and application behavior.

In the example, David reduces the quality of his location for some groups of acquaintances to protect his privacy. In the case of the city EMS system, David does not want to be followed all the time, even for

his safety. Therefore, he decreases the QoC level of his location level in a general case and increases the QoC level of his location only in an unsafe situation (e.g., in empty streets and when weather conditions are adverse).

$R4$. Evolution of Privacy Requirements

Be able to handle changes of privacy requirements. Once privacy requirements are defined, they must be applied immediately.

Applicative case:

The meta-model should provide a way to describe the context producer's privacy requirements to share context data with context consumers that correspond with its needs and provide the expected respect of their privacy. For its part, the middleware will deploy these new context contracts to get the required privacy protection.

The values, the perception of the world and the needs of an individual may change over time. Hence, it is logic that privacy needs may also change. For instance, because David is aware of risks of the lack of privacy, he may increase privacy safety modifying his privacy requirements over the time (e.g., to restrict the possible purposes, to change the context data visibility, not to share context data in specific situations). However, at other moments he can decide to relax these rules to use a given service.

$R5$. Specifying and enforcing privacy preferences

Provide mechanisms for specifying and enforcing privacy requirements. Users' privacy requirements may include authorization statements as well as usage control statements.

Applicative case:

The context owner expects the respect of his privacy requirements. The best way to ensure the respect of these requirements is by creating obligations for context consumers. The meta-model should provide a way to describe the context producer's privacy requirements and the context consumer's privacy guarantees independently. For that reason, the middleware should be able to match these requirements and guarantees to establish agreements between both parties prohibiting or granting context data access. Additionally, the middleware should be able to execute these rules to decrease/increase QoC to obfuscate context information to protect privacy.

In the applicative case, David defines who (i.e.; visibility) and how (i.e.; QoC) his context information is accessed. For example, when David defines who will have access his location, he expects that the application implements access control mechanisms to avoid unauthorized access. Similarly, David defines for what purpose a context consumer shall use his context information. He expects that end-users cannot consume it for any other purpose. Moreover, David customizes the level of QoC that will receive each of his acquaintances according to his personal criteria.

4.4 Proposed Architecture

This section specifies the proposed software architecture to enforce privacy and QoC contracts between producers and consumers in the IoT. "The software architecture is the structure or structures of the

system, which comprises software components, the externally visible properties of those components, and the relationships among them" [Bass et al., 2003]. In this section, we present how the proposed software components are included in the INCOME IoT architecture.

The section begins by a description of the proposed architecture in Section 4.4.1. Section 4.4.2 details how the architecture is structured into software components. We also show how the contracts components are integrated into the software architecture of the INCOME project. This section highlights the layer included to tackle the concerns of QoC and privacy in the IoT. Finally, Section 4.4.3 describes the organization of the modules and sub-modules to be developed.

Each section contains the following elements:

- A primary presentation that depicts the main architecture elements of the view and the main relationships between them.
- A catalogue of elements that explains and defines the elements shown in the view and that lists their properties. It is a list of element names with textual description;

4.4.1 Coarse Grain Architecture

In this section, we show a coarse grain architecture of the entities that manage the privacy and QoC requirements and the actors using this architecture. We present figures of this architecture in Section 4.4.1.1, and the catalogue of elements in Section 4.4.1.2.

4.4.1.1 Primary Presentation of the Architecture

This section introduces the proposed software architecture. Figure 4.1 displays the positioning of producers and consumers and the MUCONTEXT Contract Manager Service. In the middle, the IoT cloud takes over the distribution of context data meanwhile MUCONTEXT Contract Manager Service organizes the privacy, QoC and obfuscation management and their data models.

Figure 4.2 shows this architecture in action for a smart home use case. We observe in this example that several lucrative or non-profit organizations are interested in the information generated by a person's home. The purpose of these consumers are all different. For example, some want to offer new products or services, others to manage energy saving, or conduct a scientific study. In any case, there are many private pieces of information that may be disclosed. We believe essential to enforce a control mechanism on this information. In the proposed architecture, the MUCONTEXT contract manager service provides the implementation of privacy policies enforcement and realization of obfuscation algorithms. This service aims to reduce the risk of information leakage. In the same time, the service takes into account the QoC requirements of the consumer applications.

We assume that the MUCONTEXT Contract Manager Service is executed as part of an IoT framework which controls the forwarding and routing process of context data. The framework uses filters, which analyze the content of the context reports and decide if it corresponds to the context data requested. The framework provides also context/situation information and trust information about the users. Besides, we consider that the producer and consumer contracts are registered in this IoT platform. Thus, when a context report that corresponds to the information requested by a context consumer is received, a chain of

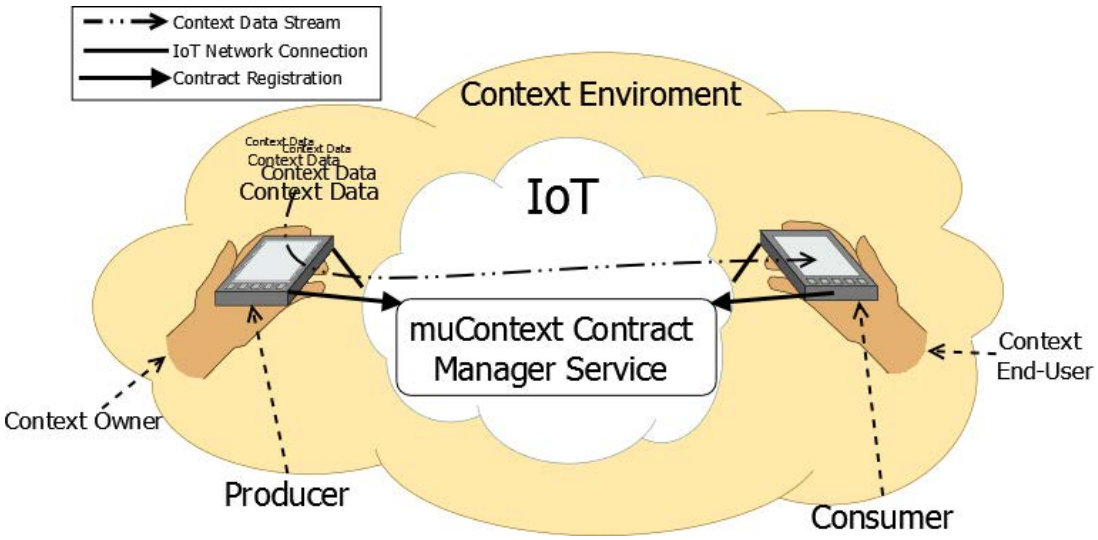


Figure 4.1: Proposed IoT Architecture (general schema)

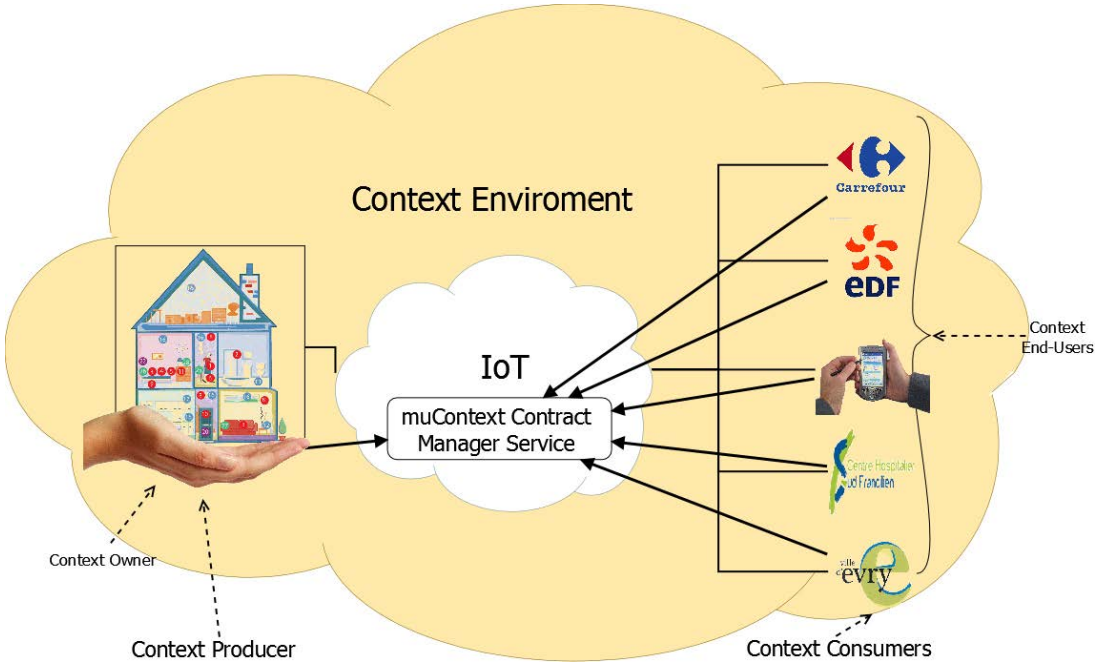


Figure 4.2: Architecture Example

validation is executed to verify if the contractual requirements and guarantees of both IoT participants are in agreement. As a result, the framework provides or not access minimizing the risk of privacy violation, and the framework forwards context reports to consumers only if they provide the expected QoC. All this

process is detailed in Section 4.4.2.1.

4.4.1.2 Catalogue of Elements

- **Context Data:** Context data represent useful raw data that have been directly acquired by a Context Collector [4] through sensing, observing, or measuring some facts. [Arcangeli et al., 2012]
- **Context Data Stream:** It is a sequence of context data (observations) made available over time.
- **IoT Network Connection:** Is a telecommunications network that allows various producers, such as communicating objects, to publish data, and consumers to receive data. The connections between producers and consumers are provided by the Internet.
- **Context Owner:** A person, a group of people, or an organization that defines privacy rules over his/her/their/its context data, to decide with whom and in which conditions these data will be shared. He/she/they initialize a producer context contract to establish his/her/their privacy preferences.
- **Context Producer:** is a software entity that provides context data with QoC guarantees.
- **Context Environment:** Is a context model used to define the enclosing environment of some system. In other words, the context environment is the surrounding context data of a system, that provides an interface and a behavioral description of the surrounding environment. It is used to characterize the context environment in which data are produced and consumed.
- **MUCONTEXT Contract Manager Service:** It is a functional element of the architecture in charge of providing the necessary information to evaluate MUCONTEXT contracts to permit, obfuscate, or deny access to context data. This service is provided from heterogeneous sources of information. The MUCONTEXT contract manager service is shared by producers and consumers. It is defined in the contracts as the context manager party. It provides the same reference and knowledge about trust, visibility, purpose, retention, QoC, and environment variables. For example, a producer and a consumer define that context data can be used for some purpose, the MUCONTEXT contract manager service contains the complete definition of that purpose to avoid misunderstandings. Moreover, it agrees on a standard data format of this information.
- **Contract registration:** The producer or consumer registers producers and consumers contracts in the Contract Manager Service. Those contracts contain QoC and privacy requirements and guarantees.
- **Context Consumer:** Entities that require context data, i.e., context-aware applications.
- **Context End-user:** A person, a group of people, or an organization that accesses or uses context data through context-aware applications.

[4] **Context collector**: A context collector is an element that provides raw context data. A context collector is a context producer. A context collector encapsulates the behaviour of acquisition, in a push or a pull mode, of raw data from the observed world (e.g., a context collector can abstract a client of an RFID reader or a COAP server collecting data from a network of sensors).

4.4.2 View "Component"

This section introduces the software architecture of the INCOME project, and especially the MUDEBS framework (multiscale Distributed Event-Based System) [Conan et al., 2013]. Besides, we describe the layer added to this architecture to manage the privacy and QoC concerns. This view shows how the system works by specifying the structure and the behaviors at run-time.

The viewpoint is "Component and connector" [Clements et al., 2011]. It shows how the system is structured as a set of elements that have run-time behavior and interactions. More precisely, components are processing units and data-storage units that communicate through sets of ports, which interact via connectors.

We briefly present the MUDEBS framework and the MUCONTEXT contract manager service layer in Section 4.4.2.1, the catalogue of elements of this view in Section 4.4.2.2.

4.4.2.1 INCOME architecture layers and components

We extend the software architecture of the INCOME project [Conan et al., 2013], adding the MUCONTEXT contract manager service layer to include the concerns of QoC and privacy in the IoT.

The INCOME architecture addresses new challenges brought by the IoT in terms of spontaneous interactions, and in term of amount of transient context data. The key feature underlined by the INCOME architecture is the distributed push communication mode adapted to the spatio-temporal decoupling of context management entities while guaranteeing context data delivery thanks to specific advertisement and subscription filters, and thanks to efficient routing algorithms.

This section depicts the general software architecture of the INCOME project in which is included the MUCONTEXT contract manager service.

Figure 4.3 shows the architecture of MUCONTEXT with its different elements and layers. Each layer is described below:

- MUCONTEXT: Context collectors (software entities which represent producers) and context capsules (software entities that transform context data) constitute what we call MUCONTEXT, which stands for multiscale context management.
- MUDEBS: MUDEBS is the framework responsible for distributing context data. MUDEBS provides an interface not dedicated to context entities. The framework MUDEBS is designed for allowing the distribution of data in large-scale and heterogeneous systems involving clouds, cloudlets, desktops, laptops, mobile phones, and smart objects of the IoT.
- MUCONTEXT contract manager service: is a middle-layer that implements all the components required to enforce privacy and QoC. This layer rules the way in which context data are published and consumed. This service is split into, the producer contract manager, the consumer contract manager and the privacy information point.

Represented in Figure 4.3, the MUCONTEXT contract manager service layer provides protection in terms of privacy and QoC to producers and consumers in the MUCONTEXT architecture. There are three parts in the process, one part is handled by the producer side, another part is handled by the consumer

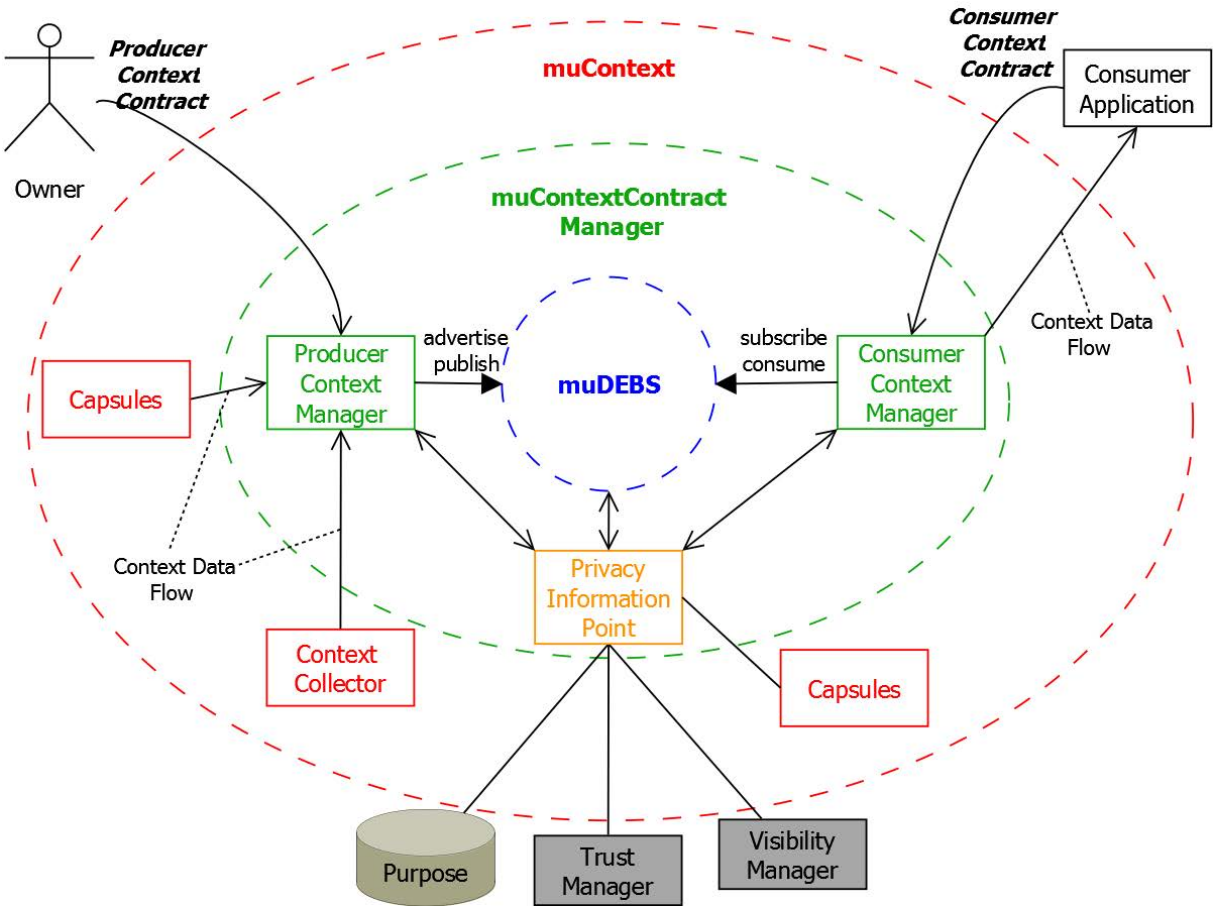


Figure 4.3: MUCONTEXT Contract Management Layer Architecture

side, the last part is handled in the middle, it is called contract matching and verifies if the producer requirements are validated by the consumer guarantees as well as if the producers guarantees are sufficient for the consumer requirements. It will be detailed in Section 4.4.3.

The producer contract manager

Firstly, to explain the producer part of this architecture let us assume that the context owners have already registered their producer MUCONTEXT contracts.

Context collectors and capsules create the advertisements and publish flow of context reports. Each report is associated to a registered producer MUCONTEXT contract. The producer contract manager captures context reports.

The producer contract manager publishes context reports only when the context conditions specified in the producer MUCONTEXT contract clause predicate are accomplished. To know when to publish or not, the producer contract manager queries the policy information point to evaluate these predicates, for

example, determine if the context owner is in danger (e.g., query connected smoke detectors, if specified in the contract).

To grant access to context consumers, the producer contract manager queries the privacy information point to matching context consumer guarantees with context producer requirements, for example, it may query external services to get trust among the members of a network, to measure the scope (visibility) of a consumer, or to verify the purpose of an application.

In some cases, for some context consumers, producer MUCONTEXT contract manager may modulate the QoC of context data based on privacy terms (the visibility, purpose and trust of possible consumers). For that objective, it can modify the reports according to the privacy requirements to obfuscate some data.

The consumer contract manager

The second part of this middle-layer is the consumer part. As we did in the previous part, let us assume that capsules and consumer applications have provided their consumer MUCONTEXT contracts with their QoC requirements and privacy guarantees. The "consumer contract manager", which is connected to a broker, subscribes to filters, which enforce consumer MUCONTEXT contract conditions.

The consumption is done based on the respect of the privacy and QoC terms. It necessitates a matching process between producers and consumers to determine (i) if the context producer generates context data with the QoC wanted for the smooth operation of the consumer application and (ii) if the consumer provides the privacy guaranties required by the producer. As a consequence, the consumer contract manager decides whether context data is finally delivered or not.

Additionally, another validation is executed on the consumer side based on the clause predicates written on the contract, the process of deciding to consume or not is based on the context situation. Context information may be required under specific context circumstances (e.g., to measure a home pollution levels only when people is at home). This validation contributes preserving privacy guaranteeing that people is only observed when it is strictly necessary.

4.4.2.2 Catalogue of Elements

- **Context capsule**: A set of context nodes constituting a functional element of the context management service and packaged as a unit of deployment. The role of a capsule is to transform context reports to produce higher level information.

- **Context collector**: A context collector is an element that provides context reports. A context collector is a software representative of a context producer. A context collector encapsulates behaviour of acquisition, in a push or a pull mode, of raw data from the observed world (e.g., a context collector can abstract a client of a RFID reader or a COAP server collecting data from a network of sensors).

- **Trust manager**: It is a service responsible for coordinating trust management tasks from which to query trust values about context owners, end-users and consumer applications.

- **Privacy Information Point**: The Privacy Information Point provides an interface that allows the MUCONTEXT contract manager service to get the necessary information to the MUCONTEXT contract matching process (see Chapter 6) (e.,g., the trust level of a context consumer, the meaning of a purpose, the circle of visibility to which belong a context consumer).
- **Visibility manager**: It is a service responsible for determining the visibility level of a context end-user or context consumer. This service holds the classification given for the context owner about the context end users.
- **Purpose repository**: Is a public shared database of well known purposes for which the context data may be used.
- **Consumer application**: An application that requires context data. It initializes a consumer MUCONTEXT contract to establish its QoC requirements and privacy guarantees.
- **Producer Contract Manager**: It intercepts the data flow from context collectors and publishes context reports under certain conditions given by the producer MUCONTEXT contract, it may also enforce obfuscation, for instance to decrease the QoC.
- **Consumer Contract Manager**: It registers the subscription filters based on the consumer MUCONTEXT contracts. It verifies the matching with producers advertisements filters before delivering context reports to the consumer applications. The Consumer Contract Manager guarantees that context applications acquire context data with appropriate QoC.
- **Policy Information Point**: the Policy Information Point looks for observables current status required to evaluate the MUCONTEXT contract condition predicates. It plays an important role in the adaptation at run-time of MUCONTEXT contract management. It uses context data itself to provide the necessary information to decide whether a MUCONTEXT contract clause is activated or not.

4.4.3 View "Dynamic" and matching process

In this section, we detail the MUDEBS framework components and especially one of its mode, which is the "local advertisement and global subscription". Furthermore, we present how is made the integration of protection of privacy and QoC in the the distribution of context data.

MUCONTEXT Contract Management adapted to MUCONTEXT and MUDEBS

Conan et al. [Conan et al., 2013] presents the architecture of MUDEBS as follows: "MUDEBS organizes an overlay network of brokers that connect producers and consumers. Producers and consumers are collectively called clients. A client is connected to only one broker at a time which is called the access broker. An advertisement expresses the set of publications that a producer is allowed to publish. A subscription expresses the set of publications that a consumer wants to consume. In practise, a filter is a function, written in JavaScript, that evaluates XPath expressions and returns false when the notification does not match the filter, or returns true when it matches the filter."

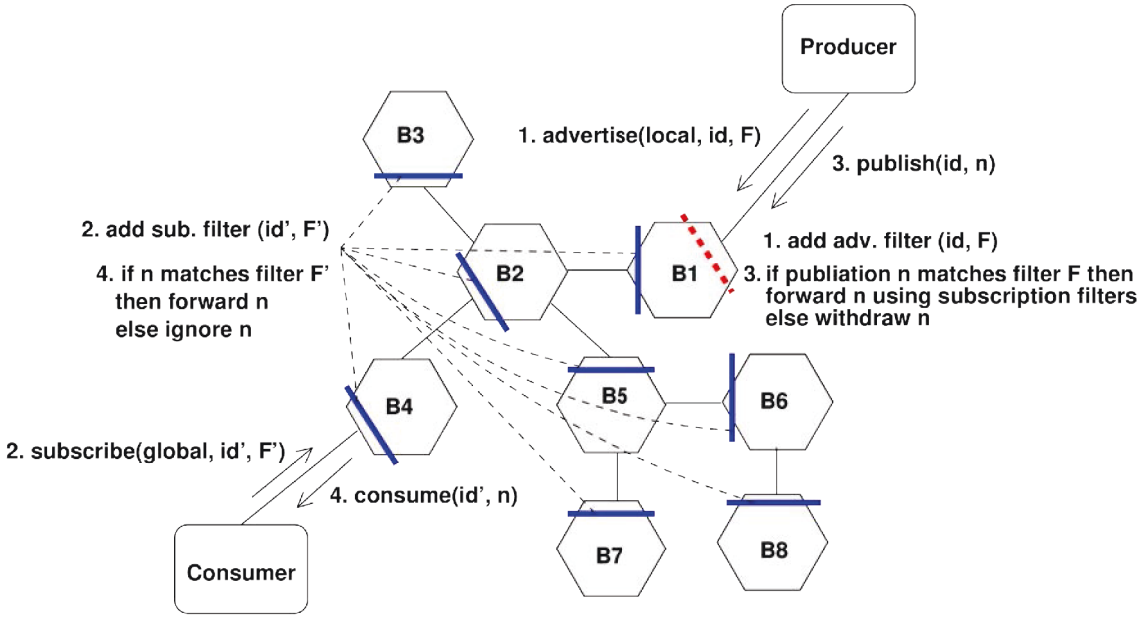


Figure 4.4: Local Advertisement and Global Subscription [Conan et al., 2013]

MUDEBS provides two operational modes for subscription and advertisement: global subscription with a local advertisement (mode 1), and a local subscription with a global advertisement (mode 2). For this thesis, we have used the first operational mode (in Figure 4.4) for the implementation of the proposed algorithms. In this mode, advertisements are kept local to the access brokers of the producers, and brokers forward subscriptions to their neighbouring brokers according to a simple routing mechanism. This mode has the advantage of minimizing the notification traffic.

MUDEBS operation mode 1 (*local advertise and global subscription mode*) is defined as follows: *A producer advertises a local filter A-F (depicted by a dotted segment), uniquely identified by its id, by calling the advertise operation on its access broker B1. B1 then registers the advertisement filter. Later on, a consumer registers a global subscription filter S-F' (depicted by a solid segment) to its access broker. The global subscription filter S-F is then installed on every broker building a spanning DAG (Directed Acyclic Graph) directed towards the consumer. When the producer publishes a notification n in the context of the filter A-F, the access broker filters out n if it does not match A-F. Otherwise the access broker of the producer evaluates all the subscription filters it is aware of. When n matches a subscription filter, let say S-F', the notification n is forwarded towards the subscriber of S-F' via the access broker of the subscriber, which is notified of n.* [Conan et al., 2013]

Every broker must implement a component of the MUCONTEXT Contract Manager service to match MUCONTEXT contracts in order to decide forwarding or not context reports according to the producer and consumer contracts. Figure 4.5 shows the data-flow diagram of this integration proposition.

1. The **Context Owner** provides a producer contract with their privacy preferences and QoC guaran-

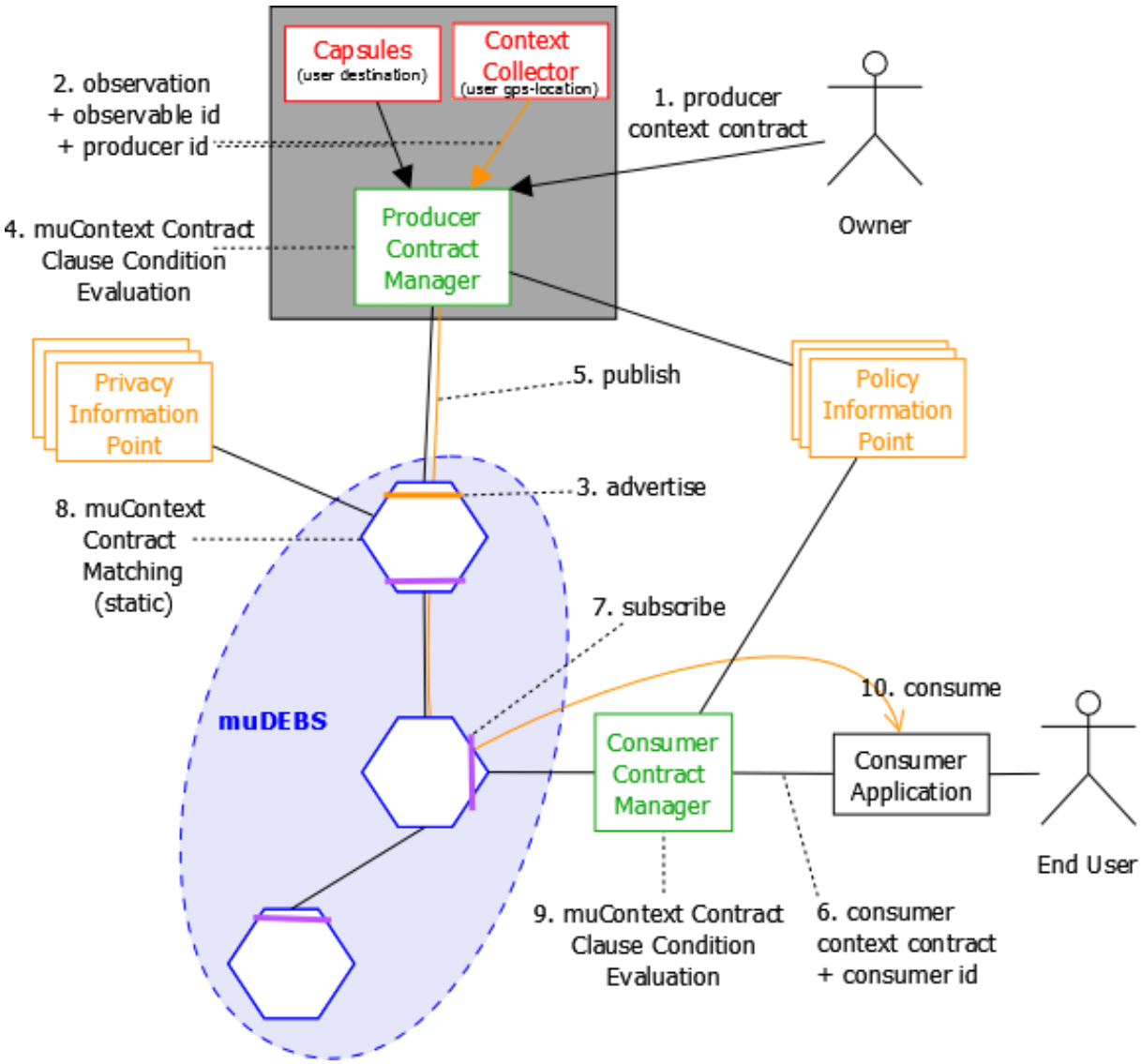


Figure 4.5: MUCONTEXT Contract Management Data-flow Diagram

tees to the **Producer Context Manager**.

2. **Capsules** or **Context Collectors** get context data from the context environment and send it to the **Producer Context Manager** identifying the observation (observable id) and who had produced the information (producer contract id).

3. The **Producer Context Manager** generates an advertise filter for the observable including the owner's privacy requirements and QoC guarantees.

4. The **Producer Context Manager** evaluates the MUCONTEXT contract clause condition predicates (see 6.3) to decide whether the observation affected by the contract can be published.

   Optionally, the **Producer Context Manager** generates several flows of data with several QoC levels. For this purpose, the manager calls obfuscation filters based on the contract specifications. Before each publication (push), the context manager adapts the QoC of context data to deliver context data with the guarantees offered by the context owner.

5. The **Producer Context Manager** publishes the observations as they arrive.

6. The **Consumer Application** sets a consumer contract with its privacy guarantees and QoC requirements to the **Consumer Context Manager**.

7. The **Consumer Context Manager** generates a subscription filter for the observable required including its QoC requirements and privacy guarantees.

8. At this step, takes place our MUCONTEXT matching algorithm. The validation of producer and consumer requirements is detailed in Sections, 6.4 and 6.5.

9. The **Consumer Context Manager** evaluates the MUCONTEXT contract clause condition predicates (see 6.3) to decide whether it is appropriate to consume the observation affected by the contract.

10. The observation is consumed by the **Consumer Application**.

4.5 Requirements Summary

The extra-functional requirements used to define context contracts are detailed in Appendix B. Table 4.1 summarizes the five requirements identified in this chapter.

The first column contains the requirement identifier used in the document to identify each requirement. The second column contains the requirement title. The third column specifies the concerns associated to each of the requirement. The next columns show the stimulus source and the artifacts. The stimulus sources are the entities or events that trigger a specific functional reaction in the system. In our case, there are three stimulus sources: the producer, the consumer, and the context situation. The artifacts are the parts of the system that are stimulated. The artifacts correspond to the meta-model and the middleware. Each artifact is sub-divided according to where requirements are executed or implemented. Each cell indicates what or how we expect that the artifact reacts to the stimulus. The meta-model artifacts are divided into "Producer Context Contract" and "Consumer Context Contract". On the other hand, the middleware artifact is split into three parts: the "Producer Context Manager", the "Broker" and the "Consumer Context Manager".

Req.	Title	Concern	Stimulus Source			Artifact				
						Meta-Model		Middleware		
			Producer	Consumer	Context Situation	Producer Context Contract	Consumer Context Contract	Producer Context Manager	Broker	Consumer Context Manager
R1	Handling constantly evolving context data	Context Management			Changes in producer or consumer context situation	Provide context predicates	Provide context predicates	Evaluate new context situation		Evaluate new context situation
R2	Handling evolutive specifications on QoC	QoC		Design QoC requirements			Provide QoC requirements			Deployment of new consumer context contract
R3	Privacy policy language that supports QoC definitions	Privacy	Design privacy requirements with QoC guarantees	Design QoC requirements with privacy guarantees		Provide producer context contract	Provide consumer context contract			
R4	Evolution of privacy requirements	Privacy	Design privacy requirements			Provide privacy requirements		Deployment of producer context contract		
R5	Specifying and enforcing privacy preferences	Privacy	Design privacy requirements	Design privacy guarantees		Provide privacy requirements	Provide privacy guarantees	Manage obfuscation	Match guarantees vs requirements	

Table 4.1: Requirements Summary with Stimulus Sources and Artifacts

4.6 Conclusion

In the first part of this chapter, we have presented a motivating scenario that illustrates a personalized use case in which the privacy and QoC concerns are required. Based on this use case, we figured out the requirements to create applications for the IoT that respect both concerns. These requirements were split into three groups: one related to the middleware behavior to support this kind of application; a second group related with the management and obfuscation of QoC to protect privacy; a third group of requirements to define the privacy concern and its management. Finally, in this chapter, we have presented a software architecture to enforce privacy and QoC based on the previous requirements. We follow a general to specific approach to present the components of this architecture. From a coarse grain architecture view that shows the actors and the entities that manage the privacy and QoC on the IoT, we derive a more detailed view of the architecture of the INCOME project in which we have added a layer to handle privacy and QoC taking into consideration the dynamism of context environment. We show how our contributions (models and algorithms) are integrated within the INCOME architecture.

In conclusion, the defined requirements allow us to identify several autonomic cases that must be handled by the context management in the IoT concerning privacy and QoC, such as, (1) adding, modifying and removing privacy and QoC requirements and guarantees at runtime, (2) taking into account dynamic events such as modification of trust, visibility circles and context data, (3) implement an autonomic matching of context contracts and choice of appropriate requirements and guarantees.

Additionally, the distributed architecture design is important. It shows where will be handled the different elements of our models and algorithms. We realized that we need to split the algorithm into three

parts, executing on the producer side, on the consumer side and in the middle with the matching part. The producer and consumer parts often query the environment to decide when to share and deliver context data, and the matching part only seldom matches the privacy and QoC requirements and guarantees.

Chapter 5

MuContext Contract Meta-model to define Privacy and QoC concerns

Contents

5.1 Introduction

As mentioned in section 2.6, we use the notion of contract to constrain the use of context data by IoT participants (i.e., consumers and producers of context data). Producers and consumers are decoupled (i.e., they are not in the same place, they are not directly connected). These contracts allow producers and consumers to express their constraints and they allow the system to verify the agreements between the two parties.

This chapter presents our contribution of MUCONTEXT contracts following a model-driven approach to formalize the standard representation to express the requirements and guarantees of both parties.

This chapter is organized as follows. Firstly, we introduce the participants in the establishment of MUCONTEXT contracts and we present the role of the contracts from an architectural vision in Section 5.2. Secondly, we show the MUCONTEXT contract meta-model from a general view in Section 5.3. Finally, we present in Section 5.5 the MUCONTEXT contract meta-model validation through the implementation of an editor and a use case application.

5.2 Contracts between Producers and Consumers

In this section, we present the software architecture in Section 5.2.1. Then, we present the links between context entities and MUCONTEXT contracts in Section 5.2.2. Finally, we present the concepts defined in the contracts in Section 5.2.3.

5.2.1 MUCONTEXT Contracts in the IoT Architecture

As introduced in Figure 1.1, the major IoT players are the following. The **context owner** is a user that has the capacity of decision about his/her/its privacy. The **context producer** is a software entity that provides context data. The **context consumer** is a software entity that requires context data. The **context end-users** is a person who uses a context consumer application. Finally, the **context manager** is a context entity in charge of collecting, transforming and providing context data to context consumers. Following MUCONTEXT contract, it is in charge of allowing or denying access to context data to consumers with the appropriate level of QoC, while preserving the privacy of the context owner.

We present in Figure 5.1 the context management architecture. This figure highlights which participants define the guarantees and requirements in terms of privacy and QoC.

Context producers define the **privacy** requirements concerning the data they produce. The privacy requirements define in which conditions the data may be shared. On the side of context consumers, privacy guarantees state the level of protection that these consumers can guarantee when manipulating context data. This includes stating explicitly for what purpose they want to access context data.

The use that context consumers can make of context data depends on the quality of these data (**QoC**). The consumer side has requirements concerning the QoC. They define the minimum QoC required in order to take appropriate decisions. Symmetrically, context producers express in QoC guarantees the level of QoC they are able to guarantee for the context data they provide.

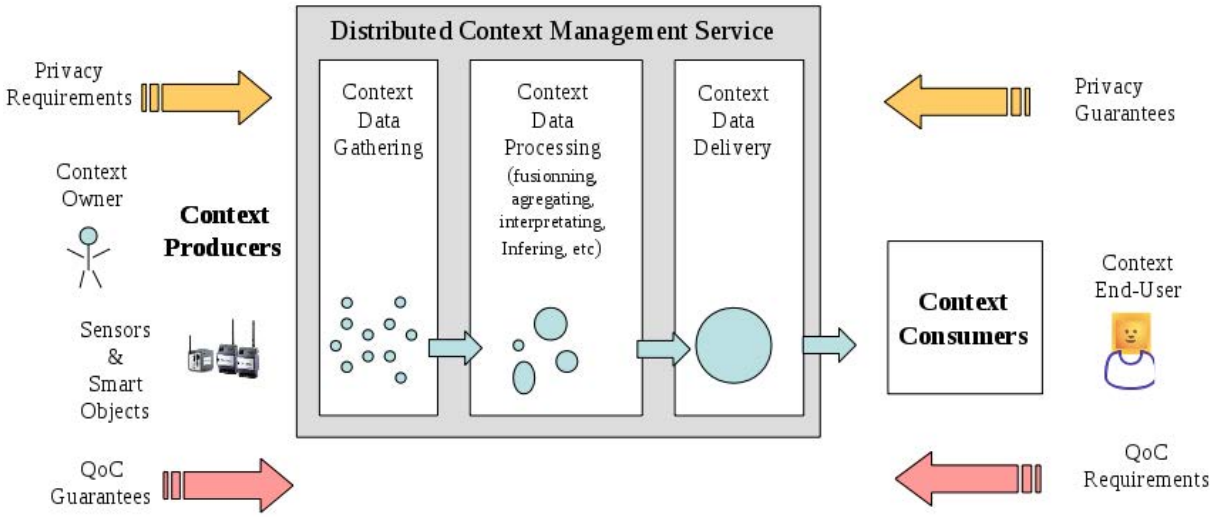


Figure 5.1: Considering Privacy and QoC requirements and guarantees in the Context Management Architecture (adapted from [Chabridon et al., 2013])

Based on both privacy requirements and guarantees and QoC requirements and guarantees, we propose to define two kinds of MUCONTEXT contract for the producer side and the consumer side as follows:

- **Producer Context Contract:**
  A context contract whose clauses are expressions of the production of context data, of privacy requirements, and of QoC guarantees;
- **Consumer Context Contract:**
  A context contract whose clauses are expressions of the consumption of context data, of QoC requirements, and of privacy guarantees;

The creation of these MUCONTEXT contracts is under the responsibility of a Consumer/Producer application designer. The application designer should be someone trained to analyze and identify the existing privacy gaps of IoT applications. He/She can propose one or more MUCONTEXT contracts to avoid these privacy gaps. Later, context owners/context end-users can customize these contracts according to their necessities. The SPEM diagram 5.2 shows the steps of this process.

5.2.2 Two Types of Context Contract

We identify in this section the relationships among the different parties involved in context contracts. Figure 5.3 represents the two kinds of contract. Each column represents one of the involved parties. Horizontally, we find the contracts represented by dotted rectangles. Finally, inside each rectangle, we find the privacy and QoC Terms, which represent the guarantees and requirements of each party. These guarantees and requirements should be in concordance to establish an agreement.

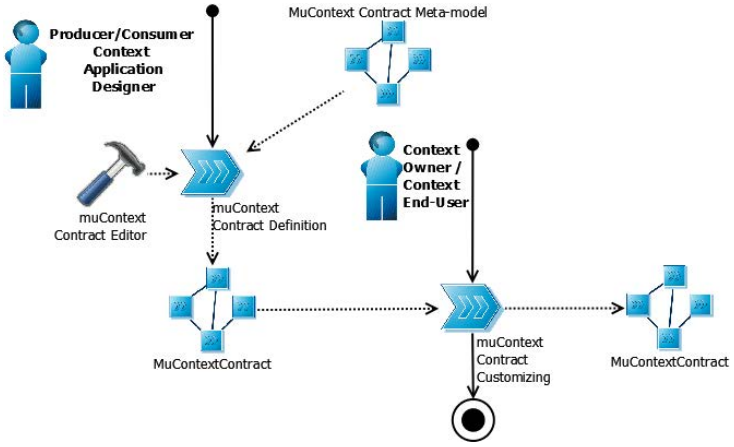


Figure 5.2: MuContext Contract Creation Process

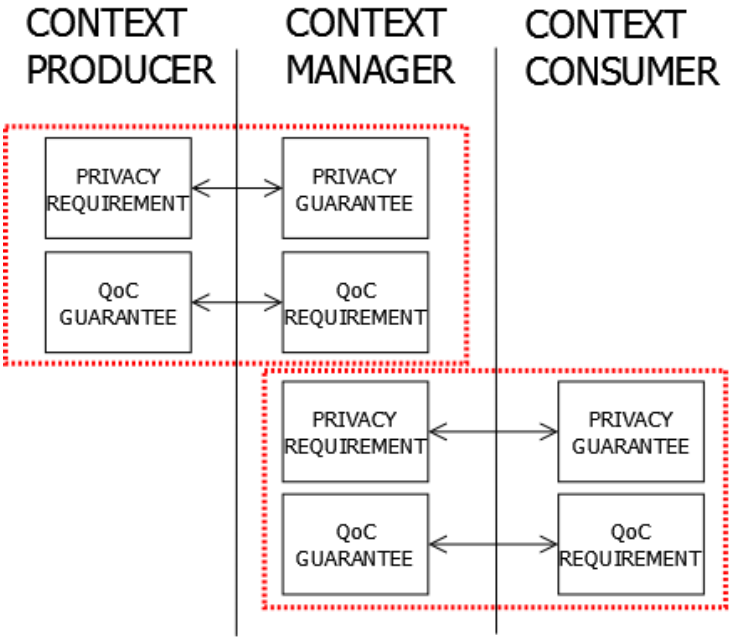


Figure 5.3: The two Types of MUCONTEXT Contracts

A context contract is always composed of two parties, one party for the requirements and another one for the guarantees. However, as context producers and consumers do not know each other, they only have indirect interactions. The matching is verified at runtime through the intermediate context management entities. The context managers are responsible for the decoupled participants problem.

Producer - Manager Contract

Assuming that context owners already have defined their privacy requirements, the context manager is obliged to guarantee these requirements.

The contract involves a mutual obligation between both parties in the accomplishment of some guarantees and each party has its respective set of demands. In this case, the context manager is committed to protect the context owner's privacy as if it was the context owner itself. The context manager will guarantee a specific level of QoC.

Manager - Consumer Contract

This contract is similar in structure to the Producer-Manager contract. Nevertheless, this contract is defined directly between the context manager and the context consumer. The context manager is representative of the context owner interests, it verifies the consumer privacy guarantees. The context manager also advocates for the context consumer needs demanding some level of QoC.

5.2.3 Context Contract Dimensions

In this section, we build on the privacy taxonomy presented in Section 2.5.3. Following this taxonomy, we determine several dimensions in which will be defined MUCONTEXT contracts in terms of privacy. Concerning privacy, the dimensions are visibility, retention and purpose. Concerning the statement of QoC requirements and guarantees, we rely on QOCIM (introduced in Section 2.3) to determine a global QoC indicator to be manipulated and which may result from a combination of various QoC indicators.

We add the **Trust** dimension as a way to bridge the concerns of privacy and QoC in the next generation context management solutions for the IoT. As context managers have to deal with the decoupling of context producers and consumers, trust then becomes critical to enable interactions between them.

As discussed in [Bisdikian et al., 2012], trust has different implications depending on the point of view that is considered. From the point of view of context consumers, trust represents the consumers' degree of belief that they can rely on the context data that a context producer has provided them with. On the other side, from the point of view of context producers, trust indicates the producers'degree of belief that a consumer will use the information they provide in the full respect of the expressed rules in terms of purpose, retention and visibility. The factor of trust is then a critical element to consider for handling context contracts at runtime. In our model, the trust determines which rules can be applied.

Figure 5.4 represents the different elements that must be taken into account when a context contract is created or designed. This figure also shows an example taken from the motivating scenario "Meet Friends on the road" in Section 4.2. David wants to meet his girlfriend. For that, David shares his location with her, but only until they meet. Additionally, the figure presents some examples of the different dimensions and trust levels. Each square encloses the party involved and its role in the contract. For instance, the first square (up-left) indicates the requirements of the context owner in terms of privacy, (e.g., the purpose of meeting known people). The second square (down-left) represents the guarantees offered by the context producer in terms of QoC (e.g., to deliver a medium QoC level). The third square (up-right) represents the QoC requirements of the context consumer (e.g., to receive a medium QoC level). Finally, the fourth

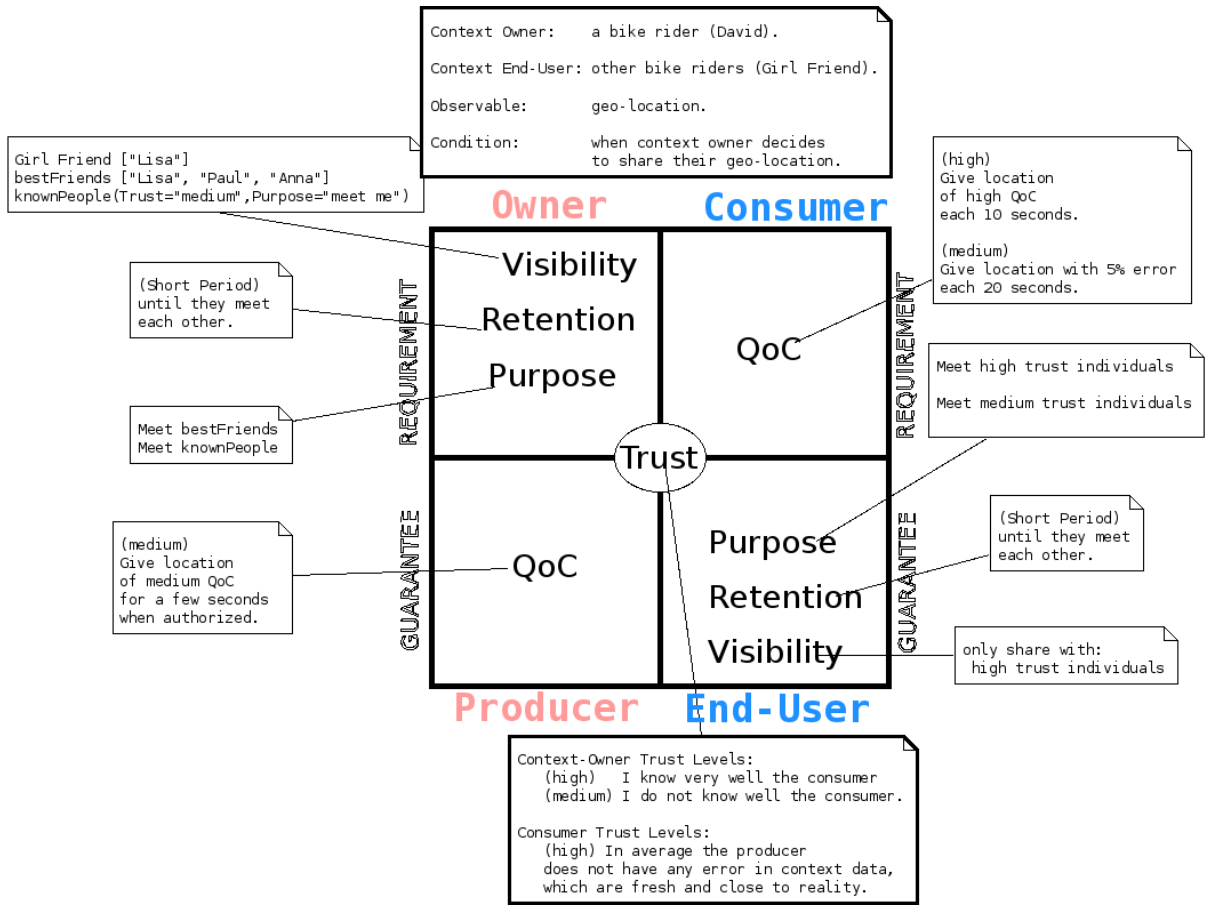


Figure 5.4: Context Contract Dimensions

square (down-right) represents the guarantees in terms of privacy offered by the end-user and the context consumer (e.g., to store location until the purpose is accomplished).

The trust is the glue among these different requirements and guarantees. In other words, because context producers and context consumers do not know each other, they use trust as a mechanism to establish the conditions in which both are willing to sign a context contract. We can see symmetry between the parties involved in a contract. That is to say, for all privacy requirements there exist some guarantees indicating what needs can be fulfilled. And, for all QoC requirements there exist some guarantees that fulfill those requirements.

5.3 MUCONTEXT Contract Meta-Model

In this section, we describe the meta-model we propose for the definition of context contracts in Section 5.3.1. Then, we present the MUCONTEXT contract for producers and consumers in Sections 5.3.2 and 5.3.3. Next, in Section 5.3.4, we detail the PrivacyTerm and QoCTerm classes that specify the contractual limits and obligations of each party in the MUCONTEXT contract. Finally, we present the integration of the QoCTerm class with the QOCIM meta-model in Section 5.3.5.

5.3.1 Definitions

The meta-model specifies the different concepts necessary to define MUCONTEXT contracts. The context contract meta-model is shown in Figure 5.5 and the clause meta-model is depicted in Figure 5.6.

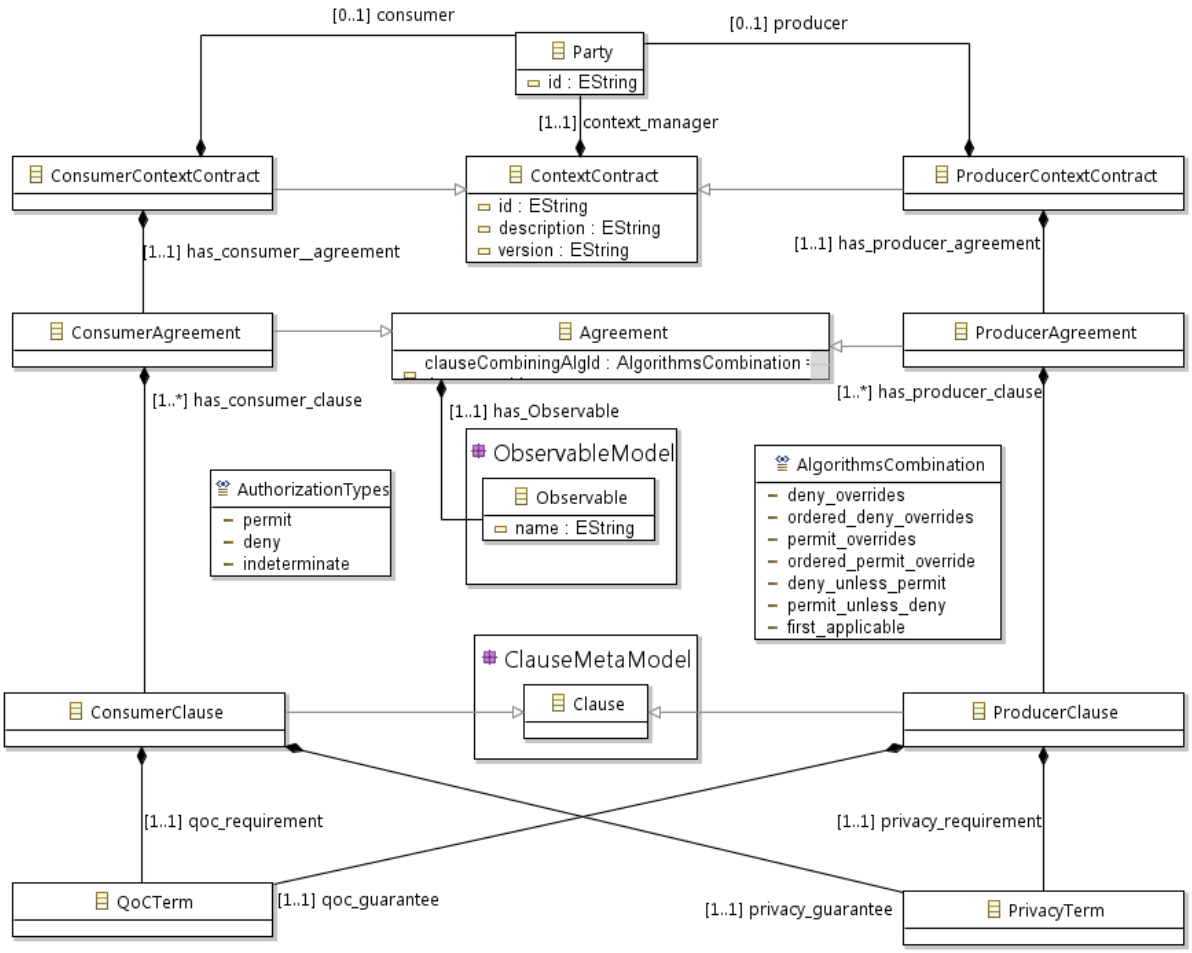


Figure 5.5: Context Contract Meta-Model

A consumer context contract and a producer context contract establish agreements between context producers and context consumers to give access to context data respecting the precepts of privacy and guaranteeing a good QoC. The peculiarity of a context contract is that the parties enter into it voluntarily without prior knowledge of each other. A context contract has two parties and at least one agreement. One party represents either the context producer or the context consumer and the other party is the context manager(context_manager relation), which handles the agreement negotiation representing the counterpart in the contract.

An agreement specifies one observable (abstraction which defines something to watch over (observe) [Taconet and Kazi-Aoul, 2010]) on which the terms of the contract will be applied. The agreement also contains one or several clauses described in the Clause meta-model, which are composed of general conditions and one trust condition. These conditions state the events that trigger the actions that must or must not be executed according to the terms of the contract. These conditions are expressed in the form of predicates.

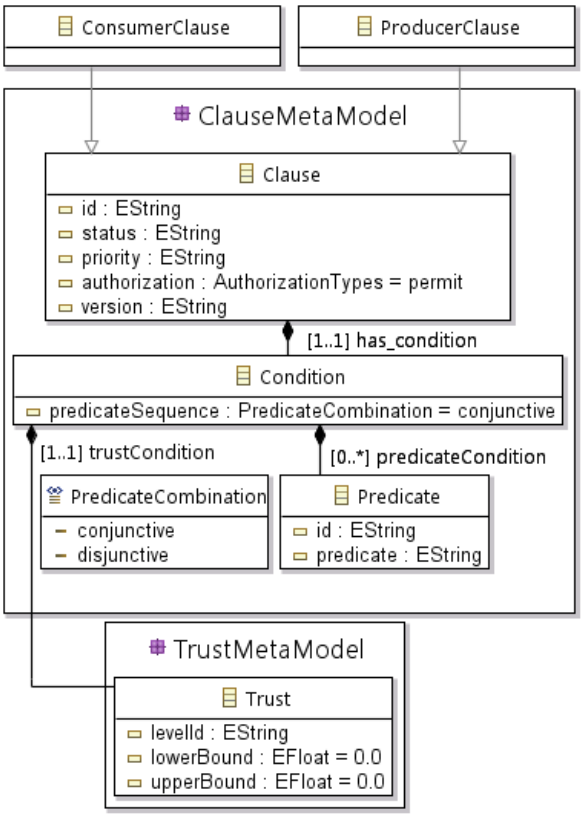


Figure 5.6: Clause Meta-Model

Figure 5.6 depicts the Clause meta-model. The Clause meta-class has a condition which is evaluated according to the predicateSequence attribute, which can take two values conjunctive or disjunc-

tive. A conjunctive sequence indicates that predicates are combined using the logical AND operation. And, a disjunctive sequence indicates that predicates are combined using the logical OR operation.

A Condition meta-class is composed by one trustCondition and a set of predicates. The trustCondition defines the trust level (represented here by a package) a consumer must have to be granted to consume context data. The set of predicates defines the context condition, that determine whether context data are shared or not. This feature allows dynamic adaptation to environment situations given by context data evaluation. For example, we can determine to share our location (context data) only in case of health problem: For example only if our blood pressure and body temperature (context data) are under the normal values during a certain period of time.

We can setup, in the Clause meta-class, the evaluation behavior of clauses through the following attributes:

- The clauseAuthorization attribute defines the authorization decision, which is the result of evaluating clause's Condition. If the clause is applicable then a function evaluates to Permit, Deny or Indeterminate according to the value of this attribute.
- The clausePriority defines the evaluation order of the different clauses. For instance, if the clause-combining algorithm is setup to First-applicable the result will change according to the evaluation order.
- The clauseStatus indicates if the clause is active or not. That is, if a clause is deactivated, it is not evaluated.

A term specifies the contractual limits and obligations that must be followed by each party. A term is composed of two parts: a requirement and a guarantee. The requirement describes properties and activities performed by a party over context data (observations), constraining what the other party must fulfill. The guarantee is the promise made by one party to respect the requirements of the other one under the established conditions. The guarantees represent a risk for the party which accepts it, that is why a guarantee also represents a limit of how much the party is willing to offer. Privacy and QoC terms are detailed in Section 5.3.4.

The main objective of this context contract meta-model is to define the rules to access to context data while taking into account privacy and QoC. Because context owners and context end-users do not know each other, the MUCONTEXT meta-model has been designed for the creation of two types of half-contracts, a producer context contract and a consumer context contract. These half-contracts will be matched at runtime by a context manager. We further detail these two types of contracts in the following subsections.

5.3.2 Producer Contract

A producer contract is composed of the following meta-classes: ProducerContextContract, ProducerAgreement, ProducerClause, PrivacyTerm and QoCTerm (on the right in Figure 5.5)

The ProducerContextContract meta-class is the main meta-class of the contract. It specifies involved parties and setup one or several producer agreements represented by the ProducerAgreement meta-class.

The ProducerContextContract meta-class inherits from the ContextContract meta-class. Thus, the ProducerContextContract involves two parties identified by the roles of context manager and producer. Also, the ProducerContextContract has one or several producer agreements represented by the ProducerAgreement meta-class.

The ProducerAgreement meta-class inherits from the Agreement meta-class which is composed of only one observable following the Observable meta-model represented here by a package (see [Taconet and Kazi-Aoul, 2010] for more details on the Observable meta-model). The ProducerAgreement is composed as well by one or several clauses represented by the ProducerClause class.

We configure in the ProducerAgreement the way ProducerClause evaluates clauses. For that we use the clauseAlgoritm attribute. We borrow from XACML [XACMLv3, 2013] the concept combining algorithms, which defines a number of combining algorithms that can be setup at clauseAlgoritm attribute. In our case, the clause-combining algorithm defines a procedure for arriving at an authorization decision given the individual results of evaluation of a set of clauses.

We chose three algorithms as examples of standard combining algorithms (see [XACMLv3, 2013] for a full list of standard combining algorithms), because, they are easy to understand and implement. Furthermore, they are already developed by the XACML 3.0 implementation. These algorithms are:

- Deny-overrides (Ordered and Unordered),
- Permit-overrides (Ordered and Unordered) and
- First-applicable

In the case of the Deny-overrides algorithm, if a single clause is encountered that evaluates to Deny, then, regardless of the evaluation result of the other clauses in the agreement, the combined result is Deny. Likewise, in the case of the Permit-overrides algorithm, if a single Permit result is encountered, then the combined result is Permit. In the case of the First-applicable combining algorithm, the combined result is the same as the result of evaluating the first applicable clause in the set of clauses [XACMLv3, 2013].

The ProducerClause meta-class inherits from the Clause meta-class. Additionally, The ProducerClause contains two terms: one to set up the privacy requirements represented by the PrivacyTerm class and another one to set up the QoC guarantees represented by the QoCTerm class. These two classes are explained further in Section 5.3.4.

5.3.3 Consumer Contract

The purpose of a context contract is to ensure that the context consumer will receive the appropriate QoC from trustworthy context producers. And for that, is willing to offer some privacy guarantees. For instance, the trustCondition defines the trust level (represented here by a package) a producer must have to accept to consume context data from it.

A consumer context contract meta-model differs on three aspects from a producer context contract. First, the prefix of each meta-class changes from Producer to Consumer. Accordingly, a consumer context contract is composed of the following classes: ConsumerContextContract, ConsumerAgreement, ConsumerClause, PrivacyTerm and QoCTerm. Secondly, the parties involved in a consumer

contract possess the manager and consumer roles. And finally, the ConsumerClause composition roles are reversed compared to the ProducerClause class. That is to say the QoCTerm class sets up the QoC requirements and the PrivacyTerm class sets up the privacy guarantees.

These two classes are explained in detail in Section 5.3.4.

5.3.4 PrivacyTerm and QoCTerm classes

As we mentioned before, a term specifies the contractual limits and obligations that must be accomplished by each party. In our context contract meta-model, a term can act as a requirement or a guarantee. So the models behind the PrivacyTerm and QoCTerm classes should take this into account as described in this section.

PrivacyTerm Class

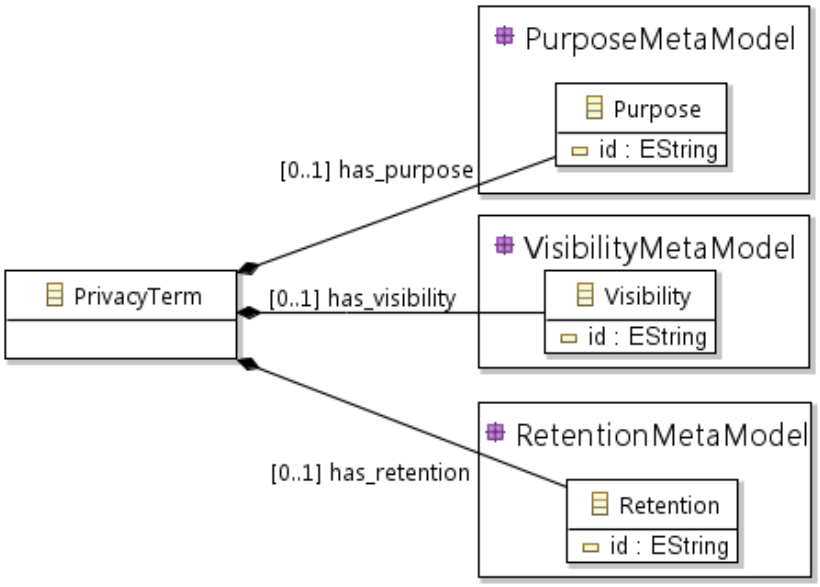


Figure 5.7: Privacy Term Meta-Model

The PrivacyTerm meta-class is composed of all the elements that define the privacy. As a requirement, these elements define the access restrictions and actions to be taken when a context data is accessed. As a guarantee, the PrivacyTerm meta-class expresses the way in which context data will be accessed and for what purpose.

Four meta-models enable designers to describe privacy and QoC. They are detailed in Section 5.4. In Figure 5.7, these meta-models are represented by the corresponding packages showing their main inner classes and the different relationships that exist among these classes and the classes of the ContextContract meta-model. For instance, when a context owner defines the visibility as a requirement

he/she indicates who has the right to access the context data. In the case of a location-aware application, he/she can specify that only those entities with a high trust level may access his/her geo-location. On the other side, when a context consumer guarantees a visibility, it is telling that it guarantees to fulfill the credentials necessary to access the context data.

QoCTerm Class

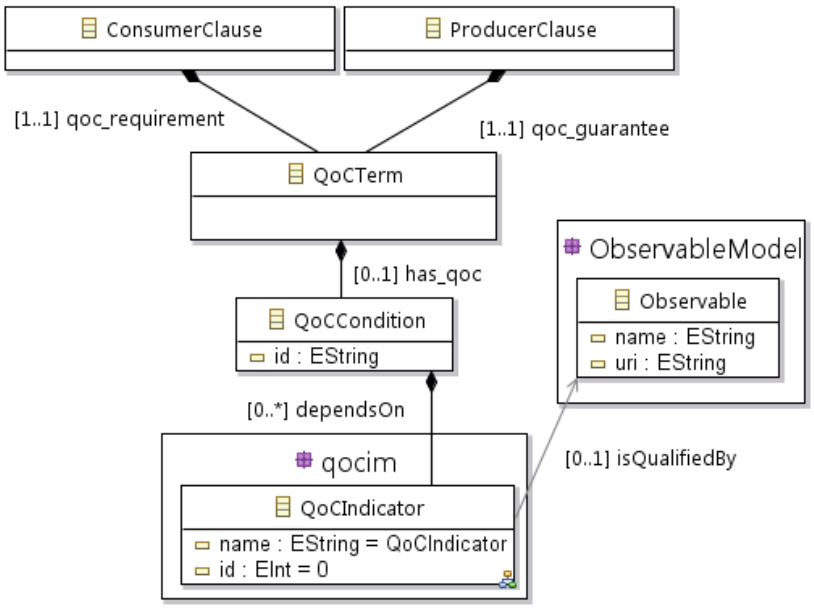


Figure 5.8: QoC Requirement and Guarantee Meta-Model

The QoCTerm class is composed of all the elements that define the QoC, as modeled in QOCIM (see Section 5.4.4 and Figure 2.3). In Figure 5.8, QOCIMis represented by the QOCIM package showing its main inner class and the different relationships that exist with the other classes of the ContextContract meta-model.

When considered as a guarantee, the QoCTerm class expresses the way in which context data will be delivered by context producers. For instance, a context producer can indicate that only those context consumers with a high trust level may access its geo-location information and that it guarantees a margin of error around 5% but those with a low trust level may access its geo-location with a guarantee of 30% of error.

On the side of context consumers, QoC requirements are expressed. A context consumer can therefore express that it needs a certain QoC level to make its calculation or run its process without error. On the same example of a location-aware application, a context consumer may state that it will be able to satisfy the context end-users expectations with no more than 5% of error margin.

5.3.5 Integration of the QoCTerm class with QOCIM

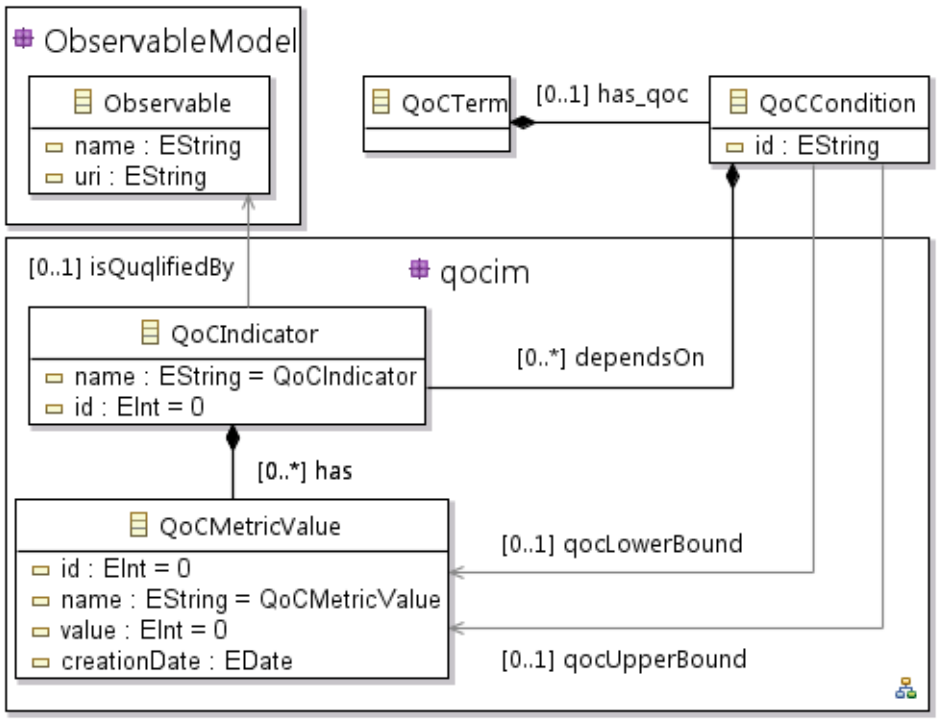


Figure 5.9: Integration of the clause model with QOCIM

Figure 5.9 presents how the QoCTerm class is integrated with QOCIM via the intermediate of the class QoCCondition.

The QoCTerm class comes from the context contract meta-model presented in Figure 5.5. The QoCIndicator class comes from QOCIM as shown in Figure 2.3. Figure 5.10 shows the class model defined for the example of the use of QOCIM to express the QoC requirements of a context producer on the precision QoC criterion. The context consumer instantiates the QoCMetricValue class to specify its required lower and upper bounds.

qocLowerBound and qocUpperBound associations, in the class QoCCondition, are specified with the id attribute of the QoCMetricValue meta-class.

In this example, there are two QoCMetricValue instances, urn:qocim:metric-value:id:precision-value:LB and urn:qocim:metric-value:id:precision-value:UB which respectively define the lower and the upper required values for the precision criterion.

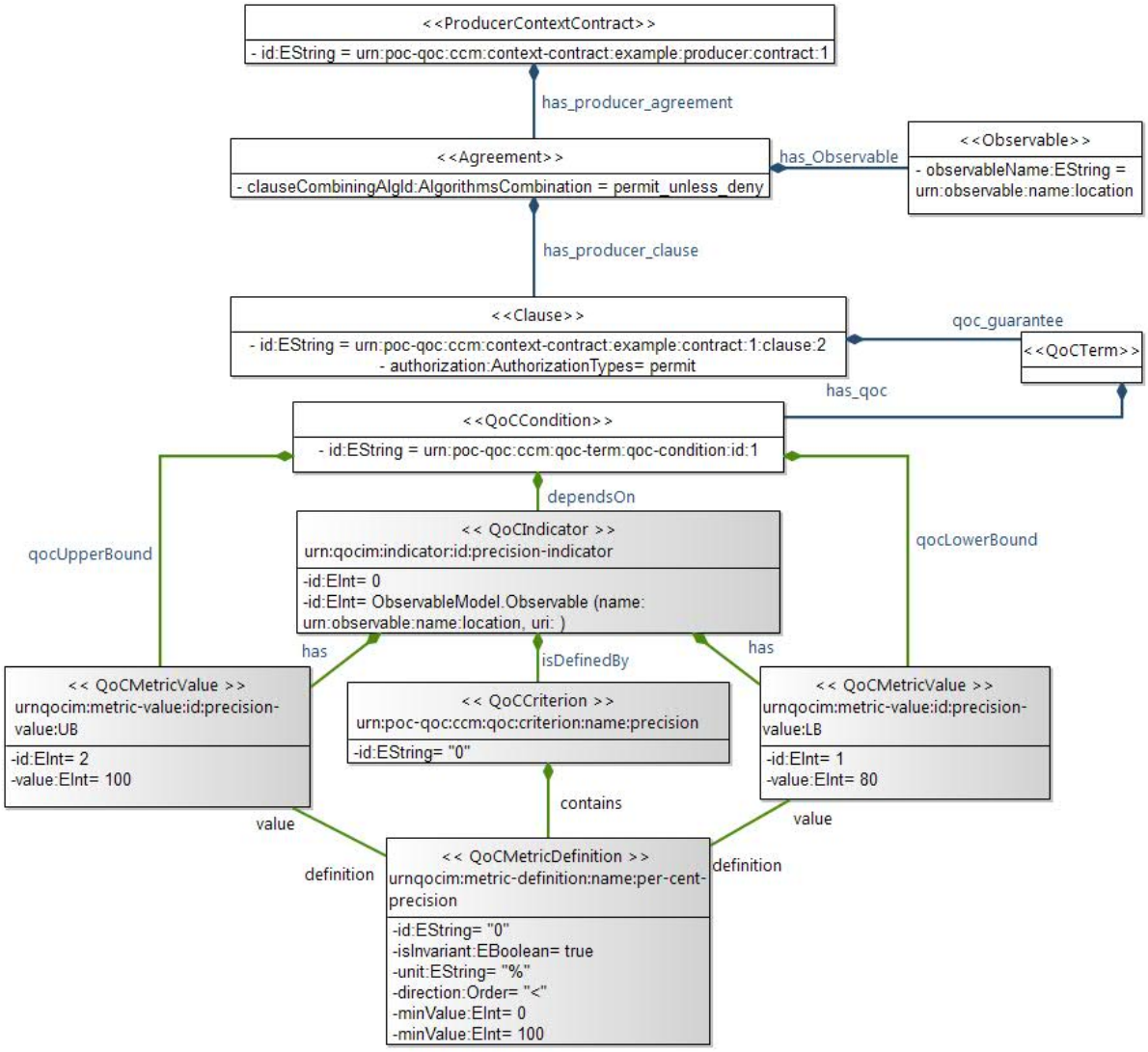


Figure 5.10: QoCIM based precision criterion model used as a context consumer clause

5.4 Privacy and QoC Protection Meta-Models

This section describes the meta-models associated to the fundamental protection concepts (i.e., purpose, visibility and retention) and the QoC concepts.

We describe the proposed protection meta-models for context data with respect to the dimensions of purpose (Section 5.4.1), visibility (Section 5.4.2) and retention (Section 5.4.3).

For the QoC, we use the QoC model presented in [Marie et al., 2013a]. Section 5.4.4 explains the reasons of this choice. We now describe the four corresponding basic meta-models.

5.4.1 Purpose Meta-model

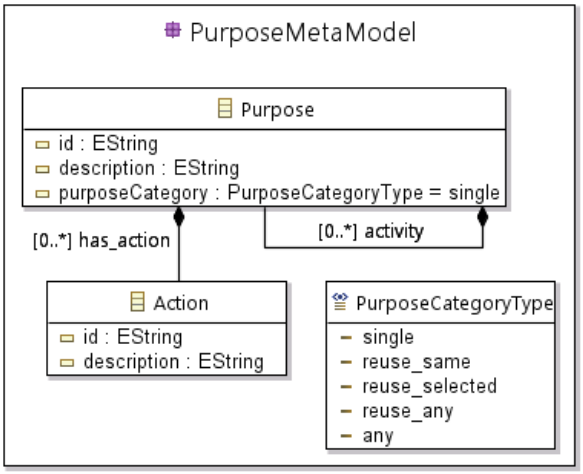


Figure 5.11: Purpose Meta-Model

The purpose meta-model shown in Figure 5.11 describes the intended use of context data.

The Purpose meta-class represents the intentions or the reasons for which context consumers or context end-users need context data.

The intentions are made of actions (Action meta-class) that are performed over the context data such as consulting, transforming, publishing. A set of actions defines a purpose. Additionally, one purpose can be composed of other purposes through the composition activity. The objective of the use of activities is to simplify, organize and reuse purposes. Activities are used to put together some purposes to which the same privacy rules apply.

Summarizing, a purpose achieves a goal. A purpose has activities, that represent a task that is achieved through a set of actions.

Purpose Tree

Using this model to describe and organize the purposes, we create a recursive tree, where purposes are the vertex and leaves of the tree. Figure 5.12 presents an example of purpose tree.

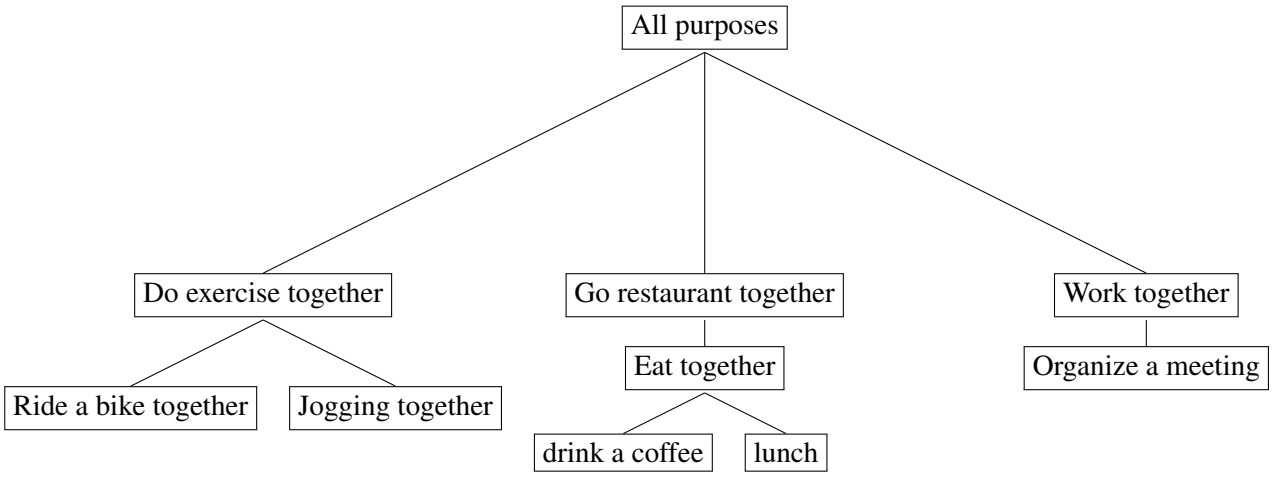


Figure 5.12: Purpose Example Tree

This example shows, how purposes are related with other purposes. The tree root includes all purposes, then each node is a purpose composed by other purposes. Thus, the same privacy rule applies to all the children of a purpose. However, a rule does not apply over the parent of a purpose.

5.4.2 Visibility Meta-model

The visibility meta-model describes how to organize context end-users and their respective access levels. Figure 5.13 describes the Visibility meta-class which represents the different circles of visibility that context information can have.

A VisibilityCircle groups several members associated to a visibility. Members can be persons or organizations or software entities,— i.e. context end-users, or context-aware applications — and are modeled in classes. The VisibilityCircle concept is motivated by the aim of grouping members based on their common objectives, or because the same set of rules apply to them.

These groups can be defined in two ways: automatically or manually. In the automatic way, they are dynamic and ad-hoc groupings. Also, they can be formed by filtering processes which identify specific features or functions of context-aware applications in order to decide which visibility circle they belong to. In the manual way, these groups are assigned by context owners according to their privacy preferences.

For privacy concern, the behaviour of context consumers and context end-users is evaluated continuously to determine how much the context owner can trust them. That evaluation could be done in several ways: by recommendation, by ranking, and so on. The result of this evaluation is normalized between 0 and 1, where 0 represents no trust at all, and 1 corresponds to a total confidence in the context consumer.

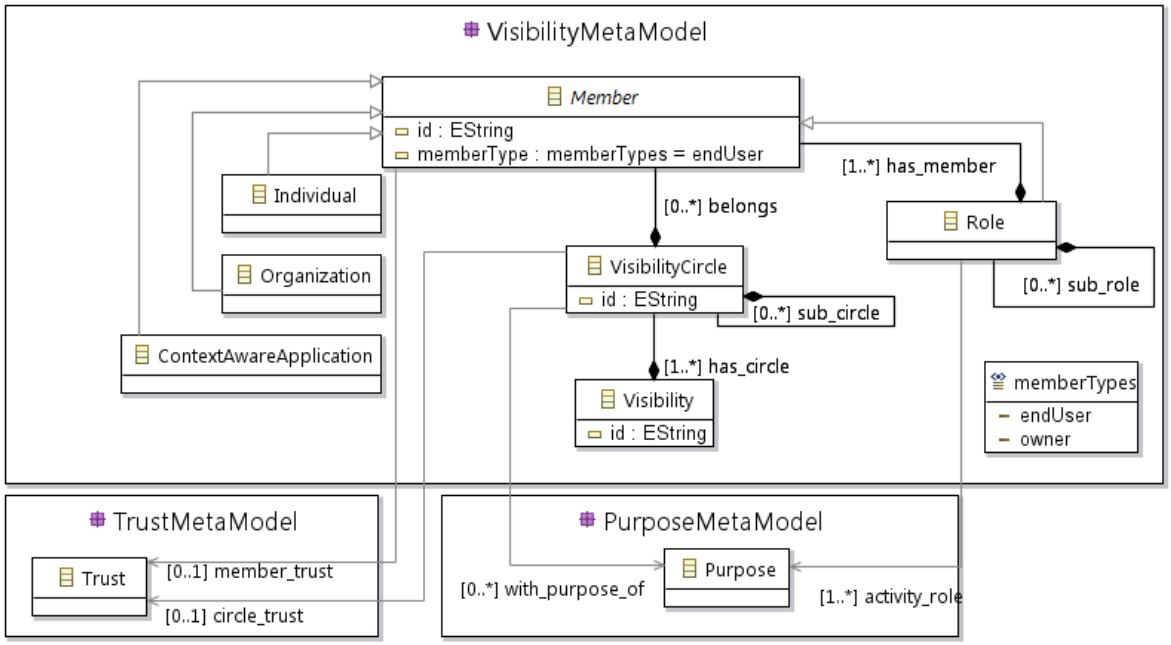


Figure 5.13: Visibility Meta-Model

Visibility circles can be handled dynamically. A visibility circle may be defined by its members or by the properties of its members (trust, purpose). In this latter case, the system will determine at runtime the circle a member belongs to.

The member_trust relationship is used to characterize the ranking of the trust in a member. So, we assume that a value of trust may be assigned to context consumers and context end-users to determine if the context owner can trust them. The circle_trust relationship indicates a minimum trust threshold required for all visibility circle members. Thanks to the member_trust and circle_trust relationships, visibility circles can be created dynamically.

In this document, we focus on how trust is used and not on how it is evaluated. Therefore, we assume that an initial value of trust for context consumers and context end-users is extracted from an external system like a social network.

Purpose also plays a role in the creation of visibility circles. The with_purpose_of relationship establishes the intentions that have all the members of a visibility circle. In this way, all the individuals, organizations, etc, which declare purposes that match with all those indicated with this relationship and reach the appropriate trust level can become members of a visibility circle. If the purpose or trust level of a member changes, it can be expelled immediately from the visibility circle.

For instance, when a context owner wants to install an application on his/her smart phone, he/she considers the declared permissions required by the application over the phone's resources (user location, network communication, WI-FI connections, storage, other applications information, contracts, etc). With this information, a context owner can estimate how invasive the smart phone application is.

We identify different types of members: Individual, Organization, Context-aware Application and Role. The Individual meta-class represents a person or a thing. The Organization meta-class represents " those with formal and informal commitments required by their entity membership " [Skinner et al., 2006b]. A Role groups a set of members that play a specific function. For example, a patient agrees to share his medical records with doctors or nurses. Here, Doctor and Nurse are roles with a specific function. Thereby, a role can be seen as an organizational position. According to [Cuppens and Miège, 2003], the concept of role is a really convenient manner to structure entities because it is a quite intuitive notion, and it enables the establishment of a relevant matching between the organization's structure and access control policies. Finally, roles and visibility circles can be composed of other roles or visibility circles simplifying the administration and reuse of these groups. For example, the organizations "Insurance XYZ" and "Insurance ABC" are insurance companies; the Insurance role is created to group such kind of companies.

Visibility Tree

As we did with purposes, the visibility circles can be organized into a recursive tree structure, where visibility circles are the vertex and leaves of the tree. A visibility circles can be formed by one or several sub visibility circles. A visibility circle identifier is used to tag context consumer or end-users. Each node has a unique identifier. By definition, each context owner may have its own visibility classification of consumers or end users.

We assume that should exist a common visibility tree shared between consumers and producers from which context owners start to personalize their own visibility circles. Context owners's new visibility circles will create new branches and leaves under the common visibility tree.

If the common visibility tree is extended each context owner can update its version easily.

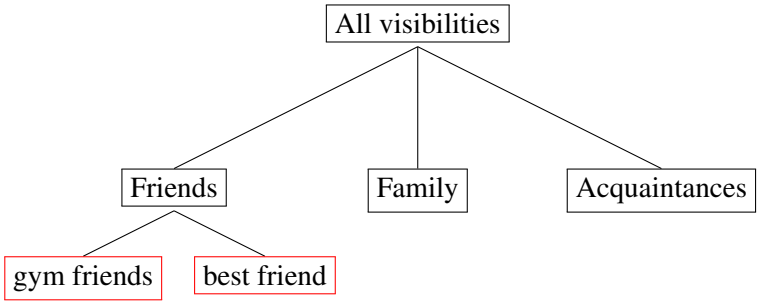


Figure 5.14: Visibility Tree Example

For example, in Figure 5.14 the black boxes represent the common visibility tree, and red boxes the customization by a context owner.

A visibility circle node shares the members of its children. The rules that apply to a visibility node also applies for its parents but not to its siblings. On the contrary, the context owner is restricted to use only the privacy rule defined for the common visibility tree.

For instance in our example 5.14 we can define that "best friend" only will have access to context data for "eat together" purpose. Thus, we can eat with "friends" but not with the "gym friends".

5.4.3 Retention Meta-model

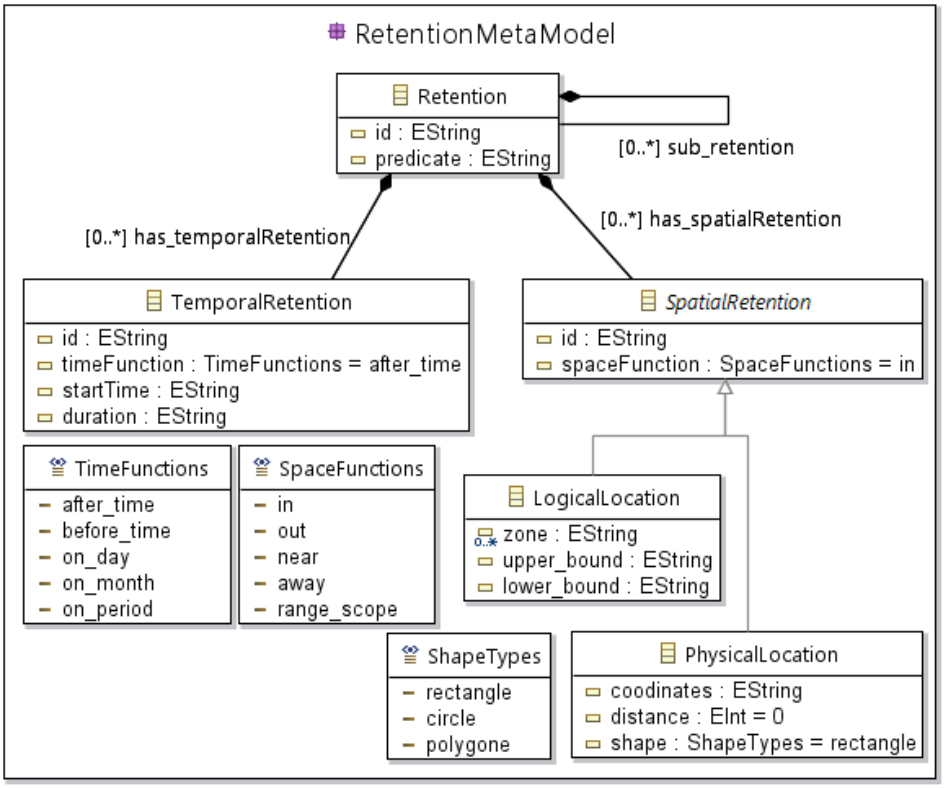


Figure 5.15: Retention Meta-model

The retention meta-model (Figure 5.15) describes the conditions to define the retention period of context data. The way in which we define Retention is inspired from the OrBAC context definition [Cuppens and Miège, 2003], in which the authors define a taxonomy of conditions in access control rules. We use the same retention concepts to determine the conditions in which context data should be stored on the consumer side and when they should be deleted. There are two main types of retention namely temporal and spatial which are detailed below.

Temporal

With a temporal retention condition, it is possible to express that some given context data are authorized to be logged on the consumer side only for a given time interval. The temporal conditions can correspond to a maximum storage duration but also to a fixed day of the week, to a time of the day, etc.

For instance, if we want to specify that information should be deleted every monday morning, we create an instance of retention with the following predicate "Monday & Morning". This example is

shown in Figure 5.16 where each predicate element refers to a temporal retention instance. Through the time functions, we define the interval of time in which the deletion should take place.

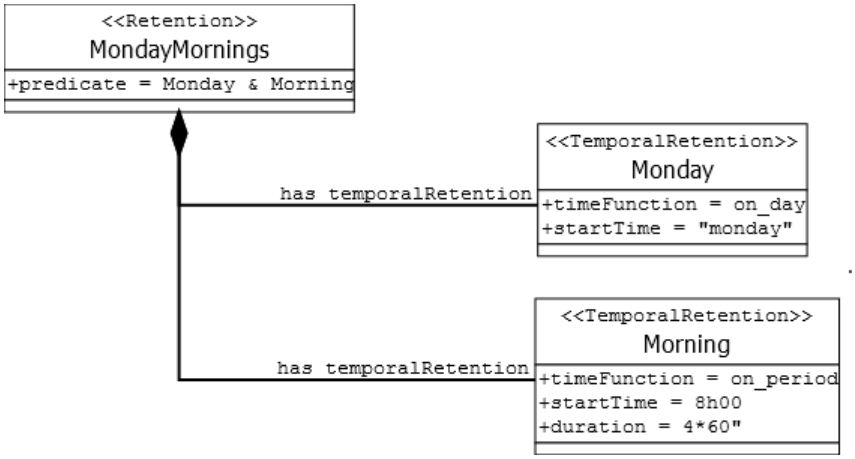


Figure 5.16: Temporal Retention Example

Spatial

Knowing the location from where the context data are taken and/or the location of the context owner, it is possible to specify in which spatial areas the context data should be stored. For instance, a context owner may decide to provide only context data to close computing devices.

[Cuppens and Miège, 2003] distinguishes two different types of spatial information: physical and logical spatial information. Physical spatial information corresponds to the physical location of the user, namely his office, a security area, a specific building, the country, etc. Validation of such a context may rely on GPS modules, employee's badge tracking, etc. A logical spatial context corresponds to the "logical location" the user stands in, for example, it can be the computer, the network or the sub-network, the cell in the case of radio communication such as in UMTS, etc.

In the case of spatial retention, as soon as the context owner goes in or out of a specific area, the context data should be deleted. So, with this model we can specify three types of areas: squares, circles, and polygons. Coordinates will be interpreted according to the shape. For example, for a "circle", the first value indicates the center and the second the radius. For a logical location, the zone attribute indicates a set of cell identifiers corresponding to the 3G phone network cells detected by the user phone; this delimits the area from which the context owner is connected.
In the case of mobile context owners and mobile context consumers, a scope area (attribute range_scope) around them is defined. When both areas are intersected, context data may be captured and stored. So, as soon as both areas are disjoined, context data should be removed. For example, two persons that share their location until a "top secret" meeting. As soon as the meeting is over, they should delete the location information of the other person.

Retention Tree

It is important that both producers and consumers have a common reference of retention definitions. Thanks to the composition "sub_retention" we can describe and organize the retention into recursive tree structure, where a retention definitions are the leaves and vertices of the tree.

A retention node is grater than its children. For example, one hour retention may contain 10 minutes retention. Similarly, a country area contain state areas, and state areas contain city areas.

An example of temporal retention tree could be (see Figure 5.17):

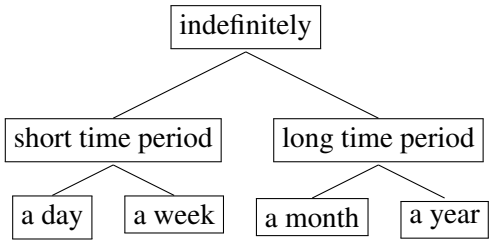


Figure 5.17: Retention Example Tree

5.4.4 QoC dimension

Granularity is implicitly related to the notion of precision or level of detail as in [Anciaux et al., 2006] and is often associated to obfuscation techniques where more or less details may be provided on some given information [Wishart et al., 2005, Chakraborty et al., 2012]. Regarding context data, we consider that the term of granularity imposes some limitation on the kind of privacy mechanisms that could be put in place and we rather manipulate the level of quality of the context data. The concept of QoC is introduced in the INCOME deliverable 2.1 [INCOME, 2013a] and QoC models are analysed in the Chapter 2 of this document.

QoC is more general than granularity and may actually subsume it. Various QoC criteria can be associated to context data and offer as many opportunities to adjust the quality level of the context data provided. This allows to define rich obfuscation solutions for the manipulation of context data at the appropriate QoC level to protect the privacy of the context owner.

Marie et al. [Marie et al., 2013a] proposed the QoCIM meta-model as a unified solution to model heterogeneous QoC criteria, after the analysis of several existing QoC models. We rely on QoCIM in our solution as it offers a generic, computable and expressive solution to handle and exploit any QoC criterion within distributed context managers and context-aware applications. QoCIM defines the concept of QoC indicator to associate a QoC criterion to a metric value. A QoC indicator can then be compared to a given QoC level. Figure 2.3 show the QoCIM meta-model.

"QoCIM qualifies Observations with the QoCIndicator meta-class. An indicator is a container of values, QoCMetricValue meta-class, which belong to the same QoC indicator. An indicator is identified with the attribute id. A value, QoCMetricValue meta-class, is also identified with an attribute id. QoCIM offers to store the valuation of the QoCMetricValue meta-class into the attribute value. The date of creation of the value of the indicator is stored into the attribute creationDate. The attributes

of the QoCMetricDefinition meta-class define the production of QoCMetricValues. The Description meta-class brings semantics for the QoCMetricDefinition meta-class. And, the last element allowing to build composite criterion the recursive link set on the QoCMetricDefinition meta-class. This link supports the ability to model and use a criterion that is based on other criteria. Therefore, QOCIM allows QoCMetricDefinition depending on other classes QoCMetricDefinition." [Marie et al., 2013a]

We have detailed the way we use the QOCIM model in MUCONTEXT contracts in Section 5.3.4.

5.5 MuContext Contract Meta-Model Validation

In this section, we validate the MUCONTEXT contract meta-model vocabulary expressiveness and applicability to describe the terms of privacy and QoC rules.

The remainder of this section is divided as follows. Firstly, we introduce a use case through which we perform a qualitative validation in Section 5.5.1. Secondly, Section 5.5.2 presents the editor created to model the MUCONTEXT contracts. Then, we present in Section 5.5.3 an extract of the MUCONTEXT contracts created to examplify the applicability, the expressiveness and usefulness of the MUCONTEXT contract meta-model. Finally, in Section 5.5.4, we evaluate how the MUCONTEXT contracts respond to the expectations of the requirements presented in Section 4.3.

5.5.1 Validation through Bike4All Use Case

All models are, by nature, incomplete representations of the system they are intended to model. The first validation of a meta-model consists in demonstrating its applicability to model several use cases. We validate MUCONTEXT contract meta-model vocabulary expressiveness and applicability to describe context situations, privacy rules and QoC.

To validate our model, we follow a conceptual approach. A conceptual validation concerns the question of whether the model accurately represents the system under study. That is to say, are the model assumptions credible? The conceptual validation is qualitative. We present our analysis based on the observations made by applying the MUCONTEXT contract model to the application scenario presented in Section 4.2. This scenario was also presented in the INCOME project [INCOME, 2013b].

Figure 5.18 depicts a use case of the mentioned motivating scenario. This use case is described as follows:

David uses the application Bike4all to join people when he is biking. To protect his private life, he establishes a set of rules to define in which context situations he shares his GPS location. The system is capable of recognizing several context situations like: the owner has an emergency, he is riding a bike, he is going home, he is working or he is doing exercise. For some combinations of context situations, David has established privacy rules. First of all, David does not let strangers know where he lives. Moreover, David does not like the feeling of being watched. Therefore, he only shares his location with the city EMS when strictly necessary (in an emergency situation). Also, he agrees that, in case of emergency, volunteers with a high level of confidence assist him. To define who has access to his location he has established, from his telephone contact list, several groups of visibility, for instance, Friends and Acquaintances.

Figure 5.18 shows three different context situations in which David's location is delivered or not to the context end-users. The figure shows three colored circles. Each circle represents the location accuracy radius given to each visibility group, for example, in the context of when David is going home, Bob receives David's location with a high QoC level, while Joe receives it with a very low QoC level, which makes impossible to localize precisely David. But, in case of an emergency situation, we observe that some unknown but reliable people get David's location.

Next, we present the tool implemented to create the MUCONTEXT contracts used to express this use case scenario.

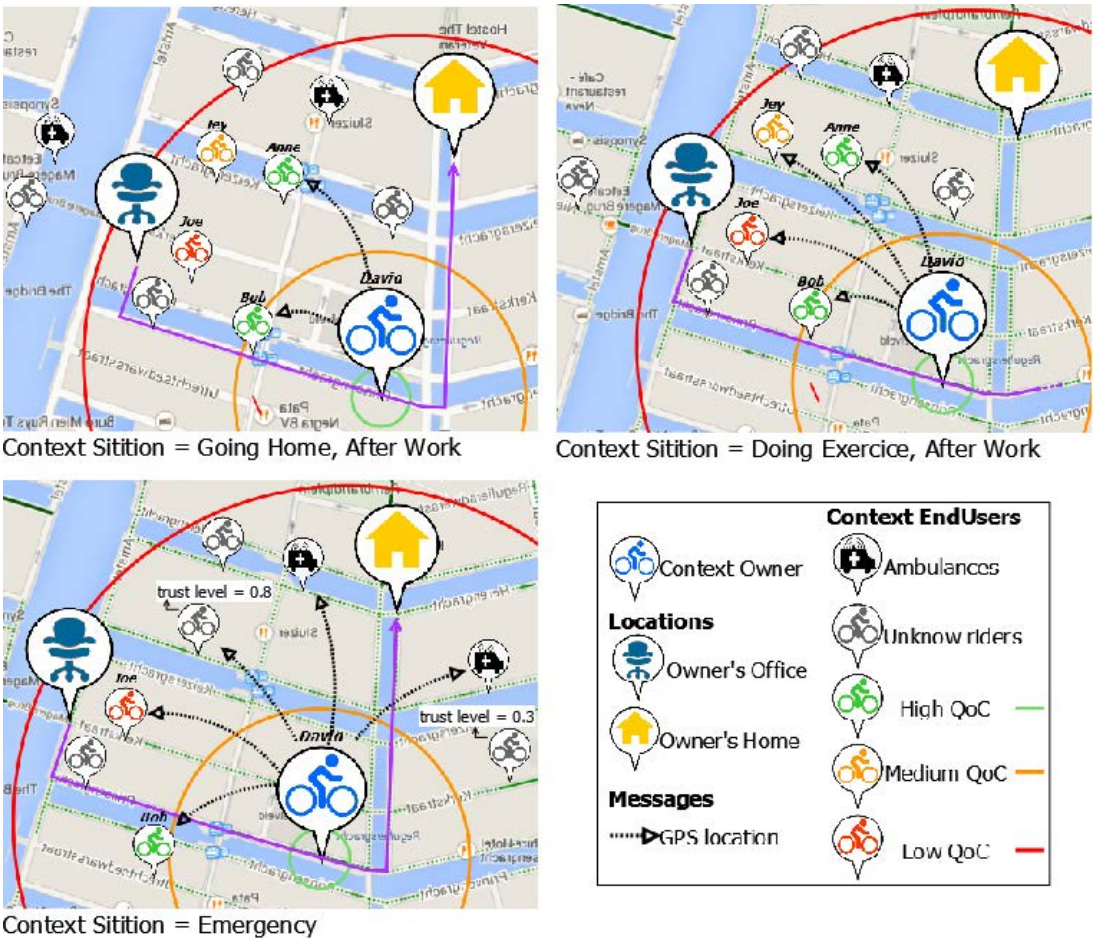


Figure 5.18: Sharing Location in a Social Bike System (Use Case Scenario)

5.5.2 Implementation of MUCONTEXT Contracts

We implemented the MUCONTEXT contract meta-model using the Eclipse Modeling Project Framework (EMF) [Eclipse Foundation, 2010]. The meta-model is defined as an instance of Ecore meta-meta-model. EMF generates a MUCONTEXT contract model editor. We have extended this editor using Obeo Designer [Obeo, 2014b] to create a graphical tool to write producer and consumer MUCONTEXT contracts instances (see Figure 5.19). The MUCONTEXT contract editor is packaged as a plugin for the Eclipse development environment.

All MUCONTEXT contracts that we present in Section 5.5.3 have been defined and validated using the MUCONTEXT contract editor.

In the next section, we present the models of MUCONTEXT contracts produced for the use case presented above.

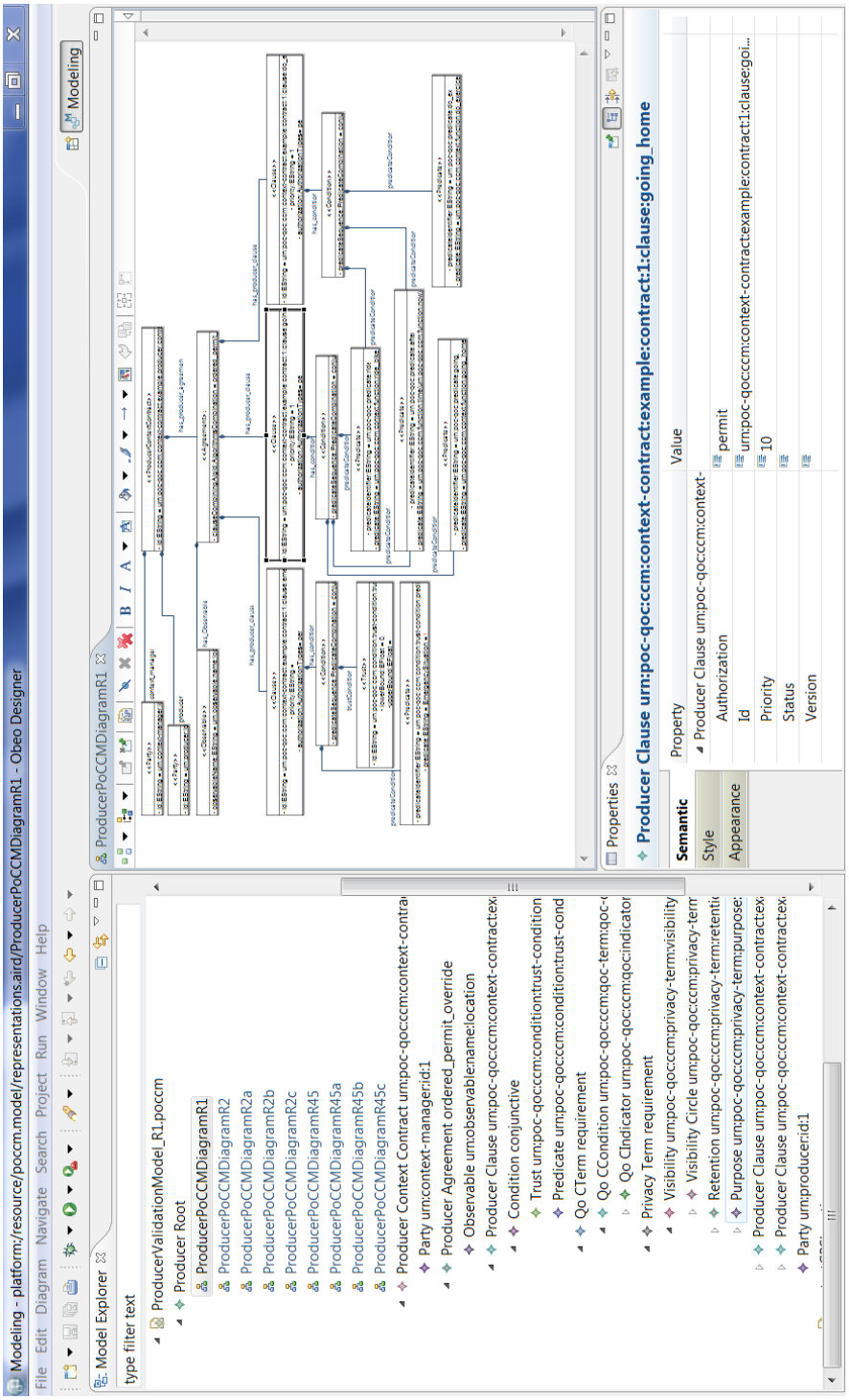


Figure 5.19: MUCONTEXT Contract Model Editor

5.5.3 Producer and Consumer MUCONTEXT Contracts for Bike4All

We start our demonstration with the producer side. The UML diagram in Figure 5.20 represents one part of the producer MUCONTEXT contract that models the example use case scenario presented in the illustration 5.18. This figure shows conditions required to activate each clause.

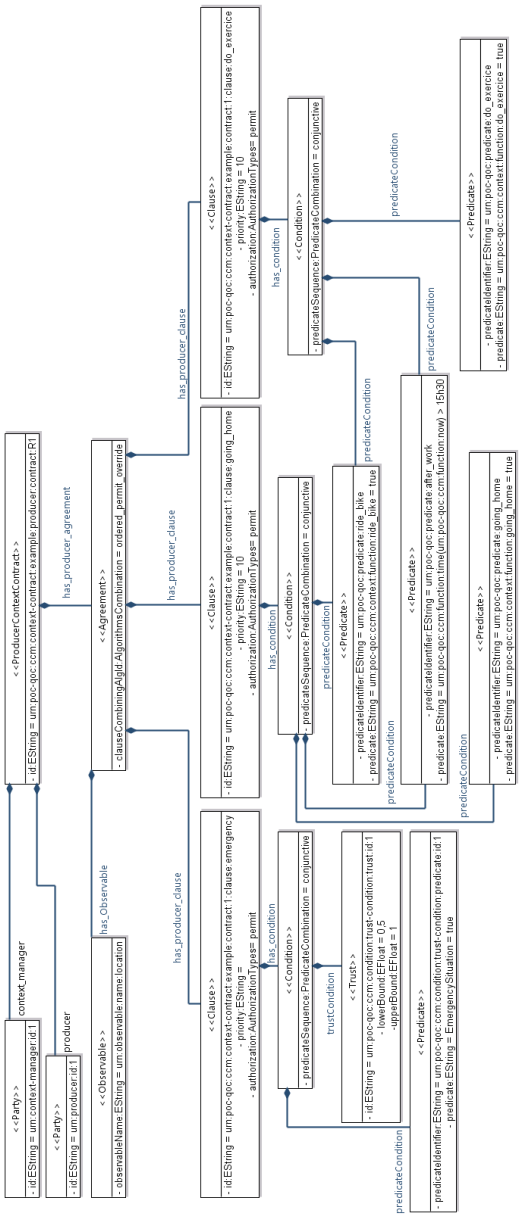


Figure 5.20: Producer MUCONTEXT Contract Diagram - Context Situations

These clauses can share the same contract due to the clause combination algorithms and the clause priority attributes. The property clauseCombinigAlgId in the agreement indicates how the middleware will apply the clauses when several ones are applicable at the same time. In this case, the clause's priority attribute is used to overwrite those with lower priority.

The first clause on the left can be explained in English as "the access to owner's location is permitted, when an emergency situation is detected, and we can trust in the consumer".

The second one, on the middle, means that the consumer can access the owner's location when the following three conditions are met: The owner is riding a bike; it is more than 5:30p.m. and the owner is going home.

The last clause on the right is very similar to the second one with the difference that the owner does not head towards his house, but he is exercising on his bicycle.

However, the access allowed by the three clauses is done under the conditions established in the terms of privacy and QoC established in the same contract. These terms are explained later in this section.

With this example, we show in a use case that the MUCONTEXT contract model is able to express different conditions linked to the user situation or the environment. Moreover, by using the combination algorithms and priorities, it is possible to modify the contract adding or deleting clauses without affecting other clauses.

Figures 5.21, 5.22, and 5.23 show the QoC terms of the clauses presented above. These QoC terms set the QoC level guarantee for the producer. The middleware should enforce the QoC before delivering the location, which means obfuscating the context data values to protect the owner's privacy or to filter the context data to send only those that satisfy the consumer requirements.

The QoC term depicted in Figure 5.21 sets up the QoC level between 0% and 100% because in an emergency any location is better than nothing. The QoC term in Figure 5.22 sets up the QoC level between 1% and 10% to obfuscate the real position of the owner, for instance, the application may deliver the location in a 500 Km radius. Finally, Figure 5.23 shows the QoC for the last clause setting up a high QoC level between 80% and 100% to ensure that the owner will meet his friends.

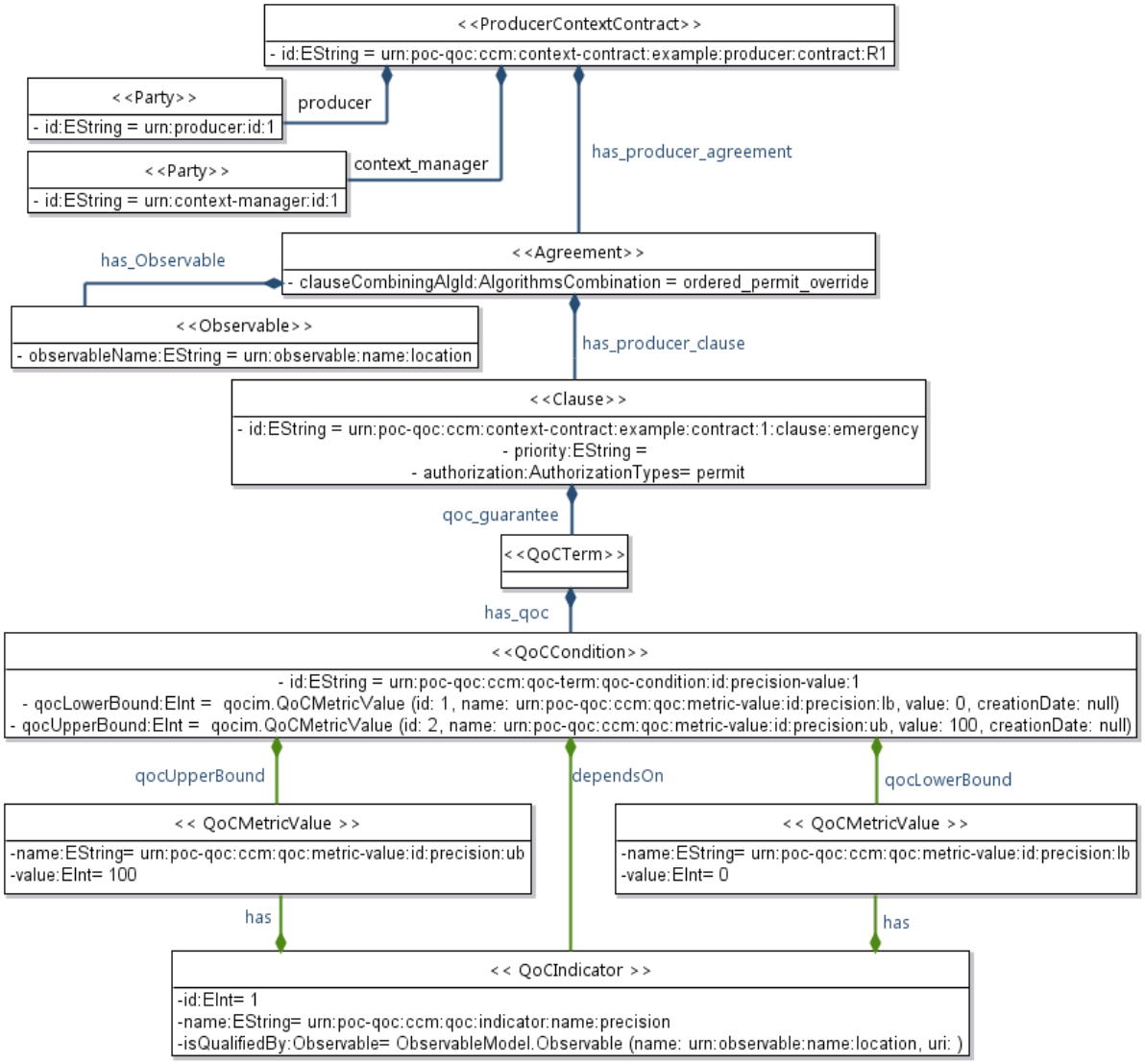


Figure 5.21: Producer MUCONTEXT Contract Diagram - QoC Guarantee (Any QoC)

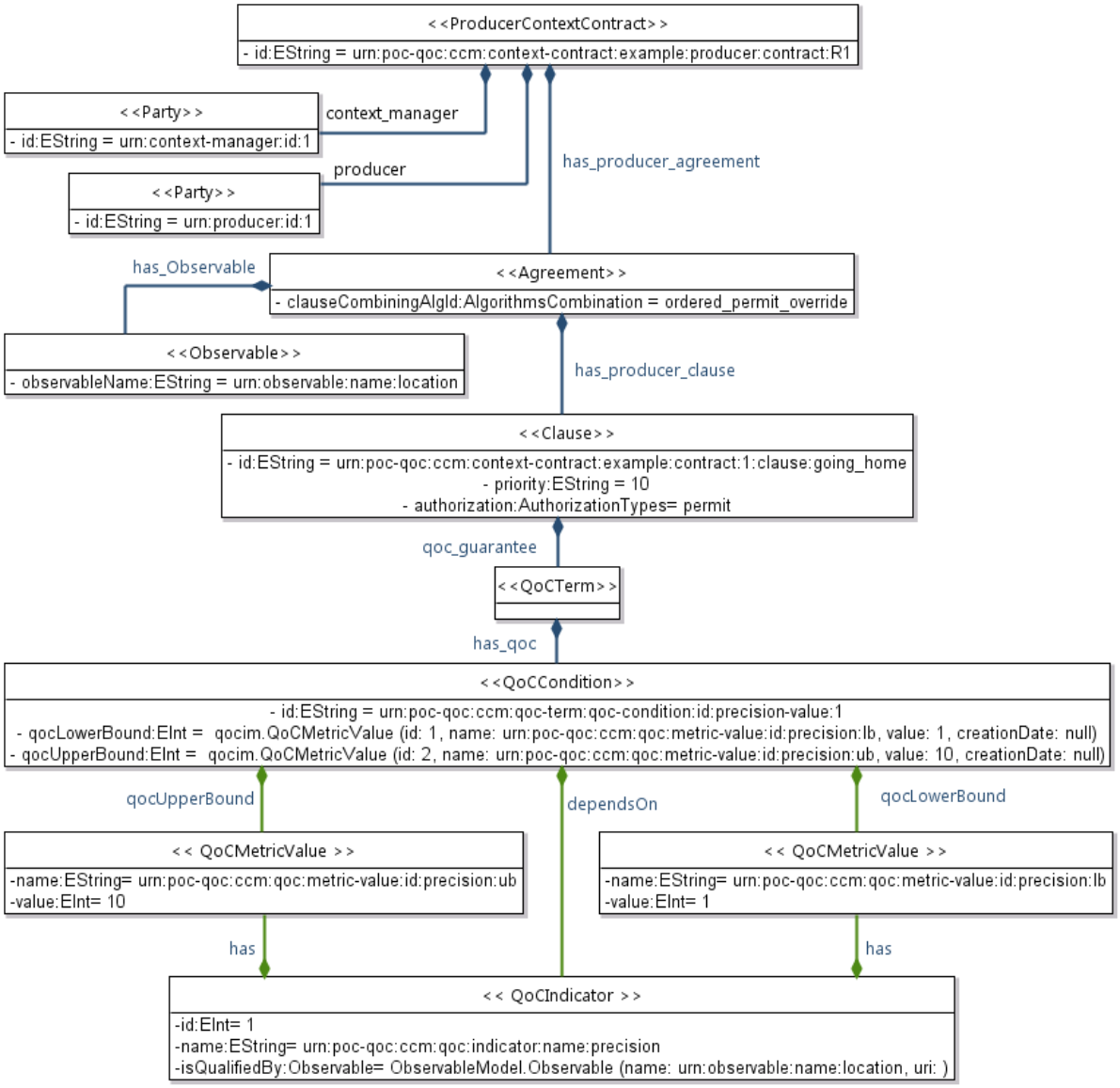


Figure 5.22: Producer MUCONTEXT Contract Diagram - QoC Guarantee (very Low QoC)

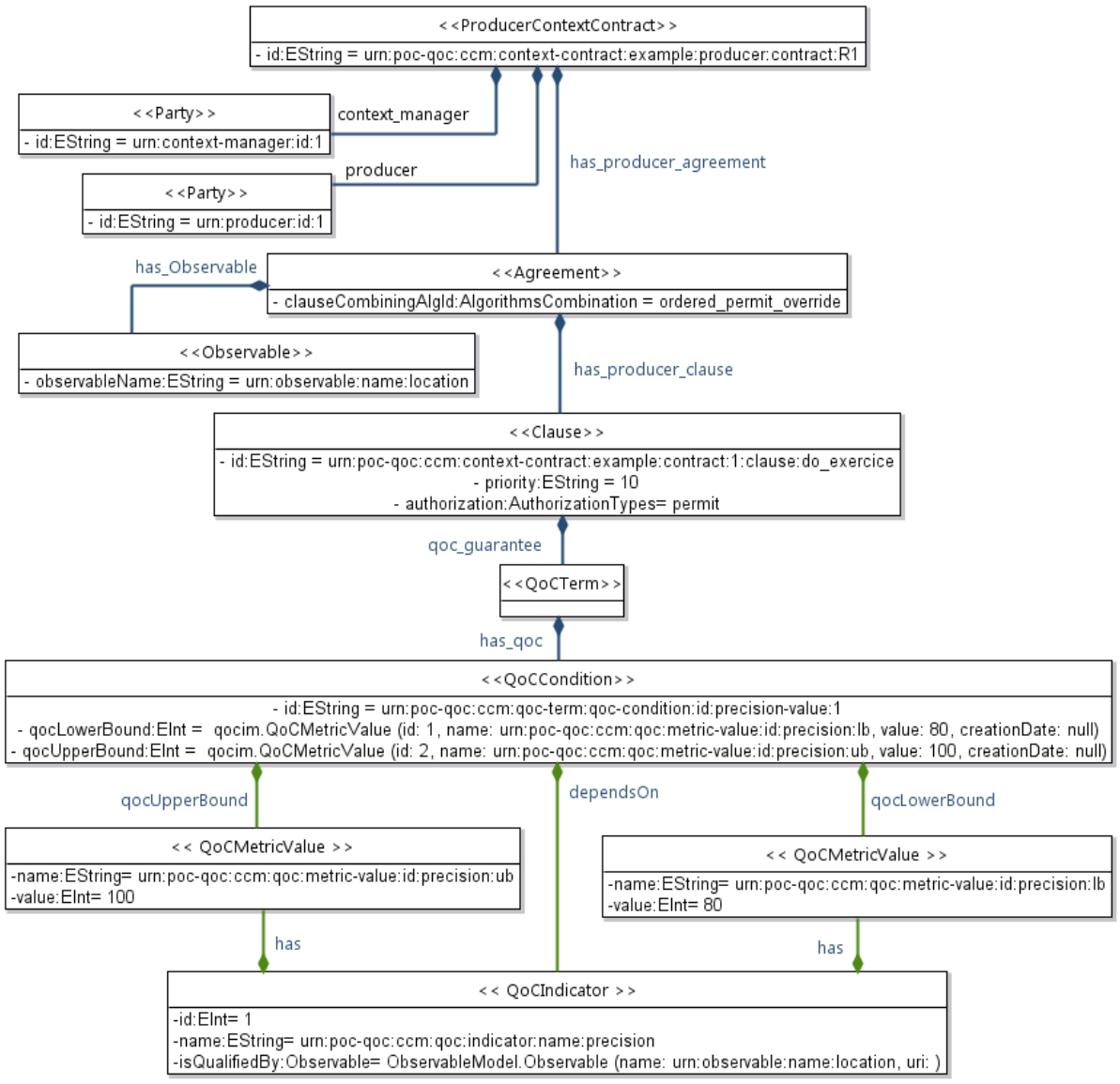


Figure 5.23: Producer MUCONTEXT Contract Diagram - QoC Guarantee (High QoC)

This example denotes how the MUCONTEXT contract model can influence the QoC delivered by the application according to different clause conditions. The use of the QOCIM meta-model, proposed in the INCOME project, proves its effectiveness to describe the QoC [Marie et al., 2013a].

Figures 5.24, 5.25, and 5.26 show the privacy terms of the clauses presented above. These privacy terms set privacy requirements for the context owner. The middleware should allow the owner's location to access only those context consumers that guarantee the respect of these requirements. The privacy term establishes who, how and for how long the access to the owner's location is allowed. For instance,

in Figure 5.24 only trusted groups, which belong to the EMS of the city (e.g., firemen, police officers or trained volunteers) may access the location. To help the owner in case of he/she suffers a heart attack.

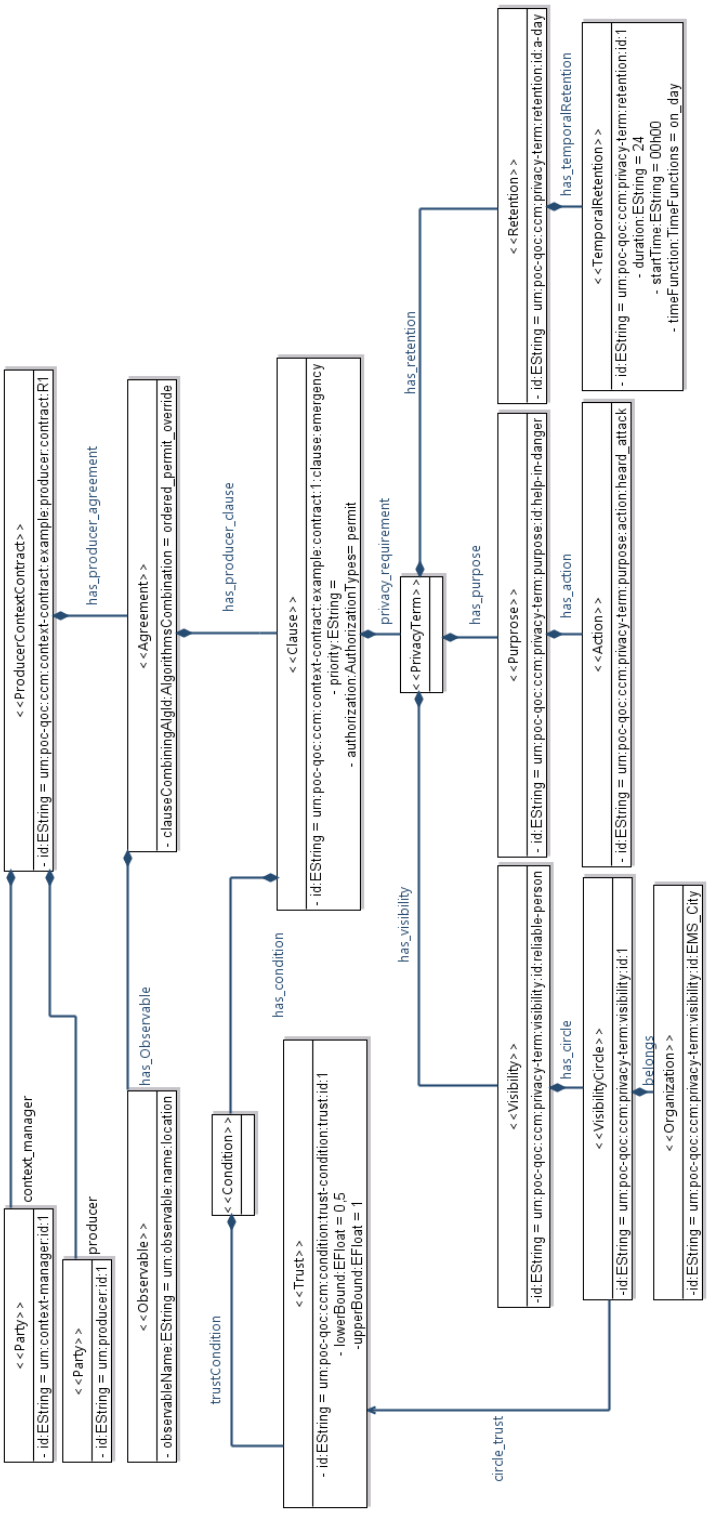


Figure 5.24: Producer Contract - Privacy Requirement Example (part A)

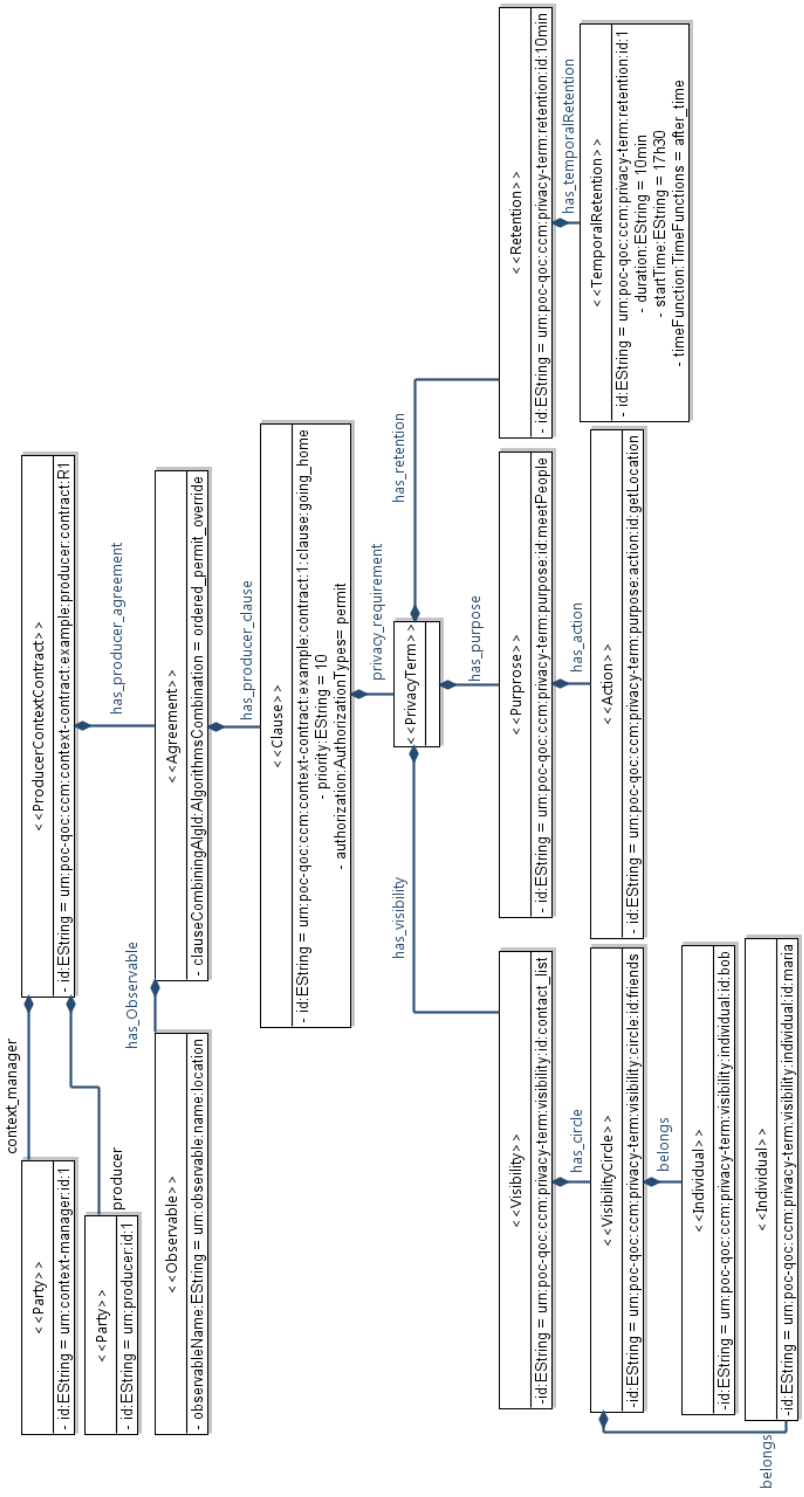


Figure 5.25: Producer Contract - Privacy Requirement Example (part B)

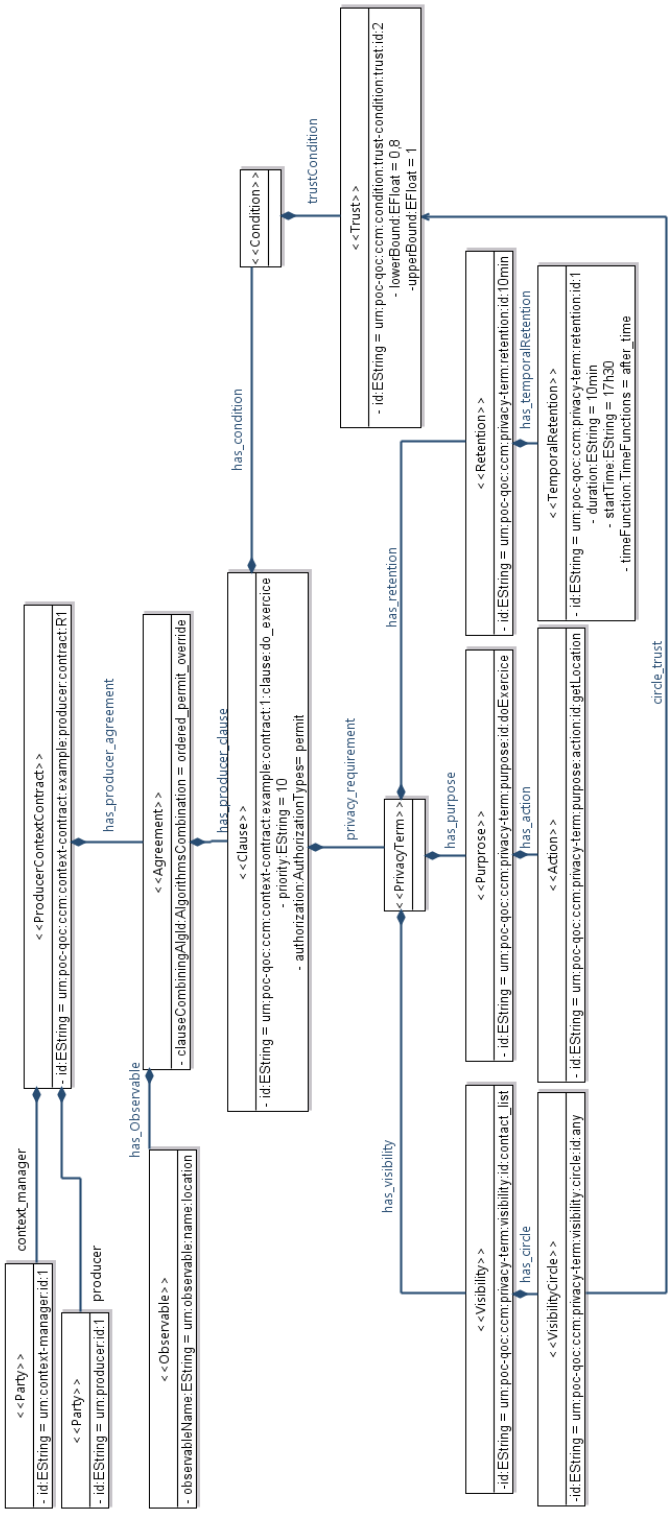


Figure 5.26: Producer Contract - Privacy Requirement Example (part C)

We demonstrate through this use case example that the MUCONTEXT contract provides mechanisms for specifying privacy requirements that can be enforced by IoT services to preserve users' private life, not only describing privacy but also ruling the QoC.

5.5.4 MUCONTEXT Contracts vs Requirements

In this section, we evaluate whether the MUCONTEXT contract meta-model has responded or not to the expectations of each requirement presented in Section 4.3. The next subsections analyze and answer the questions derived from each requirement one by one.

5.5.4.1 Handling constantly evolving context data (*R1*)

To answer the question, *Does the* MUCONTEXT *contract model allow contract designers to express different preferences according to changing situations?* , we observe how in Figure 5.20 the different conditions linked to the user situation or the environment are expressed by the MUCONTEXT contract model. Concluding that the MUCONTEXT contract can handle agreement changes based on the constant evolution of context data.

The main motivation of this requirement is to allow one to define context situations and their variability to activate privacy and QoC rules. For example, someone may want that when going home only his/her friends know his/her location. In this way, his/her home address cannot be inferred by other end-users, reserving this information for a selected group. There are uncountable examples where private information can be revealed thanks to context data. For this reason, it is important that a middleware can modify QoC or stop to collect context data from those that are not authorized even for short time periods.

The thesis scope does not cover the detection of situations. Nevertheless, the middleware needs to know the current situation. Therefore, we assume that an external or internal component provides the middleware with this information. In the applicative case, 'going to home' and 'riding on a bike' are boolean variables. Their values determine which actions the middleware will take according to the MUCONTEXT contract specifications.

The MUCONTEXT contract meta-model allows context owner and Consumer Context Application Designer to indicate the situations in which rules and behaviors should be adopted to protect the privacy. Based on that, the middleware sets up the access control and the QoC.

5.5.4.2 Handling evolutive requirements on QoC (*R2*)

Some systems can accept low levels of QoC, some others can not guarantee expected results if the QoC does not correspond to a minimum level. In the example, it does not matter if David can not meet any of his friends. However, in the case of first aid applications, it is vital to find the victim of an accident as soon as possible. After the system detects that an accident occurred, the QoC must be at its maximum to track down the victim and send help. Determining in which situations the middleware has to increase or decrease the QoC can help to improve the system functioning and increasing users' acceptance. Demanding high levels of QoC just when it is necessary improves the comfort of the people because they do not feel watched all the time.

The fact that we have chosen QOCIM as a model to represent the QoC answers positively to the question: *Does the* MUCONTEXT *contract model allows consumers to express the desired QoC taking into account the dynamic QoC variations?* (see section 4.3.2). QOCIM allows to exploit and manipulate QoC criteria in an expressive, computable, and generic way. QOCIM defines the concept of QoC indicator to associate a QoC criterion to a metric value. The metric values matched with the QoC condition lower and upper bounds. It allows us to define variation ranges for the values of the QoC. The QOCIM model has been tested by the INCOME project, enabling us to ensure its reliability.

In conclusion, we confirm that the MUCONTEXT contract meta-model offers a mechanism to enable the IoT applications to dynamically express variations of some QoC criteria.

5.5.4.3 Privacy Policy Language that supports QoC definitions (*R3*)

As we have seen in the above example, David has defined different circumstances in which he agrees to share his context data with consumers. Moreover, he defines who will access his location (based on trust or on a predefined end-users list), he defines the allowed purposes as well as the retention time for his information. With the MUCONTEXT contract meta-model, we can define the conditions before delivering the context data and the level of QoC (which may lead to obfuscation). Also we express who (Disclosure Access Control), why (Purpose Access Control), and how long (Retention Access Control) context data is shared. Those conditions are compliant with the protection criteria list proposed in Section 2.5.6.

To the question: *Is the* MUCONTEXT *contract meta-model itself a language to express privacy policies using the QoC with this purpose?* (See section 4.3.3), we answer positively. The MUCONTEXT contract meta-model is a unified solution to model heterogeneous privacy requirements and QoC criteria allowing users to express authorization and usage control constraints based on QoC criteria. The MUCONTEXT contract meta-model offers a common language to express requirements and guarantees for producers and consumers of context information. By this way, the new generation of context managers will be able to match the needs of producers and consumers. Thus, context managers could evaluate the QoC all along the life cycle of context information and apply filtering policies based on the QoC.

5.5.4.4 Evolution of Privacy Requirements (*R4*)

Is a contract easy to modify? (see section 4.3.3).

In the above example, we follow a permissive approach to guarantee that every consumer can access the location by default. Then, based on privacy or QoC requirements the policies become more restrictive to protect privacy or guarantee QoC. Depending on the needs, administrators can have permissive or restrictive approaches to create access policies. For example, firewall administrators create a permissive policy with full access and then create restrictions as it is needed (a company wants employees to access all Internet content less Facebook). On the contrary, with a restrictive approach, policies deny all access and then the administrator opens some ports. We can take any of these approaches thanks to the clause priority as well.

It is important that clauses are independent. For example, if we delete a clause, it should have no impact on the rest of the system because the end-users that are allowed to receive the location, continue to receive it, and any other privacy configuration does not change.

It is hard to measure how complex it can be to modify a MUCONTEXT contract at run-time. This complexity depends on the interdependence between the clauses that compose the contract. As seen in the example, the priority management and the combining algorithms help to solve this problem. Allowing one to add or delete clauses without causing a significant impact. For instance, if a clause is in conflict with another one (e.g., the same priority contradictory authorization), the combining algorithms should be able to take a final decision. However, this problem is also dependent on the settings made by the application designer when he creates the privacy policies.

In conclusion, the MUCONTEXT contract meta-model allows contract modifications. Nonetheless, more evidence is needed to obtain conclusive results on the runtime modification requirement.

5.5.4.5 Specifying and enforcing privacy preferences (*R5*)

Regarding the question: *Does the* MUCONTEXT *contract model allow one to express the actions to minimize the risk of privacy invasion?* (See section 4.3.3), we conclude that the MUCONTEXT contract meta-model provides the language for specifying authorization statements. Those statements can be used by the middleware (e.g., context manager) to enforce privacy requirements.

However, since we have not implemented the MUCONTEXT contract model on a real-life platform, we can not be sure of the real risk of leakage information. We assume that consumers will comply the MUCONTEXT contract dimensions (purpose, visibility, and retention), and we assume that these dimensions are sufficient to protect privacy. That is why, we include the possibility of using QoC as a mechanism to obfuscate context data while confidence between producers and consumers is created.

Measuring the trust at run-time is outside of the scope of our work. The trust model we chose to express and measure the trustfulness represents an example that should be deepened and validated by future research works.

A context owner has to be aware that sensors are not under his control and collect information about himself. The owner has two possible choices if he wants to protect his privacy: prohibit everybody to use his context data or create privacy requirements for each consumer.

Assuming that the number of context consumers can increase exponentially in the IoT, it will be impossible to know all of them to define privacy requirements for each one. Even, creating rules for known ones is a tedious and challenging task for owners and administrators. Therefore, it is necessary to establish autonomous mechanisms to allow producers to share their context data securely, even though if they do not know who will use the context data.

To solve this problem, we propose use: the trust level, the current context situation, and guarantees offered to decide the way context data will be consumed. For the example, a stub MUCONTEXT contract can be used to setup the context producers default behavior. Thereby, an IoT application can question the context owner what to do when encountering a not covered situation.

Each application should propose its stub MUCONTEXT contract, according to an analysis done by a privacy designer to determine the basic privacy needs of application users. Figure 5.27 and 5.28 show an extract of the stub MUCONTEXT contracts for the Bike4All system.

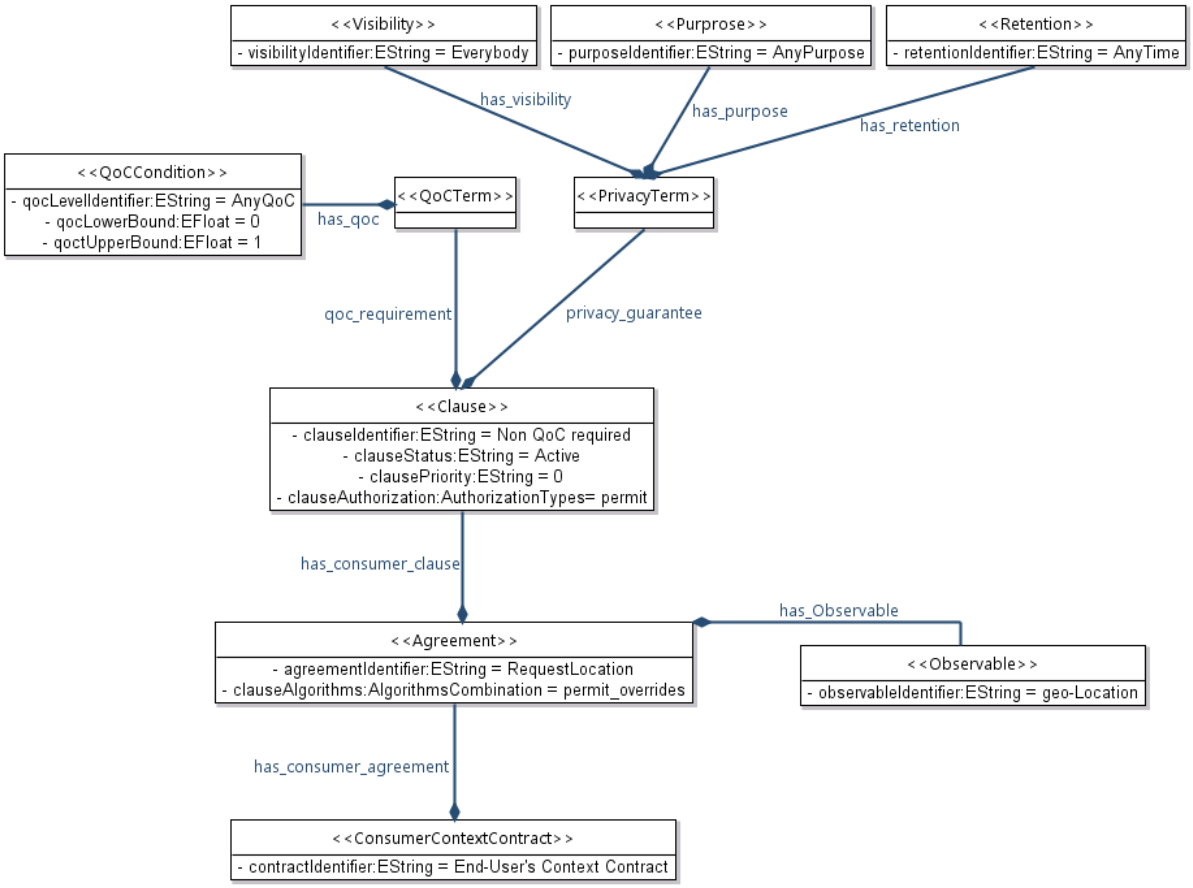


Figure 5.27: Stub Consumer Contract

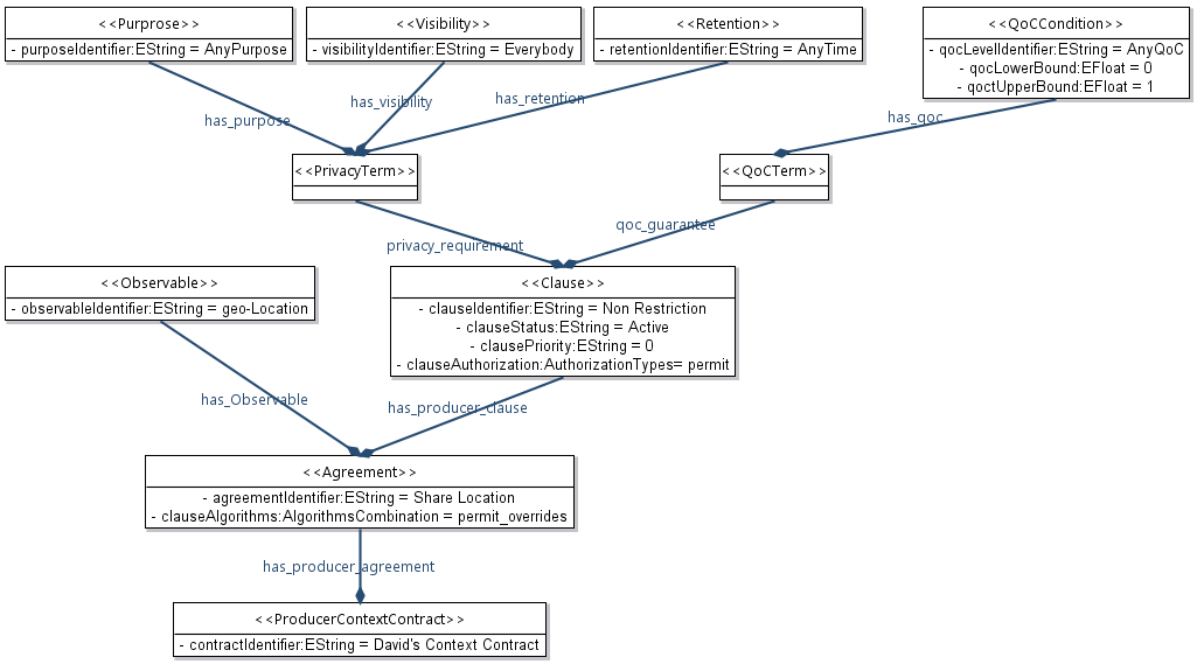


Figure 5.28: Stub Producer Contract

The context situation allows the producer to know when to share information. The purpose, the visibility and the retention set the sharing limits, and the trust allows owners to decide if they share information with strangers. In conclusion, the trust is the mechanism that allows a context owner to share his context data with unknown consumers. Additionally, the trust creates links between the producers and the consumers. Allowing producers to deliver better QoC making consumer applications more reliable.

5.6 Conclusion

In the first part of this chapter, we have analyzed the architecture in which producers and consumers interact in the context of a mass market application using the IoT. Assuming that producers and consumers are decoupled, we have introduced the notion of symmetrical contracts. Those symmetrical contracts allow producers and consumers to independently express their requirements and guarantees. We have shown that MUCONTEXT contracts have to be symmetric —i.e., a guarantee for the producer is a requirement for the consumer (and vice versa).

For all these objectives, we have defined the MUCONTEXT meta-model, which enables producers and consumers to express their contracts. These contracts allow context data producers and context data consumers to express QoC and privacy requirements and guarantees. These contracts will have to be matched at run-time by an IoT middleware (i.e., context managers). Following the state of the art of privacy taxonomy (see Section 2.5.3), we have refined the MUCONTEXT meta-model for each identified dimension: purpose, visibility, retention. We have also added a QoC dimension through the integration of

QOCIM in MUCONTEXT contracts. Moreover, we have proposed to include in MUCONTEXT contracts the **trust** concept to take into account the measure of how these agreements are respected.

Finally, we have evaluated MUCONTEXT contracts using a qualitative approach. As a first evaluation, we have shown the expressiveness of the MUCONTEXT contract meta-model through the definition of the requirements and guarantees of the scenario presented in Chapter 4. This evaluation has used the MUCONTEXT contract editor we have implemented. Secondly, we have shown that through these contracts, we validate important requirements introduced in Section 4.3 such as including context situation in the definition of contracts, extending contracts with new requirements or guarantees on QoC and privacy, providing a Privacy Policy Language that supports QoC definitions and, specifying various privacy requirements and guarantees. We should continue this validation work, for example through the definition of MUCONTEXT contracts for various IoT applications. This would necessitate the usage of tools to facilitate the manipulation of the contracts by designers.

The MUCONTEXT contract concept is a first step towards privacy and QoC enforcement. The contracts have to be evaluated at runtime by context managers. Context managers are responsible for retrieving context information needed to assess the MUCONTEXT contracts and make the run-time decisions according to the bi-party contracts, —i.e., deny access to information or also decrease QoC level for privacy purpose. Meanwhile, the contract must be flexible enough to withstand changes, without causing a significant impact on applications. In the two next chapters, we will present matching algorithms and tools to implement runtime contract matching.

Chapter 6

Algorithm to Create Agreements between Context Producers and Context Consumers

Contents

6.1 Introduction

This chapter contains the description of the matching process between producer and consumer MUCONTEXT contracts proposed in this thesis. We develop the algorithms to create agreements to share

context data between producers and consumers. These algorithms determine whether context data delivering to consumers are authorized or not, according to the matching result. We present the algorithm in three steps. Section 6.3 shows the main algorithm represented as a flow chart. Sections 6.4 and 6.5 describe the process of validation of privacy and QoC respectively. Then, Section 6.6 gives an example of an authorization decision process given a set of clauses. Finally, we present in Section 6.7 the MUCONTEXT contract matching algorithms validation through a code implementation in JAVA.

6.2 Concepts and Assumptions

In this section, we describe a generic algorithm to settle agreements between producers and consumers in analysing their respective MUCONTEXT contracts. The matching process between producer and consumer contracts addresses new challenges brought by the IoT in terms of spontaneous interactions among decoupled participants preserving the privacy concern of owners of context data using the QoC as a parameter to guarantee an accurate service.

For each observation made, the middleware decides if a report is delivered to the consumer or not. The result of the MUCONTEXT matching algorithm is necessary in the decision process (See Figure 4.5 in Section 4.4.3). A **context report** is *a structure containing the context data value (observable)*, and associated meta-data such as the QoC. The **matching between MUCONTEXT contracts** is defined as *the comparison process of producer* MUCONTEXT *contract and consumer* MUCONTEXT *contract to determine if both respect the requirements an guarantees of each other*. As both participants have symmetric contracts when one defines its requirements the other defines its guarantees. The matching process verifies if the requirements of one part are included among the guarantees offered by the other part. Therefore, the decision is based on this compatibility verification. The result of this evaluation from the producer point of view is to permit or deny the access to its context data. Complementarily, from a consumer point of view, the result is to be allowed or not to consume context data with the sufficient QoC.

6.3 Main MUCONTEXT Contract Matching Algorithm

In this section, we draw the main lines of the MUCONTEXT contract matching algorithm, which determines if the context producer allows access to the requested context data and if the context information corresponds to the consumer needs.

Figure 6.1 depicts the MUCONTEXT contract matching process. The set of necessary validations are organized from less to more expensive ones in terms of resources consumption.

We split the main MUCONTEXT contract matching algorithm into three parts according to the place where the decision processes are executed: producer context manager, MUDEBS broker, and consumer context manager. The first one (at the top and left of the figure) occurs on the producer side. The decision of sharing context data is based on the evaluation of the predicates stored in the producer contract. For that, a dynamic querying of context situations is required. Additionally, the QoC content validation evaluates if the QoC meta-data contained in the context report corresponds to the QoC guaranteed by the producer MUCONTEXT contract. The second one (at the right of the figure) occurs in the MUDEBS

brokers. This validation takes place once, when the matching of advertise and subscription filters is executed. This process matches the requirements and guarantees of both participants stored in their MUCONTEXT contracts in both senses as result an agreement or not of sharing context data is obtain. Finally, the last part of this validation process (at the bottom and left of the figure) is done at the consumer side to decide if it is the correct moment to consume the context data based on the predicates stored in the consumer Contract and the current context situation.

In this thesis, we develop the matching algorithm that is evaluated in the broker. This process is described as follows:

- After the QoC content validation, the algorithm evaluates, if it exists, mutual trust between the producer and the consumer. For that objective, a trust manager, which belongs to the IoT platform, is requested to verify if the producer has the level of trust specified by the consumer and if the consumer has the level of trust specified by the producer on their respective contracts.
- The next two validations, the producer requirements validation (see Section 6.4) and the consumer requirements validation (see Section 6.5), are the core of our proposal, in which the MUCONTEXT contracts are evaluated to decide whether the context report is forwarded or not.
- Finally, if all evaluations are true, the context report is forwarded by the IoT platform to the context consumer.

6.4 Producer Requirements Validation Algorithm

This validation verifies if the requirements of the producer fit with the guarantees given by the consumer.

6.4.1 Formalization of MUCONTEXT contracts

Before presenting the algorithm, we formalize a producer contract and a consumer contract. Those two contracts contain two clauses. The first pair of clauses is presented in Figure 6.2 and the second pair of clauses in Figure 6.3. For these two figures, the producer context clauses are on the right side and the consumer context clauses on the left.

We define the elements of these contracts. Let pContract and cContract be the two MUCONTEXT contracts. pContract is a producer contract and cContract is a consumer contract. O_1 is an observable. If ($O_1 \in$ pContract) and ($O_1 \in$ cContract) are equal, then both contracts are comparable for the first validation of the algorithm.

In this example both contracts are each composed of two clauses. Let pClause_1 (C_1 in the figure) and pClause_2 (C_2 in the figure) be clauses, such that pClause_1 and $\text{pClause}_2 \in$ pContract. pClause_1 and pClause_2 have the privacy terms privacyR_1 and privacyR_2 respectively that define the privacy requirements. And, let cClause_1 (C'_1 in the figure) and cClause_2 (C'_2 in the figure) be clauses, such that cClause_1 and $\text{cClause}_2 \in$ cContract. cClause_1 and cClause_2 have the privacy terms privacyG_1 and privacyG_2 respectively that define the privacy guarantees.

In both contracts, each privacy term contains conditions of visibility, purpose and retention. V is a set of visibilities, P is a set of purposes, and R is a set of retentions. Let privacyR_1 be a tuple of

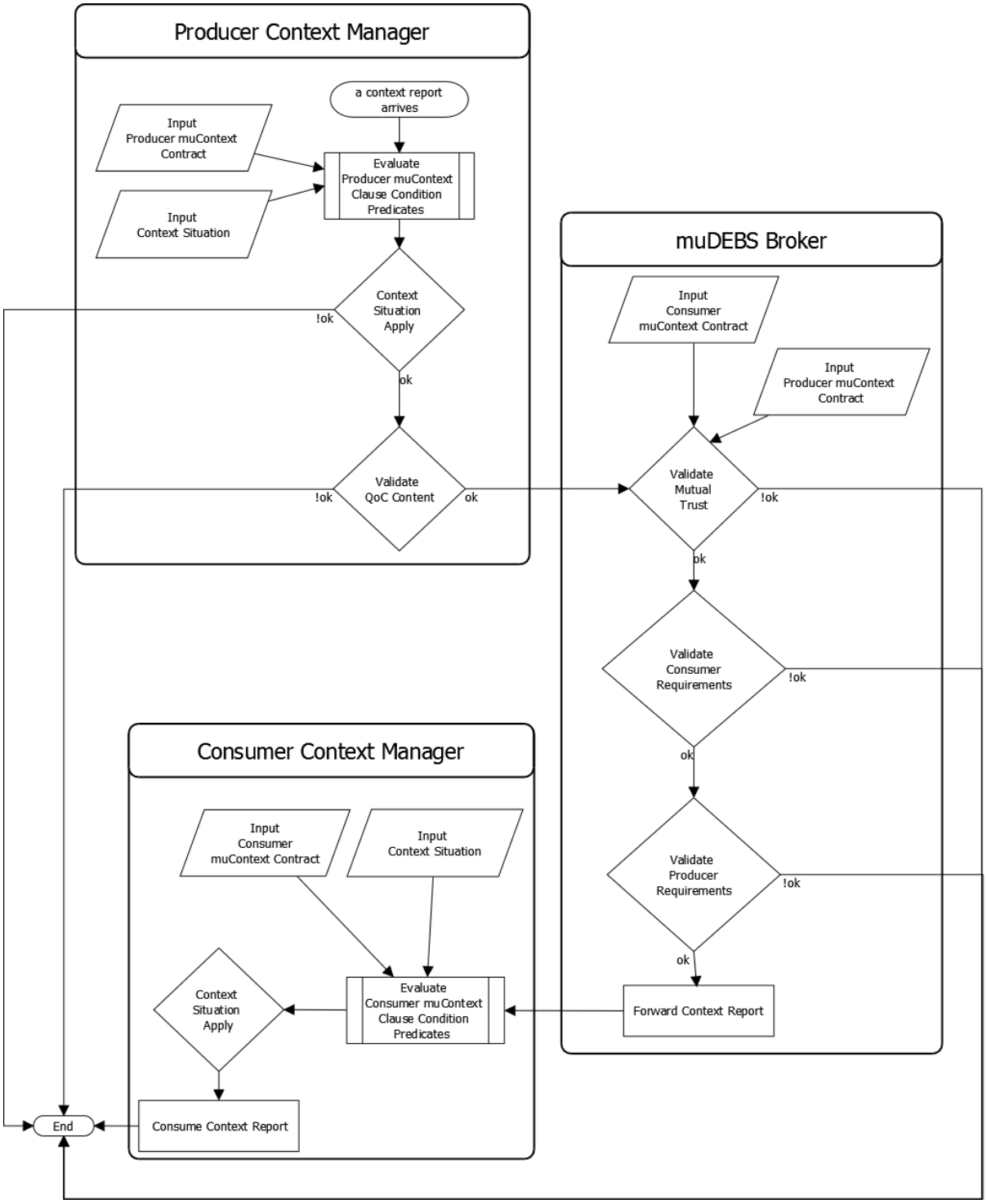


Figure 6.1: Main MUCONTEXT Contract Matching : Flow Chart

privacy conditions (v_1,p_1,r_1) such that $v_1 \in V$, $p_1 \in P$, and $r_1 \in R$. Let $privacyR_2$ be a tuple of privacy conditions (v_2,p_2,r_2) such that $v_2 \in V$, $p_2 \in P$, and $r_2 \in R$.

And let $privacyG_1$ be a tuple of privacy conditions (v'_1,p'_1,r'_1) such that $v'_1 \in V$, $p'_1 \in P$, and $r'_1 \in R$. Let $privacyG_2$ be a tuple of privacy conditions (v'_2,p'_2,r'_3) such that $v'_2 \in V$, $p'_2 \in P$, and $r'_3 \in R$.

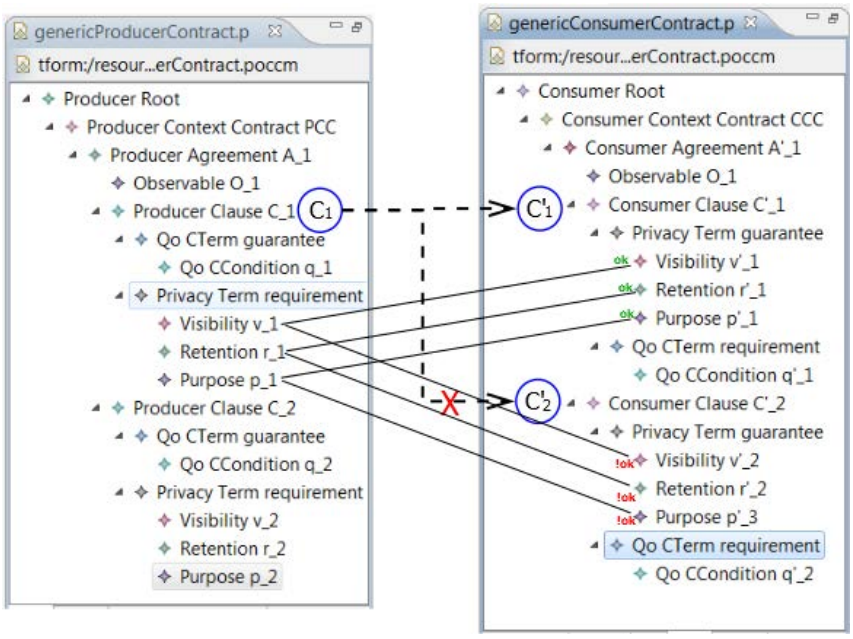


Figure 6.2: Privacy Match Validation ($pClause_1$ Iteration)

6.4.2 Producer requirement validation algorithm

In this section, we present the algorithm used to implement the producer requirement validation. The MatchValidation.privacy algorithm (see Algorithm 1) is the heart of our implementation solution. It checks if the set of clauses from the producer MUCONTEXT contract matches with the set of clauses consumer MUCONTEXT contract. A comparison of each producer clause is performed with all the consumer clauses to indicate which producer clauses match which consumer clauses. Then the algorithm takes a decision based on the algorithm combination parameter (of the Agreement class of Figure 5.5) and the clause authorization parameter (of the Clause class of Figure 5.6).

Two parameters are in input of Algorithm 1:

- ProducerContextContract, which contains all the producer privacy requirements and the parameters necessary to take a decision,
- ConsumerContextContract, which contains all the consumer privacy guarantees.

As defined above, a producer MUCONTEXT contract (pContract) is a set of clauses (pClause =

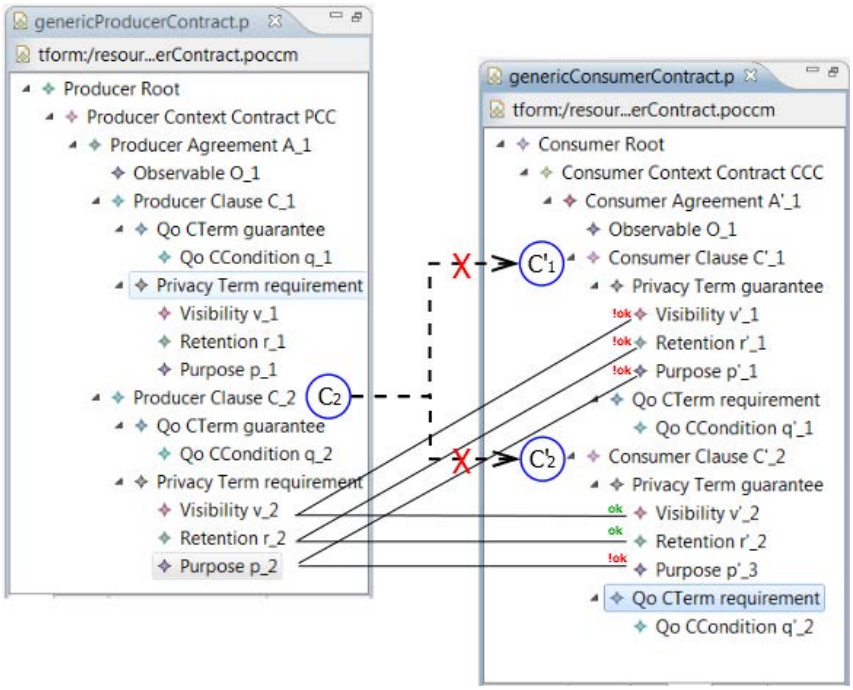


Figure 6.3: Privacy Match Validation ($pClause_2$ Iteration)

pContract.has_producer_agreement.has_producer_clause), which contains a privacy requirement (pClause.privacy_requirement) defined as a triple (p, v, r), where:

- p is a pClause.privacy_requirement.has_purpose, which represents a purpose.
- v is a pClause.privacy_requirement.has_visibility, which represents a visibility level.
- r is a pClause.privacy_requirement.has_retention, which represents a retention condition.

And, we define a consumer MUCONTEXT contract (cContract) as a set of clauses (cClause= cContract.has_consumer_agreement.has_consumer_clause) which contains a privacy guarantee (cClause.privacy_guarantee) as triple (p', v', r'), where:

- p' is a cClause.privacy_guarantee.has_purpose, which represents a purpose.
- v' is a cClause.privacy_guarantee.has_visibility, which represents a visibility level.
- r' is a cClause.privacy_guarantee.has_retention, which represents a retention condition.

The producer MUCONTEXT contract (pContract) defines the combination algorithm (pContract.has_producer_agreement.clauseAlgorithm) to be used to decide whether to publish or not. Each producer clause (pClause) defines the authorization to be obtained after the match

(pClause.authorization). If the matching process succeeds, the decision result is Permit or Deny. The evaluation is made by comparing each pClause with all the cClauses.

Finally, there are two outputs from the algorithm 1, the final decision and the decision table used by the combination algorithm.

Algorithm 1 consists of three fit functions and two basic functions:

- Fit functions:

  The definition of functions to compare the privacy criteria is out of the scope of this thesis. We suppose that each IoT framework provides services to compare each privacy criterion. Therefore, we create an interface named FitMethods to define the privacy comparison functions. The APIs of the methods to be implemented for privacy are:

  - method visibility (Visibility requirement, Visibility guarantee, Party producer): boolean,
  - method purpose (Purpose requirement, Purpose guarantee): boolean,
  - method retention (Retention requirement, Retention guarantee) : boolean.

  As a proof of concept, we provide a simple implementation of those methods. The algorithms of those methods are presented by the Algorithms 2, 3 and 4. Each criterion is organized as a tree structure shared by both parties. A requirement fits a guarantee if the parameter belongs to the same branch of the tree. That is to say, privacyG_1 corresponds to privacyR_1, if $v'_1 = \text{childOf}(v_1) \wedge p'_1 = \text{childOf}(p_1) \wedge r'_1 = \text{childOf}(r_1)$. If privacyG_1 corresponds to privacyR_1 then a decision can be taken based on the pClause_1 authorization parameter.

  The input values of each method may depend on the implementation. As a minimum, all methods require the requirements and guarantees defined by the terms of their respective clauses. In addition, in our simple implementation, the identifier of the producer, stored in its MUCONTEXT contract, is also necessary to determine the visibility circles of a producer.

- clauseEvaluationDecision (AuthorizationType clauseAuthorization, Boolean clauseVerified): AuthorizationType.

  The function clauseEvaluationDecision() (Algorithm 5) returns Permit or Deny based on the clauseAuthorization parameter setup in the producer contract for this clause and on the evaluation for the producer clause. The first parameter indicates the type of authorization setup in the producer MUCONTEXT contract (See the Clause meta-class attributes in Section 5.3.1)

  The producer expresses if the clause should be verified or should not by the consumer. The second parameter determines if the producer clause is verified by at least one consumer clause. For example, if clauseVerified is True and the clauseAuthorization is DENY the function finally returns DENY, because the contract has defined that the result must be denied if the clause is verified.

- optionCombinationAgorithm (AlgorithmsCombination clauseAlgorithm, Result(List) decision_table): AuthorizationType

The function optionCombinationAgorithm (Algorithm 6) returns a final decision based on the combination of the different results obtained for the evaluation of each clause in the MUCONTEXT contract. (See the Agreement meta-class attributes in Section 5.3.1)

The first parameter indicates which combination algorithm will be used to take the final decision. The second parameter is a list which stores all the decisions obtained from the evaluation of each clause in the producer MUCONTEXT contract.

Figures 6.2 and 6.3 represent the iteration 1 and iteration 2 respectively of the privacy validation process for the two examples of MUCONTEXT contracts of Section 6.4.1.

In our example, the iteration 1 matches the clause C1 with clauses C'1 and C'2, obtaining only a positive match between C1 and C'1. If we assume that the pClause.authorization is set to Permit for the individual decision of this clause, and assuming that the pContract.has_producer_agreement.clauseAlgorithm is set to PERMIT_OVERRIDES (the combination algorithms are explained next in Section 6.6), then if one of the decisions is Permit, the final decision is Permit.

6.5 Consumer Requirement Validation Algorithm

In the following, we describe the consumer requirement validation (Algorithm 7). In our model the main concerns of context consumers is to obtain from context data an adequate level of QoC. This algorithm is used to match the consumer requirements versus the producer guarantees. This algorithm is very similar to the producer requirement validation (Algorithm 1). Accordingly, we do not go thoroughly into the explanations, we only remark the differences.

The input parameters ProducerContextContract and ConsumerContextContract are passed to Algorithm 7. On the contrary to Algorithm 1, the ConsumerContextContract indicates all the requirements of QoC and the configurations to take a decision. And the guarantees are represented by a ProducerContextContract.

As defined above, a producer muContext contract (pContract) is a set of Clauses (pClause = pContract.has_producer_agreement.has_producer_clause), which contain a QoC guarantee (pClause.qoc_guarantee) as (q), where:

- q is a pClause.qoc_guarantee.has_qoc, which represents a level of QoC.

And, we define a consumer muContext contract (cContract) as a set of Clauses (cClause = cContract.has_consumer_agreement.has_consumer_clause), which contain a QoC requirement (cClause.qoc_requirement) as (q'), where:

- q' is a cClause.qoc_requirement.has_qoc, which represents a level of QoC.

Algorithm 7 consists of one fit function (See Algorithm 8) and the same basic functions used in Algorithm 1. Finally, an important remark is, that the evaluation is made by comparing each cClause with all the pClauses.

6.6 Clause Combination Algorithms

The MUCONTEXT contract meta-model defines the behavior of the algorithm itself. For this purpose, it includes the choice of the algorithm to be used to combine the different clauses. The combination functions define the procedures for reaching an authorization decision given the individual results of evaluation of a set of clauses. In this section, we introduce some of these algorithms inspired by the XACML 3.0 [XACMLv3, 2013] specification. As an example, we choose the Deny Overrides Algorithm (Section 6.6.1), and Permit Overrides Algorithm (Section 6.6.2). We choose these algorithms because they are part of the examples we presented in Section 5.5.

6.6.1 Deny Overrides Algorithm

The deny_overrides combining algorithm is intended for those cases where a deny decision should have priority over a permit decision. This algorithm has the following behavior:

1. If any decision is Deny, the result is Deny.
2. Otherwise, the result is Permit.

6.6.2 Permit Overrides Algorithm

The permit_overrides combining algorithm is intended for those cases where a permit decision should have priority over a deny decision. This algorithm has the following behavior.

1. If any decision is Permit, the result is Permit.
2. Otherwise, the result is Deny.

6.7 Validation of MUCONTEXT Contract Matching Algorithms

In this section, we discuss the correctness and efficiency of the presented algorithms. To verify if the algorithms really solve the problem for which it was designed, we follow an experimental analysis (testing) approach. We test the algorithm for different input data.

In Section 6.7.1, we define the goal of our validation and describe the expected results. Secondly, we describe the experimental setup in Section 6.7.2, before presenting the results in Section 6.7.3.

6.7.1 Experiment

To experimentally evaluate MUCONTEXT matching algorithm, we have designed a test data set [5]for each test case [6] we wrote.

[5]input data required for each test.

[6]"A test case has components that describes an input, action or event and an expected response, to determine if a feature of an application is working correctly."

Broadly, there are two classes of results we are interested in. The first one relates to the correctness of the algorithm. Taking into consideration the different number of possible combinations between the combining algorithms, authorizations, number of clauses, number of matches between consumer requirements and producer requirements, as well as the guarantees. These results are discussed in Section 6.7.3.1. We also analyze the differences between the MUCONTEXT Contract terms match and the MUCONTEXT contract condition-predicate match with the environment to take the decisions of forwarding context data.

The second category of results that are of interest relates to the impact of the terms matching algorithms (i.e., QoC and Privacy) on the operational performance of the MUCONTEXT Manager Service, for example, the time necessary to compare a requirement with a guarantee. We also experiment with different MUCONTEXT Contracts constructs during this study, based on a varying number of clauses and their components. These results are discussed in Section 6.7.3.2.

6.7.2 Experimental setup

The algorithms are written in Java and have four major components.

- MUCONTEXT contract creator that loads and parses the XMI representation of MUCONTEXT contracts complain of meta-model definition generated with MUCONTEXT Contract editor presented in Section 5.5.2;
- main MUCONTEXT contract matching algorithm, which executes the flow of validations (See Figure 6.1) required to take the decision of forwarding a context report;
- consumer requirement validation, which is used to match the consumer QoC requirements versus the producer QoC guarantees by comparing each consumer clause with all the producer clauses;
- producer requirement validation involves comparing whether the privacy requirements of the producer fits with the privacy guarantees given by the consumer.

For our experiments, we have considered three sorts of MUCONTEXT contracts for each party based on the number of clauses declared (one, two and three clauses). These contracts were used by the automation test process as templates to setup the different inputs required.

We design an automation test procedure that executes in batch mode the MUCONTEXT contract matching algorithm. Each test matches two MUCONTEXT contracts that are setup from a Comma-separated values (CSV) file. We observe for each iteration the variable status, the forwarding decision, and the execution time.

The data set input for each test is defined by the following fields shown in Tables 6.1 for consumers and 6.2 for producers:

In the tables, the first column indicates the MUCONTEXT contract template to be used in the tests. The second column identifies the test. The following columns correspond to the properties of the contract to be modified by the automation process, for instance, getHas_producer_agreement().setClauseCombiningAlgId() column sets the combining algorithm of the MUCONTEXT contracts.

We prepared some test cases with their data test sets that incorporate all the algorithm features we want to test. Each data test set contains the possible combinations among the combining algorithm (permit_overwrite and deny_overwrite), clause authorization decisions (permit and deny), and guarantees and requirements (qoc, purpose, visibility and retention). For example, a data test set may look similar to Table 6.3

It is important to remark that we are interested in testing the matching of contracts, but not in how the terms are compared. For this reason, we implement a fit method to compare the QoC and privacy terms of each contract by comparing their identifiers. An implementation of the fit method has to be modified according to the requirements of the IoT platform. We detail below the test cases.

6.7.3 Experimental analysis

The algorithm takes in input two MUCONTEXT contracts, one for a producer and another one for a consumer, and verifies if they are compatible in terms of guarantees and requirements (See 6.3). To finally obtain two decisions: one for the producer, permit or deny access to context data, and the second one for the consumer to permit or not context data consumption.

Here, we discuss the correctness of both the producer requirements validation algorithm (Algorithm 1) and the consumer requirements validation algorithm (Algorithm 7), which are the heart of the MUCONTEXT matching algorithm, using an experimental analysis approach.

6.7.3.1 Correctness

There are several works using data flows for selecting test cases as presented in [Rapps and Weyuker, 1982]. Rapps et al. apply data flow analysis techniques to examine test data selection criteria. The data flow criteria that they have defined can be used to traverse each path. In contrast to their formal approaches, we use a visual approach for selecting our test cases. In the following, we concentrate on creating test cases at three different levels:

1. Test cases for a single consumer requirement (QoC matching): The QoC matching process will have to check whether the context data access decision is correctly taken. Each context data access decision can be tested independently from the whole process. In order to create, test, and execute these test cases respectively, we require necessary input and output data for each path of the process [Rapps and Weyuker, 1982] (see Figure 6.4, The elements highlighted in green represent the path followed in the test case). In order to provide appropriate input and output data, we create and test a data set composed by 32 tests that combine the different combining algorithms by authorization decision, and equality of the QoC terms. Our approach enables to compare a producer and consumer MUCONTEXT contracts with only one clause.

2. Test cases for a single producer requirement (privacy matching): The privacy matching process will have to check whether the context data access decision is correctly taken. For each possible path in this producer requirement validation, a test case has to be created, tested, and executed in order to verify the correctness of each path. Figure 6.5 examplifies the different paths of the context data access decision process. The green arrows mark a specific path within the process

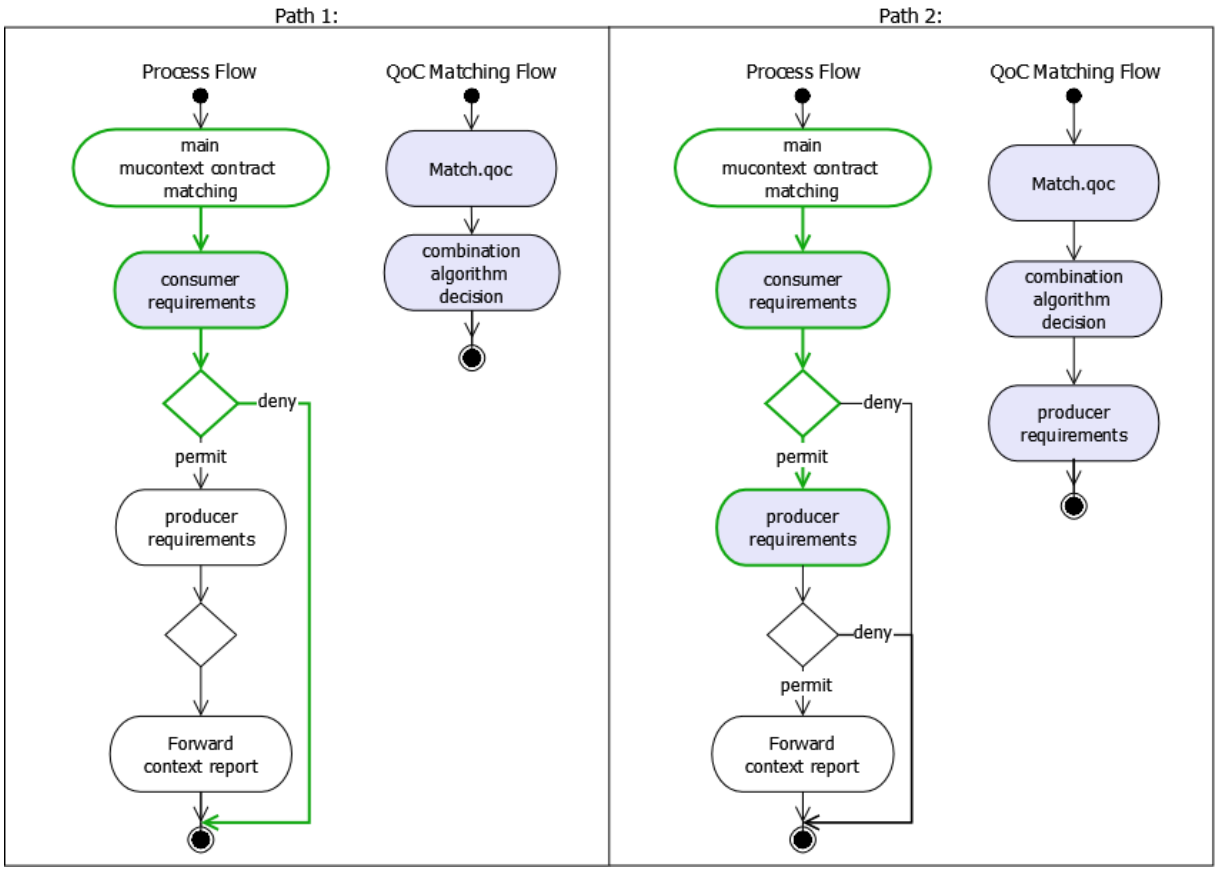


Figure 6.4: Motivating example for Consumer Requirement Validation Algorithm, Paths 1 and 2

flow. In order to provide appropriate input and output data, we create and test a data set composed by 32 tests that combine the different combining algorithms by authorization decision, and equality of the Purpose terms. We assume that testing only one privacy dimension is enough to validate this algorithm feature. Our approach enables to compare a producer and consumer MUCONTEXT contracts with only one clause.

3. Test cases for integrating producer and consumer requirements matching process: The privacy and QoC matching process will have to check whether the context data access decision is correctly taken. Each context data access decision must be tested from the whole process. In order to provide appropriate input and output data, we create and test a data set composed by 259 tests that combine the different combining algorithms by authorization decision, and equality of the QoC terms and privacy terms (Purpose, Visibility and Retention). Our approach enables to compare a producer and consumer MUCONTEXT contracts with only one clause.

   The execution of this test gives as a result that 192 tests took the path 1, 59 tests follow the paths 2 and 3. Finally, only 8 tests give access to the context data following the paths 2 and 4.

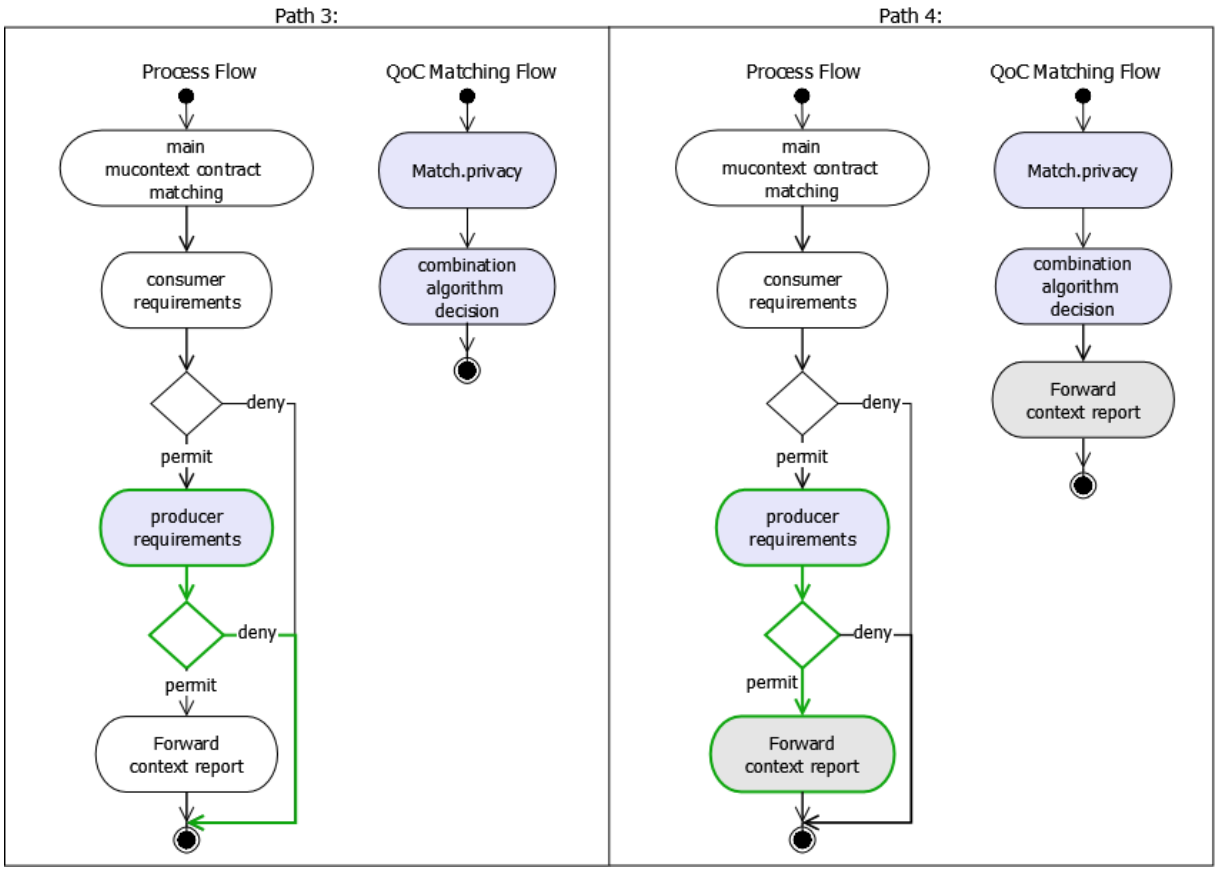


Figure 6.5: Motivating example for Producer Requirement Validation Algorithm, Paths 3 and 4

In conclusion, the third test case results confirm the results of the two previous test, proving the correctness of the MUCONTEXT contract matching algorithm. The next section tests the efficiency of our algorithm.

See in Appendix D the input and output data for each test case executed.

Function: MatchValidation.privacy ;

Input: ProducerContextContract pContract
Input: ConsumerContextContract cContract
Output: AuthorizationTypes decision
Output: List(Result) decision_table
ProducerClause pClause;
ConsumerClause cClause;
FitMethods fitMethod `// Class defining the match functions`
CombinationAlgorithms ca `// Class defining the decision combination functions`
Boolean pEvaluation;
Boolean vEvaluation;
Boolean rEvaluation;
int order = 0;
`// Clauses evaluation order`
foreach *pClause* ∈ *pContract.has_producer_agreement.has_producer_clause* **do**
    Boolean,clauseVerified = False `// clause requirement exists in consumer clause set`
    foreach *cClause* ∈ *cContract.has_consumer_agreement.has_consumer_clause* **do**
        vEvaluation = fitMethod.visibility(pClause.privacy_requirement.has_visibility,
        cClause.privacy_guarantee.has_visibility,
        pContract.producer) ;
        pEvaluation = fitMethod.purpose(pClause.privacy_requirement.has_purpose,
        cClause.privacy_guarantee.has_purpose);
        rEvaluation = fitMethod.retention(pClause.privacy_requirement.has_retention,
        cClause.privacy_guarantee.has_retention);
        clauseVerified = (vEvaluation ∧ pEvaluation ∧ rEvaluation);
        if *clauseVerified* **then**
            break
        end
    end
    `// Store each decision by evaluated clause`
    Result result;
    order = order + 1 ;
    result.order = order;
    result.pClauseId = pClause.id;
    result.cClauseId = cClause.id;
    result.priority = pClause.priority;
    result.decision = clauseEvaluationDecision (pClause.authorization, clauseVerified);
    decision_table.add (result);
end
decision = optionCombinationAgorithm
(pContract.has_producer_agreement.clauseAlgorithm , decision_table);
return decision, decision_table ;

Algorithm 1: Producer Requirement Validation algorithm

Function: FitMethod.purpose;
Input: Visibility requirement
Input: Visibility guarantee
Output: Boolean
`// If Producer Contract did not define a purpose requirement`
if *isNull(requirement.id)* **then**
| **return** True
end
return matchFunction(requirement, guarantee)

Algorithm 2: Generic Purpose Match Function

Function: FitMethod.visibility;
Input: Visibility requirement
Input: Visibility guarantee
Input: Party party
Output: Boolean
`// If Producer Contract did not define a visibility requirement`
if *isNull(requirement.id)* **then**
| **return** True
end
return matchFunction(requirement, guarantee, party)

Algorithm 3: Generic Visibility Match Function

Function: FitMethod.retention;
Input: Visibility requirement
Input: Visibility guarantee
Output: Boolean
`// If Producer Contract did not define a retention requirement`
if *isNull(requirement.id)* **then**
| **return** True
end
return matchFunction(requirement, guarantee)

Algorithm 4: Generic Retention Match Function

```
Function: clauseEvaluationDecision;
Input: AuthorizationType clauseAuthorization
Input: BooleanclauseVerified
Output: AuthorizationType
if clauseVerified then
    return clauseAuthorization
end
else
    if clauseAuthorization == AuthorizationType.PERMIT then
        return AuthorizationType.DENY
    end
    else if clauseAuthorization == AuthorizationType.DENY then
        return AuthorizationType.INDETERMINATE
    end
end
```

Algorithm 5: clauseEvaluationDecision()

```
Function: optionCombinationAgorithm;
Input: AlgorithmsCombination clauseAlgorithm
Input: Result(List) decision_table
Output: AuthorizationType
if clauseAlgorithm == AlgorithmsCombination.DENY_OVERRIDES then
    return deny_overrides(decision_table);
end
if clauseAlgorithm == AlgorithmsCombination.ORDERED_DENY_OVERRIDES then
    return ordered_deny_overrides(decision_table);
end
if clauseAlgorithm == AlgorithmsCombination.PERMIT_OVERRIDES then
    return permit_overrides(decision_table);
end
if clauseAlgorithm == AlgorithmsCombination.ORDERED_PERMIT_OVERRIDE then
    return ordered_permit_override(decision_table);
end
if clauseAlgorithm == AlgorithmsCombination.FIRST_PRIORITY_APPLICABLE then
    return first_priority_applicable(decision_table);
end
```

Algorithm 6: optionCombinationAgorithm()

```
Function: MatchValidation.qoc ;
Input: ProducerContextContract pContract
Input: ConsumerContextContract cContract
Output: AuthorizationTypes decision
Output: List(Result) decision_table
ProducerClause pClause;
ConsumerClause cClause;
FitMethod fitMethod;
// Class defining the match functions CombinationAlgorithms ca;
// Class defining the decision combination functions
Boolean qEvaluation;
int order = 0 // Clauses evaluation order;
foreach cClause ∈ cContract.has_consumer_agreement.has_consumer_clause do
    Boolean clauseVerified = False;
    // clause requirement exists in producer clause set foreach
    pClause ∈ pContract.has_producer_agreement.has_producer_clause do
        qEvaluation = fitMethod.qoc(cClause.qoc_requirement.has_qoc,
        pClause.qoc_guarantee.has_qoc) ;
        clauseVerified = (qEvaluation);
        if clauseVerified then
            break;
        end
    end
    // Store each decision by evaluated clause
    Result result ;
    order = order + 1 ;
    result.order = order;
    result.pClauseId = pClause.id;
    result.cClauseId = cClause.id;
    result.priority = cClause.priority;
    result.decision = clauseEvaluationDecision (cClause.authorization, clauseVerified);
    decision_table.add (result);
end
decision = optionCombinationAgorithm (
cContract.has_consumer_agreement.clauseAlgorithm, decision_table);
return decision, decision_table ;
```

Algorithm 7: Consumer Requirement Validation algorithm

Function: FitMethod.qoc;
Input: QoC requirement
Input: QoC guarantee
Output: Boolean
`// If Consumer Contract did not define a QoC requirement` **if** *isNull(requirement.id)* **then**
  return True
end
return matchFunction(requirement, guarantee)

Algorithm 8: Generic QoC Match Function

Input: Result(List) decision_table
Output: AuthorizationType
foreach *evaluation* ∈ *decision_table* **do**
  decision = decision_table[evaluation].decision;
  if *decision == AuthorizationType.DENY* **then**
    return AuthorizationType.DENY;
  end
end
return AuthorizationType.PERMIT

Algorithm 9: deny_overrides()

Input: Result(List) decision_table
Output: AuthorizationType
atLeastOneDeny = False;
foreach *evaluation* ∈ *decision_table* **do**
  decision = decision_table[evaluation].decision ;
  if *decision == AuthorizationType.PERMIT* **then**
    return AuthorizationType.PERMIT ;
  end
end
return AuthorizationType.DENY ;

Algorithm 10: permit_overrides()

Consumer Contract Template (File name)	TestID	Consumer Contract						
		getHas_consumer_agreement(). setClauseCombiningAlgId()	.getHas_consumer_agreement().getHas_consumer_clause() Clause 1				Clause 2	Clause 3
			.setAuthorization()	.getQoc_requirement(). getHas_qoc().setId()	.getPrivacy_guarantee(). getHas_purpose().setId()	.getPrivacy_guarantee(). getHas_visibility().setId()	.getPrivacy_guarantee(). getHas_retention().setId()	...	...

Table 6.1: Consumer Data Set Test Fields

Producer Contract Template (File name)	TestID	Producer Contract						
		getHas_producer_agreement(). setClauseCombiningAlgId()	.getHas_producer_agreement().getHas_consumer_clause() Clause 1				Clause 2	Clause 3
			.setAuthorization()	.getQoc_guarantee(). getHas_qoc().setId()	.getPrivacy_requirement(). getHas_purpose().setId()	.getPrivacy_requirement(). getHas_visibility().setId()	.getPrivacy_guarantee(). getHas_retention().setId()	...	...

Table 6.2: Producer Data Set Test Fields

ConsumerContractFile	ProducerContractFile	testId	getHas_consumer_agreement() .setClauseCombiningAlgId()	getHas_consumer_clause() .setAuthorization()	getHas_producer_agreement() .setClauseCombiningAlgId()	getHas_producer_clause() .setAuthorization()	pQoC = cQoC
1X1Consumer.poccm	1X1Producer.poccm	1X1_1	deny_overrides	deny	deny_overrides	deny	true
1X1Consumer.poccm	1X1Producer.poccm	1X1_2	deny_overrides	deny	deny_overrides	permit	true
1X1Consumer.poccm	1X1Producer.poccm	1X1_3	permit_overrides	permit	deny_overrides	deny	true
1X1Consumer.poccm	1X1Producer.poccm	1X1_4	permit_overrides	permit	deny_overrides	permit	true
1X1Consumer.poccm	1X1Producer.poccm	1X1_9	deny_overrides	deny	deny_overrides	deny	false
1X1Consumer.poccm	1X1Producer.poccm	1X1_10	deny_overrides	deny	deny_overrides	permit	false
1X1Consumer.poccm	1X1Producer.poccm	1X1_11	permit_overrides	permit	deny_overrides	deny	false
1X1Consumer.poccm	1X1Producer.poccm	1X1_12	permit_overrides	permit	deny_overrides	permit	false

Table 6.3: Data Test Set Example

6.7.3.2 Efficiency

In this section, we present the results of benchmarking the impact of the terms matching algorithms (i.e., QoC and Privacy) on the operational performance of the MUCONTEXT Manager Service. We designed an experiment to measure the number of MUCONTEXT contracts compared during one second and the execution time of the algorithm.

Experiment Setup

To experimentally evaluate our proposed MUCONTEXT contract matching algorithm, we build a test set, which objective is to determine if the number of clauses and the decision type (Permit or Deny) affects the execution time of our algorithm. The test setup is as follows:

- **Hardware:** For this test, we used one Laptop machine with the following features: a Intel®Core(tm) i5-2540M Processor with 2.60 Ghz speed and 6 GigaBytes of Random Access Memory (RAM), running on Windows 7 Enterprise (x64bits) Operative System.
- **Input:** The test data input were split into three groups according to the number of clauses. Table 6.4 shows these groups. For instance, Group 1 corresponds to test the matching of a consumer MUCONTEXT contract with a producer MUCONTEXT contract with only one clause each.

	Number of Clauses by MUCONTEXT Contract	
	Consumer	Producer
Group 1X1	1	1
Group 1X2	1	2
Group 3X3	3	3

Table 6.4: Test data input groups by number of clauses

The other aspect we want to measure is whether the result access decision affects the execution time. To measure that, we divided the previous groups according to the decision result: We have three possible paths that include all the possible decision results. The path 1 (see figure 6.4), path 3 and path 4 (see Figure 6.5). Obviously to reach the paths 3 and 4, the algorithm must go through the paths 1 and 2.

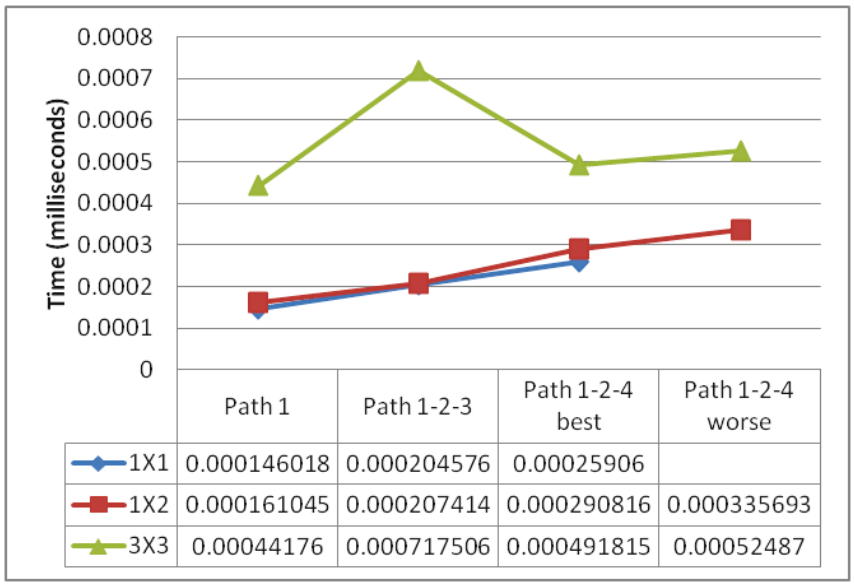
To compute the performance of our MUCONTEXT contract matching algorithm, we construct a suite of 100 independent trials for which the algorithm is executed to eliminate as much variability as possible.

In a perfect computer, the 100 trials should all require exactly the same amount of time. Of course this is unlikely to happen, since modern operating systems have numerous background processing tasks that share the same CPU on which the performance code executes.

Experiment Analysis

Our hypothesis is that the time to obtain a decision will vary directly in relation to the number of clauses and the type of decision (Permit or Deny). The bigger the number of clauses, the more time will be required to match the contracts. Figures 6.6 and 6.7 show the results of our experiment.

Figures 6.6 show the average time (over the 100 trials) in milliseconds each execution takes. Each line represents the groups of experiments according to the number of clauses and each scatter plot represents the average time that an execution has taken by path.



	Path 1	Path 1-2-3	Path 1-2-4 best	Path 1-2-4 worse
1X1	0.000146018	0.000204576	0.00025906	
1X2	0.000161045	0.000207414	0.000290816	0.000335693
3X3	0.00044176	0.000717506	0.000491815	0.00052487

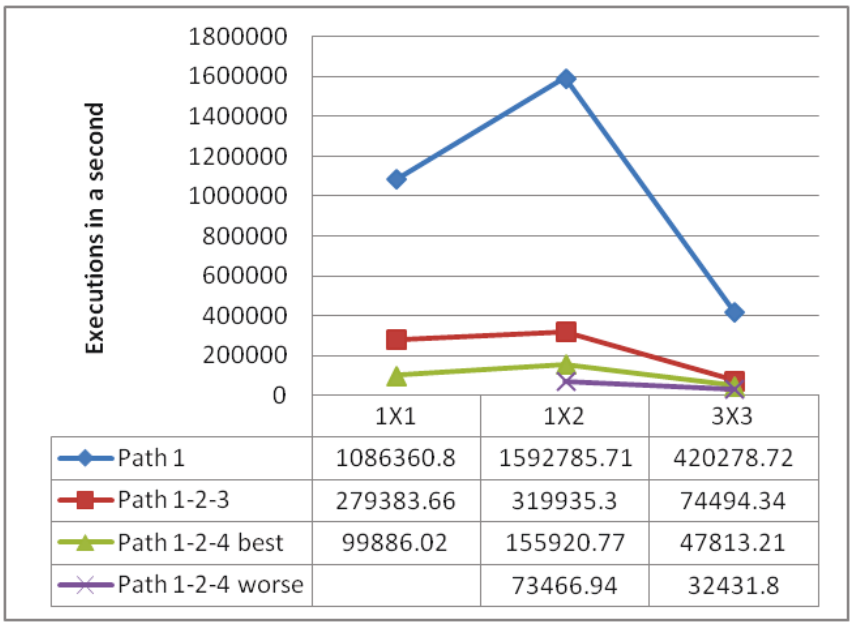
Figure 6.6: Average execution time by path and number of clauses

As we can observe that the bigger is the number of clauses, the more time it takes to obtain a decision result. However, we observe a strange trading range when the algorithm compares two contracts of the group 3X3 on the path 1-2-3. The result should be shorter than for the path 1-2-4. This phenomenon occurred because when there is no matching between the clauses. Otherwise, when it finds a match, the loop is broken.

Regarding this case, we add a new entry to experiment to consider a worse case when the clauses order forced to execute more iterations. In the worst of the cases, only one match occurs between clauses, which is found after having compared all the above clauses. So that, the algorithm must cycle through n * (m -1) times at most to find a match. Where “n” is the number of Producer Clauses and “m” is the number Consumer Clauses.

Figure 6.7 shows the average (over the 100 trials) number of executions done during one second. Each line represents the path followed by the algorithm to get a decision. We did this experiment to confirm the result of our previous experiment and we reached the same results.

In conclusion, we can confirm our hypothesis. The time of execution varies directly with the number of clauses and the result decision type. The bigger the number of clauses, the more time it takes to reach an access decision. We add a new factor that is the order of clauses that affects the time in which a clause match is found.



	1X1	1X2	3X3
Path 1	1086360.8	1592785.71	420278.72
Path 1-2-3	279383.66	319935.3	74494.34
Path 1-2-4 best	99886.02	155920.77	47813.21
Path 1-2-4 worse		73466.94	32431.8

Figure 6.7: Average number of executions during 1 second by path and number of clauses.

6.8 Conclusion

In this chapter, we have presented a deterministic algorithm that handles the decoupled IoT participants problem. Our algorithm matches IoT participants requirements and guarantees expressed into MUCONTEXT contracts to reach agreements to share context information. Firstly, we have presented a flowchart that shows an overview of the different validation stages for which the matching process must pass to grant access to context data. Each of these stages represent a part of the problem, which is done for one or more sub-algorithms. The second part shows one of these sub-algorithms, the matching process between the producer requirements and the consumer guarantees. Then, the third part shows the comparison process between consumer requirements against producer guarantees. Finally, we present a set of algorithms to exemplify how multiple results are combined to obtain a final access decision.

We conclude that the decoupled IoT participants problem split into two parts: 1) the misunderstanding of dimension definitions and 2) the independent definition of requirements and guarantees. To solve

the first part of this problem our algorithm proposes the use of a common IoT platform to handle a shared knowledge to make both parties understand each other (i.e. have the same definitions for purposes, retentions, visibilities and QoC levels). To do that, the "Fit functions" should be provided by the IoT platform. For the second part, as producers and consumers define their MUCONTEXT contracts separately, the clauses in both contracts are ordered randomly. Hence, we are forced to match each clause of one party against all the clauses of the other one. That is, the matching process searches in all the guarantees whether the analyzed requirement matches with before moving to the next clause. This process must be repeated for both the consumer and the producer.

We observe that, it is not always the context producer which denies access to context information, a context consumer can also decide not to consume it. For example, the context producer agrees to share context information, but the consumer is not interested due to the low QoC. In some other cases, the context data is no more applicable; the conditions in which the context data were needed have changed. For example, context information is necessary when the person is close to a place, but when the data arrive the person is way too far from the place. In conclusion, we do a similar comparison process twice to match both sides requirements.

Although, we include the concept of conditions in the MUCONTEXT contract meta-model for describing a predicate to indicate under what circumstances a clause should be applied. For example, do not share the GPS position if one person goes home to prevent others to infer where she lives, or only share the position if the person is close to a place. We did not include in our algorithms the evaluation of conditions. We concentrate our effort in the QoC and privacy concerns and the participants decoupling problem, this evaluation is part of a future work.

The evaluation of MUCONTEXT contracts may be expensive in resources and time. That is why, we recommend to perform this process only when it is necessary. We suggest for future works, to use a caching method and the observer pattern, to avoid unnecessary MUCONTEXT contract matching evaluations. As well as to use multiprocessing to perform producer and consumer matches simultaneously to optimize the response time.

Chapter 7

Translation of MUCONTEXT contract into XACML 3.0

Contents

7.1 Introduction

In Section 2.7.4, we introduced the fundamentals of XACML and explained the advantages of using XACML for the run-time verification of MUCONTEXT contracts. For this purpose, we propose to translate producer contracts into XACML policies and consumer contracts into XACML requests. This allows us to take advantage of existing XACML implementations[7]. Our final goal is to integrate the matching algorithms presented in Chapter 6 into XACML implementations. XACML evaluates at run-time context information in the Policy Information Points (PIP) to take access control decisions. The PIP enables dynamic access control, which is an interesting feature for the IoT.

[7] Our target implementation is Balana [WSO2, 2014]

In this chapter, we present the approach followed to translate MUCONTEXT contracts into XACML policies and requests. The main goal is that the MUCONTEXT contract matching evaluation can use existing XACML implementations with minimum extensions.

The translation from MUCONTEXT contracts into XACML policies should verify the following properties.

1. Equivalence: a MUCONTEXT contract expressed in XACML (Policy Set and Decision Request) needs not to be strictly equivalent in structure to our model. So long as the meaning remains the same and so long as the XACML (Policy Set and Decision Request) would lead to the same response (decision and obligation) defined in the algorithms presented in Section 2.7.4.

2. Lossless behavior: it MUST be possible to translate MUCONTEXT contracts into XACML representations in either direction without semantic loss.

3. Self-contained nature: the XACML representation MUST contain all the information the MUCONTEXT contract contains.

The remainder of this chapter is organized as follows: Section 7.2 introduces the approach followed to figure out the translation process from MUCONTEXT contracts into XACML policies and requests. Then, we describe the translation process in Section 7.4 showing the equivalence between the two models. Finally, the conclusions reached through the transformation process of MUCONTEXT contracts into XACML representation are presented in Section 7.5.

7.2 Global Translation Process

As we explained in Section 2.7.4, XACML not only provides a format for expressing attribute-based access control policies, but also provides an architecture to implement the policy evaluation and decision enforcement.

To define the translation process, we study how XACML evaluates the policies (c.f., Figure E.1). Firstly, we found that the way in which XACML policies are written determines the evaluation behavior of the Policy Decision Point (PDP). Secondly, in order to trigger the evaluation process, an XACML access request is sent from a requester (e.g., context consumer) to the PDP. Therefore, because it is the producer who owns the resources and the consumer who wants to access the resources, we establish that the MUCONTEXT producer contract can be translated into the XACML policy format, and the MUCONTEXT consumer contract should be expressed according to the XACML request format.

We explain the global transformation process from MUCONTEXT contracts to XACML with two SPEM (Software Process Engineering MetaModel) diagrams [OMG, 2008]. Figures 7.1 and 7.2 are quite similar. They depict the MUCONTEXT contract transformation process from the definition to the translation into XACML format. They differ in the roles that perform the activities and the end process outputs.

Figure 7.1 shows the context producer side. The Context Owner (or Producer Application designer) performs the MUCONTEXT contract definition activity (using the MUCONTEXT contract editor, see Chapter 6). As a result, a producer MUCONTEXT contract is obtained that complies with the MUCONTEXT

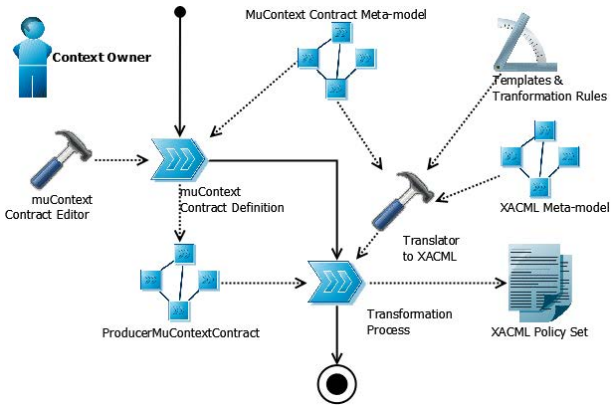


Figure 7.1: MuContext Producer Contract into XACML Global Process Translation

contract meta-model. Then, a model-to-text transformation process is executed to generate an XACML policy. A tool, such as ATL (ATL Transformation Language) [Foundation, 2014] or Acceleo (code generator Tool) [Obeo, 2014a], can be used to automate the translation process. For example, ATL is a model transformation language and toolkit that provides ways to produce a set of target models (XACML meta-model) from a set of source (MUCONTEXT contract) models. The end result of this process is an XACML policy set which expresses the same content and rules as the producer MUCONTEXT contract.

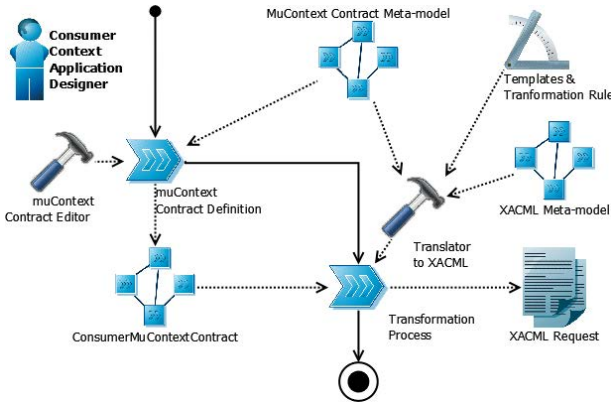


Figure 7.2: MuContext Consumer Contract into XACML Global Process Translation

Figure 7.2 represents the context consumer side. This process follows the same activities and tools as the previous one. However, the MUCONTEXT contract definition activity is executed by the Consumer Context Application Designer role. And, as a result of the entire process, an XACML Request is obtained, which expresses the same content and rules as the MUCONTEXT consumer contract.

7.3 Runtime Evaluation of a MUCONTEXT Contract

To see how the global process is integrated into the XACML architecture, we present the runtime access control evaluation with the following sequence diagram. Figure 7.3 shows the major actors involved in the XACML architecture (described in detail in [XACMLv3, 2013]). The green area in the center represents the XACML engine operation mode steps, and at the extremes are the context consumer and the context producer, which provide their respective MUCONTEXT contracts translated into an XACML request (on the consumer side) andan XACML policy (on the producer side). Because producers and consumers do not know each other, they go through a context manager, linked to an XACML engine for access control purpose.

XACML is intended to be suitable to a variety of application environments. The typical categories of attributes in the context are the subject, resource, action and environment. As they do not cover the vocabulary required to describe privacy and QoC, we define new categories and attributes to instantiate XACML policies and requests that cover the specifications of the MUCONTEXT contract.

The XACML engine time-based flow in the figure corresponds to the data-flow model of XACML described in [XACMLv3, 2013]. We assume that XACML policy sets are available to the PDP and represent the complete MUCONTEXT producer contract for a specified producer. At some time, the context consumer sends an XACML request (which represents a complete MUCONTEXT consumer contract) to the context manager. The latter then sends an access request to the XACML Engine to determine whether access to the context data is permitted or denied. Then, the PDP requests any additional information from the PIP, searches for a matching policy, evaluates the policy, and returns the authorization decision. Finally, the context manager fulfills the obligations and grants or refuses access to the context information.

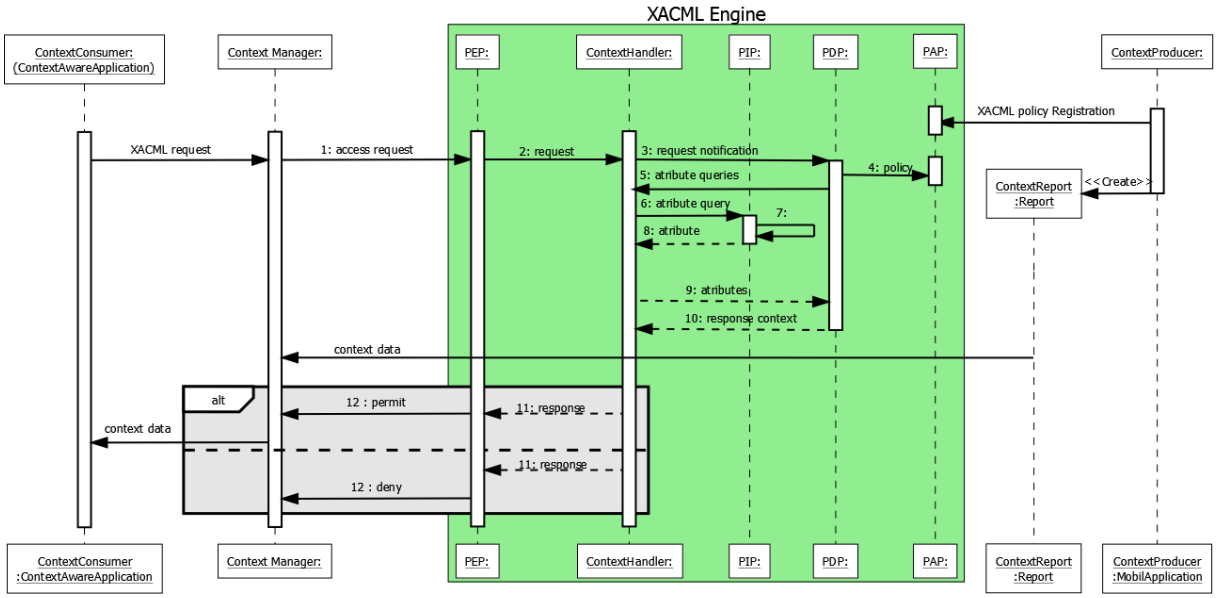


Figure 7.3: XACML Context Contract Evaluation - Sequence Diagram

7.4 Translation Process

In this section, we explain the translation of MUCONTEXT contracts into XACML policies and requests. For the translation process, we use the "Structure-Driven Approach" [Czarnecki and Helsen, 2003] technique, that have two distinct phases: the first phase is concerned with creating the hierarchical structure of the target model presented in Section 7.4.1, whereas the second phase sets the attributes and references in the target in Section 7.4.2.

As MUCONTEXT contracts are instances of MUCONTEXT contract meta-models and the XACML policies and requests are stored following a XML schema, we have decided to use a model-to-text transformation approach to obtain the XACML policy sets and XACML requests.

7.4.1 Hierarchical Structure of MUCONTEXT Contract Meta-Models

Concerning the first phase of the structure-driven approach, the hierarchical structure of MUCONTEXT contract meta-models is depicted in Figure 7.4 , (this figure represents the basic concepts of a producer or consumer contract, which comply with the MUCONTEXT contract meta-model of Figure 5.5 and 5.6). On the producer side, we have to correlate the MUCONTEXT producer contract with the structure of XACML policies depicted in Figure 2.10. On the consumer side, we have to correlate the MUCONTEXT consumer contract with the structure of XACML requests [XACMLv3, 2013].

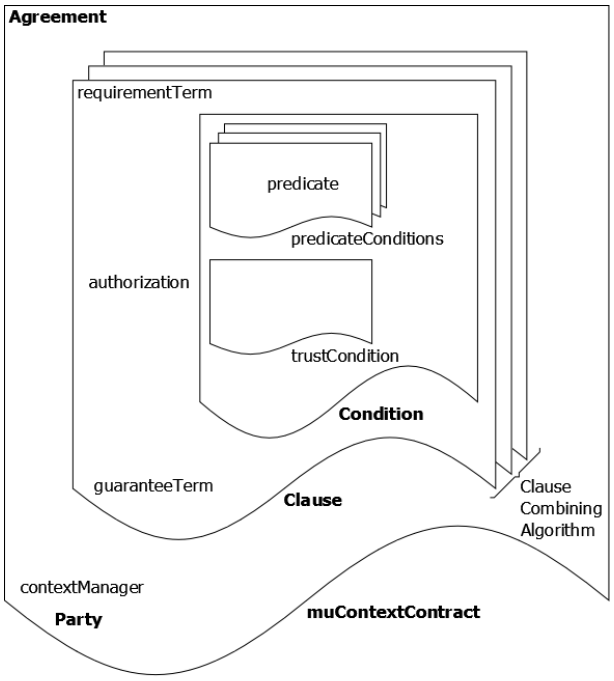


Figure 7.4: muContext Contract Hierarchical Strucure

Figure 7.5 shows the mapping between a producer contract and a XACML policy set. We can see that

a PolicySet represents a producer contract, where its targets represent the producer agreement and the context manager party. The XACML policy-combining algorithm is equivalent to the MUCONTEXT contract clause-combining algorithm. The XACML Policies represent the producerClause. And, each XACML Rule represents the clause conditions (i.e., trust condition and predicate condition). An XACML Policy Obligation represents the QoC guarantee (e.g., the level of QoC to deliver), and the XACML Policy Target represents the privacy requirements (e.g., the purpose dimension). Finally, the XACML rule effect represents the MUCONTEXT contract authorization.

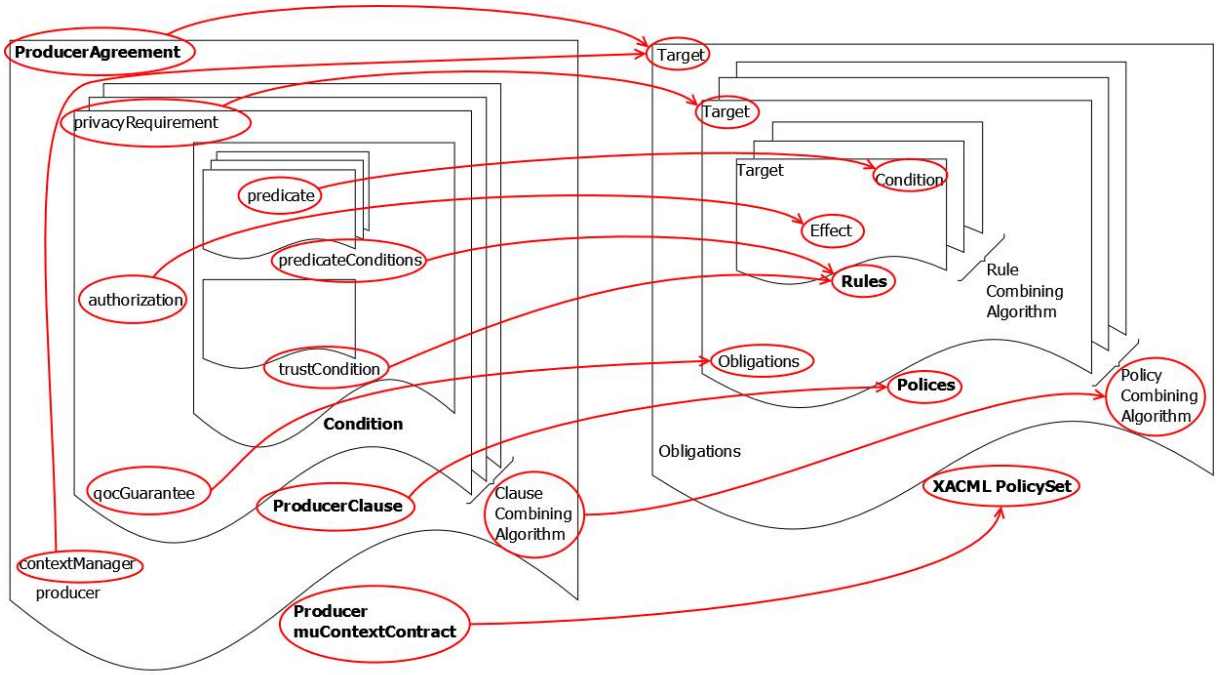


Figure 7.5: Translation of a MUCONTEXT Producer Contract into XACML Policy Set

Figure 7.6 shows the mapping between a MUCONTEXT consumer contract and a set of XACML requests. In this case, a set of individual XACML requests represents a consumer contract. Each ConsumerClause became an XACML request. To associate the different XACML requests to the same consumer contract, we repeat the consumer MUCONTEXT Contract attributes and ConsumerAgreement attributes for each XACML request. The qocRequirement, privacyGuarantee and Condition are translated into XACML request attributes (e.g., QoC requirement and privacy guarantee become attributes of the XACML request).

7.4.2 MuContext Contracts to XACML Categories, Attributes and Functions

Concerning the second phase of the structure-driven approach, we translate each dimension of the MUCONTEXT contracts into XACML categories, attributes and functions. For this purpose, we take advantage of the extensibility property of XACML, which enables to define new categories, attributes and

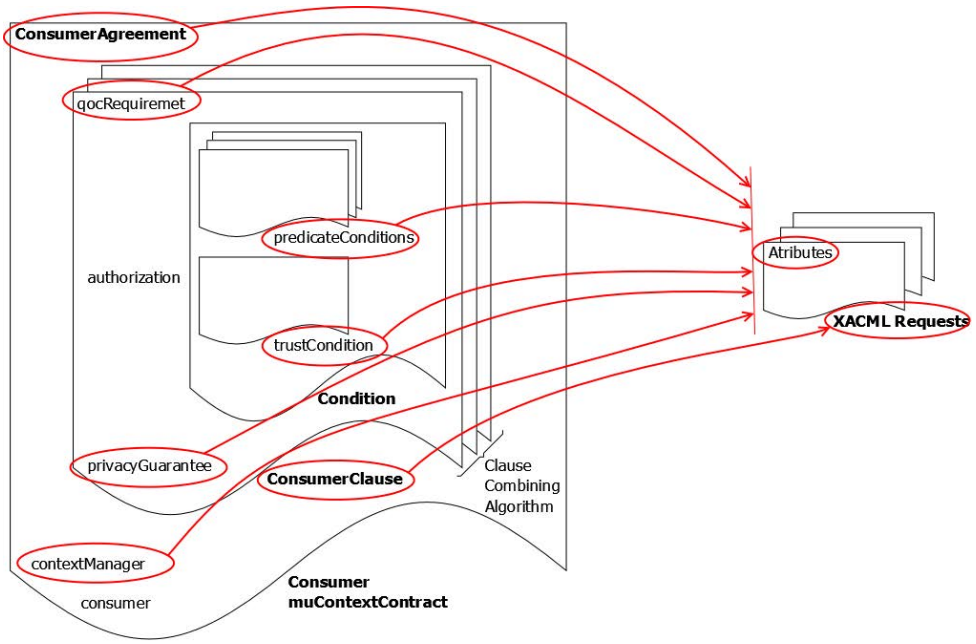


Figure 7.6: Translation of a MUCONTEXT Consumer Contract into XACML Request

functions.

The XACML specification defines a number of categories, attributes, functions, policy combining algorithms, *etc.* These XACML implemented elements are not exhaustive, therefore, we do not find all the elements necessary to express the MUCONTEXT contrats, such as, the visibility, the retention or the purpose, as well as, there are no attributes to define the trust and QoC. XACML extensibility property allows one to define these new concepts, for example, by creating an attribute purpose.id to identify the intentions of consumers. Our idea is to define new XACML categories, attributes and functions to express the concepts of privacy and QoC provided by the vocabulary of our meta-model.

In the hierarchy of the MUCONTEXT contract meta-model, we identify two parts: a Header and a Body. For each part of the contract, we present in Appendix F the summary tables with the equivalence between the MUCONTEXT contract entities and attributes and the XACML entities, cardinality, categories, attributes, and data types [8]. The first column MUCONTEXT Contract Entity contains an element of the MUCONTEXT contract meta-model, i.e., a class, a property or a relationship that must be translated into XACML. The second column XACML Entity contains the XACML element or attribute that corresponds to the MUCONTEXT Contract Entity and must be set up to write a XACML policy. The third column Cardinality is equal to the class relationships cardinality found in our meta-model and represents the number of times an element can appear in an XACML policy or request. The following columns Category, AttributeId and DataType contain the new URIs created to define the vocabulary

[8]Each XACML category, attribute and function has a URI value. These may be extended by the creation of new URIs associated with new semantics for these attributes.

defined by the MUCONTEXT contract meta-model. These values are the parameters used by the XACML policy evaluation process in which a <Target> element identifies the set of decision requests to be evaluated. In other words, a named attribute shall match an attribute if the values of their respective Category, AttributeId, DataType match[9]

Firstly, we present the equivalences between the MUCONTEXT producer contract and the XACML policy set, then the equivalences between the MUCONTEXT consumer contract and the XACML request.

- **Translation of a MUCONTEXT Producer Contract into an XACML Policy Set**

  The Header corresponds to the Agreement and Party classes of the meta-model. The ContextContract properties correspond to PolicySet element's atributes. The Agreement's atributes coresponds to targets elements of the PolicySet (c.f., Table F.1 in Appendix F).

  The different contract clauses constitute the Body. Table F.2 in Appendix F contains the equivalences with XACML elements and attributes. We split the body into three parts: the privacy terms, the QoC terms and the trust condition part shown in Tables F.3, F.4 and F.5 in Appendix F respectively.

  Table F.2 in Appendix F shows that a clause is mapped to a policy. Further, the most important thing to notice is that, all rule effects must be equal to the Clause.authorization value. And, Rule-CombiningAlgId must be set to deny-unless-permit, the reasons for this are explained in the example (Appendix G).

- **Translation of a MUCONTEXT Consumer Contract into an XACML Request**

  A consumer contract becomes an XACML request. The Header is composed by the consumerContextContract and the ConsumerAgreement attributes mapped in Table F.6 in Appendix F. The Body composed by the privacy terms, the QoC term and trust condition is mapped in Table F.7 in Appendix F.

In this section, we have shown how each MUCONTEXT contract concepts are translatable into XACML grammar elements. Concluding that, this translation allows to adopt a general-purpose tool (i.e., XACML) widely used to solve the specific problem of privacy, while using a specialized vocabulary for defining privacy and QoC for the IoT (i.e., MUCONTEXT contracts). In the next section, we show this translation in action through the motivating scenario.

7.5 Translation Process Outcomes

The conformity between the different parts of the context contract meta-model and the XACML policy and request is illustrated through the translation of simple MUCONTEXT contracts defined for the motivating scenario. The translation process example is shown in Appendix G. This example is divided into

[9]The attribute designator's Category MUST match, by identifier equality, the Category of the <Attributes> element in which the attribute is present. The attribute designator's AttributeId MUST match, by identifier equality, the AttributeId of the attribute. The attribute designator's DataType MUST match, by identifier equality, the DataType of the same attribute [XACMLv3, 2013].

two parts: one to show the translation process of a consumer contract, and the other one to show the producer contract translation process.

In this section, we present the conclusions reached through the transformation process of MUCONTEXT contracts into the XACML representation. Firstly, Section 7.5.1 shows the outcome of translation of a consumer contract into an XACML multi-request. Then Sections 7.5.2 and 7.5.3) present the conclusions of the translation of a producer contract into an XACML policy.

7.5.1 Consumer Context Request Outcome

In the example (see Appendix G.1, all the elements of the contract header are translated into an XACML request. However, the clause Combining Algorithm (Line [17]) should be fixed to deny_unless_permit which means, Indeterminate or NotApplicable must never be the result. If any decision is Permit, the result is Permit. Otherwise, the result is Deny. We include in the MUCONTEXT contract the possibility of combining clause decisions. Indeed, there may be cases in which a context consumer is not interested in consuming some context information when the producer does not meet the QoC requirements. This property is lost when the consumer contract is translated into several XACML requests, because, the XACML PDP can not combine the results of the different requests sent by the context manager. Likewise to the clause combining algorithm, the authorization SHOULD be fixed to Permit. There is no sense to create a Deny authorization clause if clauses can not be combined; this is equivalent to not sending the request.

Nowadays, there does not exist a mechanism in XACML to associate and combine the results of several requests. We consider that, the implementation of this process should be done in XACML, because, it is not part of the functions of the context manager or the context consumer application. This implementation is however out scope of this thesis.

7.5.2 Producer Policy Set (Contract Header) Outcome

As a result of the translation (See Appendix G.2), we remark that, all the elements presented in the producer contract header are translatable into an XACML policy set. Moreover, we show that, it is possible to formulate the target in a manner that the validations done are the same as those presented in the algorithms in Chapter 6.

The evaluation of the target in the policy set MUST match with the XACML request header to evaluate the clauses. Hence, it is important that the consumer and the producer interact with the same context manager to have the same concept references (e.g., the purpose help-in-danger has the same meaning for the producer and the consumer).

7.5.3 Producer Policy (Clause Example) Outcome

A MUCONTEXT contract represents a declaration of conditions and rules for defining privacy and QoC requirements and guarantees. As we have shown (See Apendix G.3), all the elements declared in the producer clause of a MUCONTEXT contract are translatable into an XACML policy set. Besides, it is possible to formulate the target to validate the same parameters as those presented in the algorithms in Chapter 6.

Nevertheless, in XACML, not only the declarations must be made but we must also include the matching process. The disadvantage is that to translate a producer contract into an XACML policy, not only we have to know the specific declarative elements of the contract but we also need to know the contract matching algorithm. Therefore, it is important to develop tools to do the translation automatically to minimize the possibility of errors in the interpretation.

From both producer and consumer contract translation processes, we can conclude that, there is a risk of a leak of information that violates privacy. This is due to the sequential processing of requests provided by XACML that does not allow us to combine the requests in order to take a single decision concerning one consumer. Therefore, we must trust that the consumer with multiple purposes will use the context information only for the purpose granted.

7.6 Translation Assessment

To validate the translation from MUCONTEXT contracts into XACML, we follow a qualitative validation. To verify that both the XACML policy and the XACML request obtained were correct regarding the syntax and logic, we developed a PDP written in Java using the Balana framework [WSO2, 2014]. With this PDP, we validate that we get the same access decision than with the MUCONTEXT matching process for the equivalent MUCONTEXT contracts.

Analysis

We present our analysis based on the observations made by applying the translation process to the producer and consumer contracts from the motivating scenario presented in Chapter 5. Our interest is to see if all the properties of the MUCONTEXT contracts and the behavior of the matching algorithm could be translated into XACML. As well, we take into consideration the number of lines generated in XACML compared to the number of lines of the original MUCONTEXT contracts.

We have shown through the translation of the MUCONTEXT contracts example presented in the Appendix G that the MUCONTEXT contract syntax is translatable into XACML grammar elements. However, to have an idea of the complexity of the XACML policy results, we have counted the number of lines of XML code required to write a MUCONTEXT contract with both languages. A consumer contract with two clauses requires 147 lines while the same contract in XACML requires 706 lines, which is four times bigger than the original consumer contract. The header of a producer contract requires 31 lines of code to be declared, although the translation into XACML requires 79 lines, which means 2.5 times bigger than the producer contract. This verbosity makes it difficult to find errors, and it is hard to be understood by people without proper tools.

Nevertheless, the translation of the matching algorithm is not the same, leaving a breach in people privacy. A way to ensure that a context consumer does not have hidden purposes (second intentions) is that the context consumers declare all their purposes at once. Therefore, when the decision is taken through the clauses combination algorithm, we can be sure that there is not a breach of privacy. The issue with XACML is that it can send only one request at the time, which means that context consumers can not declare all their purposes at once, opening a lack of privacy. This issue does not exist with MUCONTEXT

contracts as both participants can declare multiple guarantees and requirements simultaneously thanks to the symmetry of the MUCONTEXT contracts.

7.7 Conclusion

XACML is a wide used standard for enforcing access control. A dynamic evaluation of policies can be done with context data retrieved from the environment by using specialized Policy Information Points (PIP). We have tested the usability of XACML in the INCOME architecture for enforcing privacy requirements. Hence, this approach provides the advantage of reusing the XACML architecture and offers the necessary support to the problem of privacy and QoC in access control.

In order to use XACML implementations, MUCONTEXT contracts may be translated into XACML policies and requests. We favor MUCONTEXT contracts as they have been specifically designed to express QoC and privacy requirements for the IoT. As the XACML language was designed for a larger usage, the language is more verbose than MUCONTEXT contracts. For instance, the XACML request obtained after the translation process is four times bigger than the original consumer contract. This verbosity makes it difficult to find errors, and less understandable by people without proper tools. We favor, the MUCONTEXT contract language, which guides the contract developers with a specific vocabulary of privacy and QoC including also context predicates.

We propose that designers define their MUCONTEXT contracts, and that those contracts are translated into XACML policies and requests. With this proposal, XACML PIP should be extended to query specific MUCONTEXT attributes. Other parts of the XACML architecture can be reused unmodified.

We have detailed how MUCONTEXT contract concepts are translatable into XACML grammar elements. Concluding that, XACML leaves security breaches regarding the people privacy, although it is possible to translate the muContext contracts syntactically, though not all behavior of our matching algorithms is translatable, therefore an evolution of XACML to implement our solution is required. The current implementation enables to use a specialized vocabulary for defining privacy and QoC for the IoT applications of highly trusted context consumers.

In the future, we plan to automate the translation process from MUCONTEXT contracts into XACML policies and requests through model transformations. This would allow us to continue the validation with more use cases from the IoT.

Conclusions and Perspectives

Chapter 8

Conclusion and Perspectives

Contents

8.1 Conclusion

In the IoT, objects around us capture data in a pervasive way, putting our private life at risk. The users should therefore be able to manage their privacy. In the same time, IoT applications require more and more context data with adequate QoC to improve our comfort. Handling these both contradictory requirements in IoT middleware was the motivation of our work.

This thesis aimed to address the problem of managing privacy and QoC in the IoT. We consider that consumers and producers are decoupled participants, and that middleware should adapt continuously to new QoC and Privacy requirements. For those purposes, we have proposed models, algorithms and a software architecture. Indeed, it is required to develop autonomous mechanisms that find a trade-off between the users' privacy and the IoT applications' QoC requirements. We consider that the trust plays an important role in the establishment of agreements among producers and consumers relaxing the requirements of each part providing increased QoC and preserving the privacy simultaneously. We have adopted an approach based on MDE to create specific models for context contract definitions.

In Chapter 2, we compared the different principles and actions for protecting privacy proposed by several authors to obtain a list of privacy protection criteria. Moreover, we identified the dimensions to describe privacy and choose the models to be part of our solution to create contracts (QoCIM meta-model and the trust meta-model proposed by [Marie et al., 2013b] and [Neisse, 2012] respectively). Finally, we explained the reasons to choose XACML to implement contracts.

In Chapter 3, we discussed existing models, languages and frameworks that have proposed privacy protection mechanisms and that could be used in context-aware systems. We analyzed these existing works conformance to our proposed privacy protection criteria list.

We proposed in Chapter 4 a software architecture to enforce privacy and QoC contracts between producers and consumers in the IoT based on a set of specifications we extracted from a motivating scenario. Then, we analyzed the architecture in which producers and consumers interact in the case of mass market applications using the IoT.

We have introduced the notion of symmetrical contracts where producers and consumers independently express their requirements and guarantees in Chapter 5. We provided a meta-model to define MUCONTEXT contracts that supports both privacy and QoC definitions. Our contribution is a first step towards privacy and QoC enforcement. We evaluated the MUCONTEXT contracts using a qualitative approach showing its expressiveness. We implemented the MUCONTEXT contract editor to validate important requirements introduced in Section 4.3 such as including context situation in contracts definition; extending contracts with new requirements or guarantees on QoC and privacy; specifying various privacy requirements and guarantees.

These contracts have to be matched at run-time by an IoT middleware (i.e., context managers). Therefore, in Chapter 6, we have implemented the matching process between producers and consumers MUCONTEXT contracts proposed in this thesis. We developed the algorithms to determine whether context data delivering to consumers are authorized or not, according to the matching result. We have validated the correctness and efficiency of the presented algorithms following an experimental analysis approach. We have tested the algorithm for different input data and measured the number of matches that can be done during a period of time with different parameters.

Finally, in Chapter 7, we have proposed to translate producer MUCONTEXT contracts into XACML policies and consumer MUCONTEXT contracts to XACML requests, taking advantage of the use of existing XACML implementations. XACML is a good alternative to evaluate at run-time context information in the Policy Information Point (PIP) to take access control decisions, which is an interesting feature for the IoT. However, as XACML language was conceived for a larger usage, the language is too verbose. An advantage to use MUCONTEXT contracts is that both participants can declare multiple guarantees and requirements simultaneously thanks to the symmetry of the MUCONTEXT contracts which is not so easy with XACML only.

8.2 Perspectives

The object connected world has implications for the exposure of confidential information and personal data, with the consequent risk of misuse or infringement of privacy. This concern inevitably affects the relationship of people with the IoT. Definitely, privacy protection is a concern widely affecting the IoT, which brings them, much more care where and with whom people share their most sensitive information.

The study that we have presented in this thesis leaves us to consider some interesting perspectives for the continuation of our work.

The first research perspective is to assess deeply the proposals of our work regarding its expressiveness and performance. We propose to continue the validations through the definition of MUCONTEXT contracts for various IoT applications or to instantiate the MUCONTEXT contract meta-model in real world scenarios.

As the matching of MUCONTEXT contracts may be expensive in resources and time, a performance

validation is required to compare our implemented solution with the XACML alternative. Additionally, we suggest for future works, to use a caching method and the observer pattern, to avoid unnecessary MUCONTEXT contract matching evaluations. As well as to use multiprocessing to perform producer and consumer matches simultaneously to optimize the response time.

The second research perspective is to develop tools to facilitate the adoption of the MUCONTEXT contracts by integrating visibility and trust engines. Another aspect that should be considered in the future work is to develop a user-friendly interface that allows users or administrators to easily define requirements and guarantees related with privacy and QoC.

This thesis also opens up research opportunities directly related to the work carried out. One possible direction is to propose an adaptation approach for modeling transformation at run-time in order to create or modify MUCONTEXT contracts dynamically. Another area for future research is the investigation of the detection of privacy violations verifying whether the terms specified in MUCONTEXT contracts are respected.

Appendices

Appendix A

Publication List

A.1 Publications in journals, Conferences and Workshops

- **International journal papers**

  - Sophie Chabridon, Romain Laborde, Thierry Desprats, Arnaud Oglaza, Pierrick Marie, Samer Machara Marquez . A survey on addressing privacy together with quality of context for context management in the Internet of Things. Dans : Annals of Telecommunications, Springer, Numéro spécial Privacy-aware electronic society, Vol. 69 N. 1, p. 47-62, février 2014. Résumé Accès : http://link.springer.com/article/10.1007/s12243-013-0387-2 - http://oatao.univ-toulouse.fr/12723/

- **International conference papers**

  - Samer Machara Marquez, Sophie Chabridon, Chantal Taconet. Trust-based Context Contract Models for the Internet of Things, The 2013 International Symposium on Ubiquitous Intelligence and Autonomic Systems (UIAS 2013), In conjunction with The 10th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC-2013) and The 10th IEEE International Conference on Autonomic and Trusted Computing (ATC-2013), Sorrento Peninsula, Italy. December 2013.

- **National conference papers**

  - Jean-Paul Arcangeli, Sophie Chabridon, Denis Conan, Thierry Desprats, Romain Laborde, Sébastien Leriche, Léon Lim, Chantal Taconet, Raja Boujbel, Samer Machara Marquez, Pierrick Marie, Sam Rottenberg. Gestion de contexte multi-échelle pour l'Internet des objets (regular paper). Journées francophones Mobilité et Ubiquité (UBIMOB 2014), Sophia Antipolis, 05/06/2014-06/06/2014, Laboratoire i3S, (en ligne), June 2014.
    http://ubimob2014.sciencesconf.org/39139/document - http://oatao.univ-toulouse.fr/13105/

– Jean-Paul Arcangeli, Amel Bouzeghoub, Valérie Camps, Sophie Chabridon, Denis Conan, Thierry Desprats, Romain Laborde, Emmanuel Lavinal, Sébastien Leriche, Hervé Maurel, Mohamed Mbarki, André Péninou, Chantal Taconet, Pascale Zaraté, Raja Boujbel, Léon Lim, Samer Machara Marquez, Pierrick Marie, Clément Mignard, Arnaud Oglaza, Sam Rottenberg. Projet INCOME : INfrastructure de gestion de COntexte Multi-Echelle pour l'Internet des Objets (short paper). Conférence Francophone sur les Architectures Logicielles (CAL 2014), Paris, 10/06/2014-11/06/2014, ENSEEIHT, (on line), june 2014.

A.2 Contributions to the INCOME Project

In this project, I contributed to the writing of chapters 3 and 4 of **"Models of Quality and Pro-tection of Multiscalable Context Information"** Deliverable for task 4.1 of INCOME Project [Chabridon et al., 2013].

Additionally with my research work, I contributed to some INCOME Glossary definitions and the wording of the scenario "The Bike Sharing System" [INCOME, 2013b].

Appendix B

Extra-functional requirements to define Context Contracts

In this Appendix, we highlight the extra-functional requirements identified as essential for defining context contract meta-model and managing the agreements among context producers and context consumers. As shown in Table B.1, each requirement is exemplified by an **"Applicative case"** sub-section, which is a small story that illustrates the requirement. We present two mini-scenarios for each requirement. One from the context producer viewpoint and another from the context consumer viewpoint. Each **"mini-scenario"** sub-section follows the same structure:

- Description: The mini-scenario in a few sentences;
- Source of the stimulus: Some entity (a human, a computer system, or any other actor) that generated the stimulus;
- Stimulus: A condition that needs to be considered when it arrives at a system;
- Environment: The stimulus occurs within certain conditions. The system may be in an overloaded condition or may be running when the stimulus occurs, or some other condition may be true;
- Artifact: Some artifact is stimulated. This may be the whole system or some piece of it;

$R1$. Handling constantly evolving context data

Producer Mini-scenario: The middleware detects a situation that requires the activation or deactivation of some privacy policies.

- Stimulus Source: Context data
- Stimulus: Change in context situation
- Environment: The context manager is monitoring the stimulus.

 # Requirement. **Applicative case:** **Producer Mini-scenario:** • – Stimulus Source: – Stimulus: – Environment: – Artifact: **Producer Validation plan:** • – Model response: – Middleware response : – Response measure: **Consumer Mini-scenario:** ...(idem to Producer Mini-scenario)

Table B.1: Requirement Organization

- Artifact: Context management service, obligations service.

Producer Validation plan:

- Model response: There are no modifications of the context contract. Here, the context contract will determine the middleware response.
- Middleware response :
  - Evaluate context situation to determine which is the proper clause to apply.
  - Force the implementation of access rules and QoC modification and compel to delete context data (retention rule).
  - Translate context contract terms in a XACML Policies.
  - Deployment or deletion of the XACML Policies.
- Response measure :
  - Quantity of context data collected after activation.
  - Quantity of context data collected after deactication “is 0”.
  - Number of unauthorized purpose access after activation “is 0”.
  - Number of delete records after retention activation “equal to context data collected”.

Consumer Mini-scenario:
The middleware detect an specified situation that requires increases or decrease the QoC.

- Stimulus Source: Context data QoC level measurement or context end-user needs.
- Stimulus: Change in context situation or ensure the smooth running of the service when context data with unexpected QoC is received.
- Environment: The context manager is monitoring the stimulus triggered by this change.
- Artifact: A consumer context contract.

Consumer Validation plan:

- Model response: The context contract will determine the middleware response. The context contract does not change.
- Middleware response :
  - Increase or decrease QoC
- Response measure: QoC Level transmitted or received.

$R2$**. Handling evolutive requirements on QoC.**

Producer Mini-scenario:
An context owner changes the clauses of an already accepted context contract; theses modifications concern only the situation and its respective QoC.

- Stimulus Source: Context owner.
- Stimulus: Source wants to change the clauses of context contract.
- Environment: At runtime
- Artifact: Producer Context Contract and Consumer Context Contract.

Producer Validation plan:

- Model response:
  - modified Producer context contract.
  - proposed modified consumer context contract.
- Middleware response :
  - Send proposed modifications to the context consumer.
  - Implement producer context contract changes.
  - Acceptation or rejection of consumer context contract proposed modifications.

  - If modifications are accepted, implement consumer context contract changes.

- Response measure:
  - QoC Level transmitted for each situation.
  - Quantity of context data collected after changes implementation.

Consumer Mini-scenario: An end-user wants to change the clauses of an already accepted context contract; modifications concern only parameter values.

- Stimulus Source: Context Consumer
- Stimulus: Increase or decrease QoC, visibility or retention.
- Environment: Under normal execution, at runtime.
- Artifact: Consumer context contract and producer context contract.

Consumer Validation plan:

- Model response:
  - Modified consumer context contract.
  - Proposed modified producer context contract.
- Middleware response :
  - Send proposed modifications to the context producer (notification).
  - If modifications are accepted, implement consumer and producer context contract changes.
- Response measure:
  - Context data transmitted with new parameters.
  - Quantity and QoC data collected after changes implementation.

$R3$**. Privacy Policy Language that supports QoC definitions.**

Producer Mini-scenario:
To hide details that compromise his/her privacy, a context owner defines in his/her context contract how much QoC should have the information collected on him/her.

- Stimulus Source: context owner
- Stimulus: Protect privacy
- Environment: Under normal execution or at runtime.

- Artifact: Producer context contract.

Producer Validation plan:

- Model response: New or modified Producer context contract with QoC definitions.
- Middleware response : Production of context data with QoC modification.
- Response measure:
  - Quantity of context data collected after activation.
  - Quantity of context data collected after deactication “is 0”.
  - Number of context data collected with unexpected QoC “is 0”.

Consumer Mini-scenario:
To provide a smooth operation of application, a context consumer defines in its context contract how much QoC needs over collected information.

- Stimulus Source: Context Consumer Designer.
- Stimulus: Application process.
- Environment: Design time or runtime.
- Artifact: Consumer context contract.

Consumer Validation plan:

- Model response: New or modified consumer context contract with QoC definitions.
- Middleware response : Not applicable.
- Response measure:
  - Quantity of context data collected after activation.
  - Quantity of context data collected after deactication “is 0”.
  - Number of context data collected with unexpected QoC “is 0”.

$R4$**. Evolution of Privacy Requirements**

Producer Mini-scenario:
An context owner changes the clauses of an already accepted context contract;

- Stimulus Source: Context owner.
- Stimulus: Source wants to change the clauses of context contract.

- Environment: At runtime
- Artifact: Producer Context Contract

Producer Validation plan:

- Model response:
  - Modified producer context contract.
- Middleware response :
  - Implement producer context contract changes.
  - Acceptation or rejection of consumer context contract proposed modifications.
- Response measure:
  - Quantity of context data collected after changes implementation.

Consumer Mini-scenario:
An end-user wants to change the clauses of an already accepted context contract;

- Stimulus Source: Context end-user or Context consumer application.
- Stimulus: Source wants to change the clauses of context contract.
- Environment: At runtime.
- Artifact: Consumer Context Contract

Consumer Validation plan:

- Model response: Modified producer context contract.
- Middleware response : modifications.
  - Implement consumer context contract changes.
  - Stop context data consumption until agreement of the context owners.
- Response measure:
  - Quantity of context data collected after changes implementation.

$R5$. **Specifying and enforcing privacy preferences.**

Producer Mini-scenario:
An context owner create or modify the clauses of his/her context contract to define his/her privacy terms; theses terms concern the access control rules to context data based on trust levels, the purpose, the retention and in which situations the respective privacy terms are applied.

- Stimulus Source: Context Owner.
- Stimulus: Source create the terms in the producer context contract to enforce his privacy preferences.
- Environment: At runtime or design time.
- Artifact: Producer Context Contract and Context-aware Application.

Producer Validation plan:

- Model response: New or modified producer context contract
- Middleware response :
  - Translate context contract terms in a XACML Policies.
  - Deployment or deletion of the XACML Policies.
  - Force the implementation of access rules and QoC modification and compel to delete context data (retention rule).
- Response measure:
  - Quantity of context data collected after activation.
  - Quantity of context data collected after deactication “is 0”.
  - Number of unauthorized purpose access after activation “is 0”.
  - Number of delete records after retention activation “equal to context data collected”.

Consumer Mini-scenario:
An context consumer application apply the rules describe in the producer context contract to respect the requirements of the Context owner. For example, enforce new access control rules, only use the context data for the purpose allowed, delete context data after expiration time.

- Stimulus Source: Producer Context Contract
- Stimulus: Source wants to enforce the context consumer behavior.
- Environment: At runtime
- Artifact: Context Consumer Application.

Consumer Validation plan:

- Model response: The consumer context contract does not change.
- Middleware response :
  - The consumer application implements the rules of access control
  - The consumer application checks if the time of retention have no expired and delete the context data if it is the case.
  - The consumer application does not use the context data for a different purpose that those declare previously.
- Response measure:
  - Quantity of context data collected after by unauthorized end-users.
  - Quantity of context data collected after deactication “is 0”.
  - Number of unauthorized purpose access after activation “is 0”.
  - Number of delete records after retention activation “equal to context data collected”.

Appendix C

Motivating Scenario Simulation

We simulate this use case generating some context data with random numbers in Microsoft Excel to illustrate the dynamism of the context environment. Figure C.1 shows a simulation of the location of the users' application in the Cartesian Coordinate System.

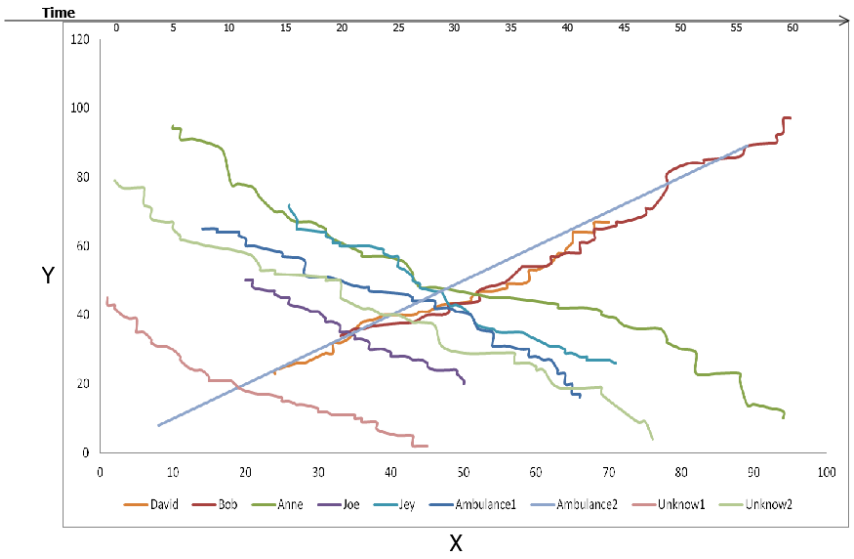


Figure C.1: GPS Location Simulation

The time Table C.2 shows the evolution of others users' proximity to David.

Figure C.3 illustrates how the modification of the QoC can be useful to hide sensitive data. More figures and tables related with this example are in the Annexe C

Table C.1 show a example of context reports generated for T1 period of time generating

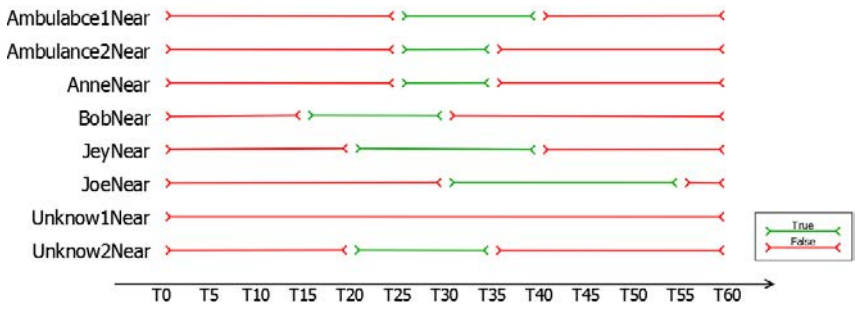


Figure C.2: Evolution of the proximity of other users respects to David (Simulation)

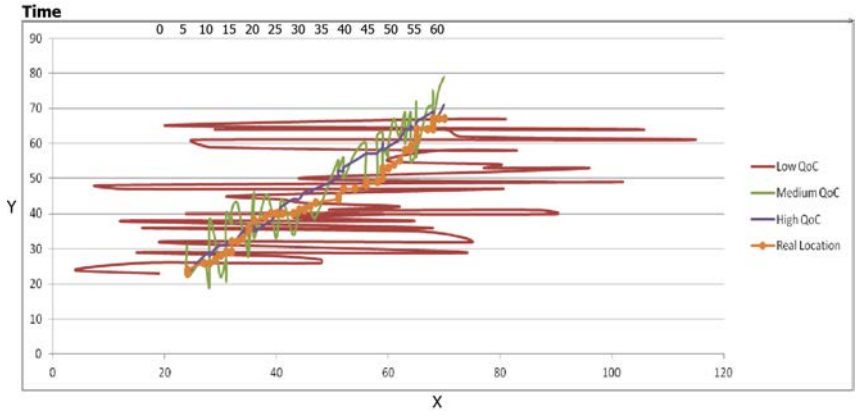


Figure C.3: Different QoC delivered by David (Location obfuscation example)

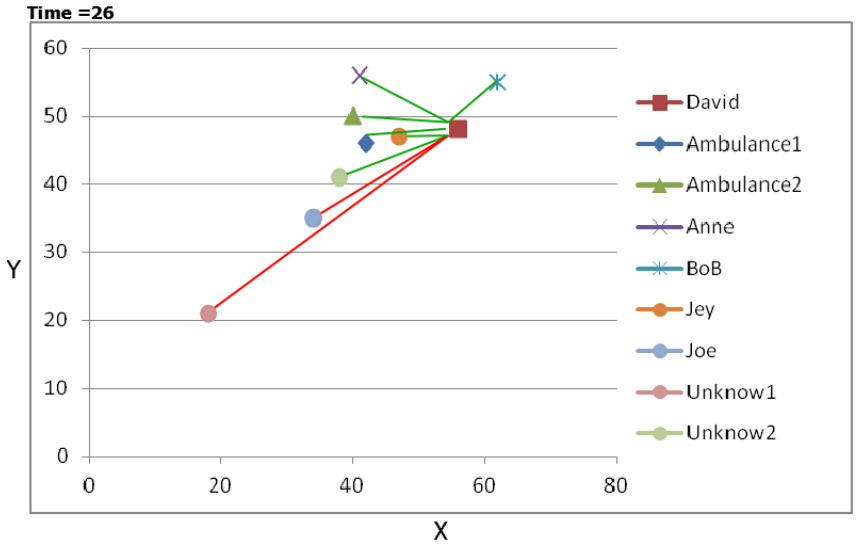


Figure C.4: Distance calculation between David and Other End-users at minute 26 Diagram

Time	Observable	Value	Owner	QoC
1	AfterWork	FAUX	David	100
1	doingExcercice	FAUX	David	100
1	Emergency	FAUX	David	100
1	GoingHome	FAUX	David	100
1	JeyNear	FAUX,41.0121933088198	App	100
1	BobNear	FAUX,49.5782210249622	App	100
1	JoeNear	FAUX,51.0783711564885	App	100
1	Ambulance1Near	FAUX,56.0357029044876	App	100
1	Unknow2Near	FAUX,63.285069329187	App	100
1	AnneNear	FAUX,66.2117814289874	App	100
1	Unknow1Near	FAUX,72.4223722339996	App	100
1	Ambulance2Near	FAUX,87.6812408671319	App	100
1	Location(X,Y)	14,65	Ambulance1	100
1	Location(X,Y)	8,5	Ambulance2	100
1	Location(X,Y)	10,95	Anne	100
1	Location(X,Y)	33,34	BoB	100
1	RideBike	FAUX	David	100
1	Location(X,Y)	70,67	David	100
1	Location(X,Y)	70,67	David	50
1	Location(X,Y)	70,79	David	90
1	Location(X,Y)	70,71	David	99
1	Location(X,Y)	71,26	Jey	100
1	Location(X,Y)	50,20	Joe	100
1	Location(X,Y)	1,45	Unknow1	100
1	Location(X,Y)	76,4	Unknow2	100

Table C.1: Context Reports during T1

Source	X	Y	Distance	isNear()
Ambulance1	42	46	14.14	True
Ambulance2	40	50	16.12	True
Anne	41	56	17	True
BoB	62	55	9.21	True
Jey	47	47	9.05	True
Joe	34	35	25.55	False
Unknow1	18	21	46.61	False
Unknow2	38	41	19.31	True

Table C.2: Distance calculation between David and Other End-users at minute 26

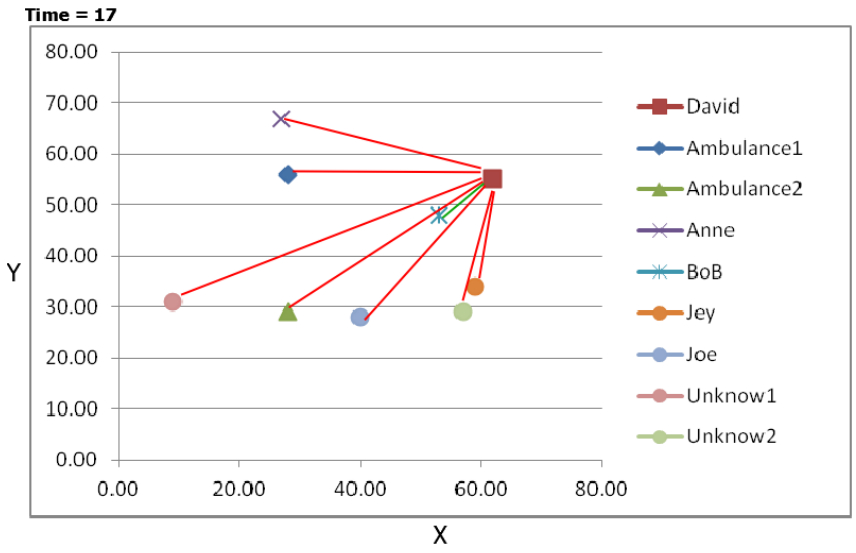
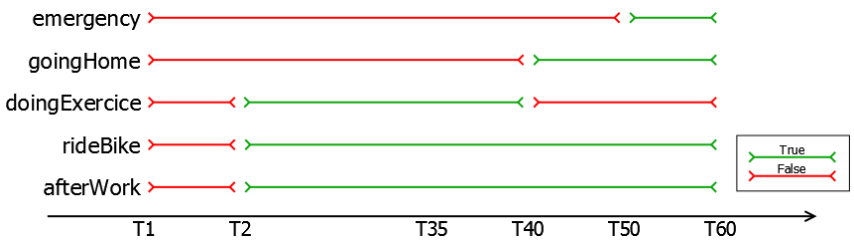


Figure C.5: Distance calculation between David and Other End-users at minute 17 Diagram



Source	X	Y	Distance	isNear()
Ambulance1	28.00	56.00	34.01	False
Ambulance2	28.00	29.00	42.80	False
Anne	27.00	67.00	37	False
BoB	53.00	48.00	11.40	True
Jey	59.00	34.00	21.21	False
Joe	40.00	28.00	34.82	False
Unknow1	9.00	31.00	58.18	False
Unknow2	57.00	29.00	26.47	False

Table C.3: Distance calculation between David and Other End-users at minute 17

Appendix D

Input and Output Data for each Test of Correctness

Table D.1 and D.2 show the input and output data for each test executed that follows the Paths 1 and 2 respectively.

Tables D.3 and D.4 show the input and output data for each test executed that follows the Paths 3 and 4 respectively.

Table D.5 shows the input and outputs data for each test executed that follows the Paths 2 and 4.

test id	Input					
	Consumer Contract			Producer Contract		
	getHas_consumer_agreement() .setClauseCombiningAlgId()	**setAuthorization()**	**getQoc_requirement() .getHas_qoc() .setId()**	**getHas_producer_agreement() .setClauseCombiningAlgId()**	**setAuthorization()**	**getQoc_guarantee() .getHas_qoc() .setId()**
1X1_01	deny_overrides	deny	QoC:1	deny_overrides	deny	QoC:1
1X1_05	deny_overrides	deny	QoC:1	deny_overrides	deny	QoC:1
1X1_09	deny_overrides	deny	QoC:1	deny_overrides	deny	QoC:2
1X1_10	deny_overrides	deny	QoC:1	deny_overrides	permit	QoC:2
1X1_13	deny_overrides	deny	QoC:1	deny_overrides	deny	QoC:2
1X1_14	deny_overrides	deny	QoC:1	deny_overrides	permit	QoC:2
1X1_17	deny_overrides	deny	QoC:1	permit_overrides	deny	QoC:1
1X1_21	deny_overrides	deny	QoC:1	permit_overrides	deny	QoC:1
1X1_25	deny_overrides	deny	QoC:1	permit_overrides	deny	QoC:2
1X1_26	deny_overrides	deny	QoC:1	permit_overrides	permit	QoC:2
1X1_29	deny_overrides	deny	QoC:1	permit_overrides	deny	QoC:2
1X1_30	deny_overrides	deny	QoC:1	permit_overrides	permit	QoC:2
1X1_03	permit_overrides	permit	QoC:1	deny_overrides	deny	QoC:1
1X1_07	permit_overrides	permit	QoC:1	deny_overrides	deny	QoC:1
1X1_11	permit_overrides	permit	QoC:1	deny_overrides	deny	QoC:2
1X1_12	permit_overrides	permit	QoC:1	deny_overrides	permit	QoC:2
1X1_15	permit_overrides	permit	QoC:1	deny_overrides	deny	QoC:2
1X1_16	permit_overrides	permit	QoC:1	deny_overrides	permit	QoC:2
1X1_19	permit_overrides	permit	QoC:1	permit_overrides	deny	QoC:1
1X1_23	permit_overrides	permit	QoC:1	permit_overrides	deny	QoC:1
1X1_27	permit_overrides	permit	QoC:1	permit_overrides	deny	QoC:2
1X1_28	permit_overrides	permit	QoC:1	permit_overrides	permit	QoC:2
1X1_31	permit_overrides	permit	QoC:1	permit_overrides	deny	QoC:2
1X1_32	permit_overrides	permit	QoC:1	permit_overrides	permit	QoC:2

test id	Output			
	qEvaluation	**cClauseVerified**	**cDecision**	**forward**
1X1_01	true	true	deny	false
1X1_05	true	true	deny	false
1X1_09	false	false	deny	false
1X1_10	false	false	deny	false
1X1_13	false	false	deny	false
1X1_14	false	false	deny	false
1X1_17	true	true	deny	false
1X1_21	true	true	deny	false
1X1_25	false	false	deny	false
1X1_26	false	false	deny	false
1X1_29	false	false	deny	false
1X1_30	false	false	deny	false
1X1_03	true	true	deny	false
1X1_07	true	true	deny	false
1X1_11	false	false	deny	false
1X1_12	false	false	deny	false
1X1_15	false	false	deny	false
1X1_16	false	false	deny	false
1X1_19	true	true	deny	false
1X1_23	true	true	deny	false
1X1_27	false	false	deny	false
1X1_28	false	false	deny	false
1X1_31	false	false	deny	false
1X1_32	false	false	deny	false

Table D.1: Consumer Requirement Validation Test (Inputs and Outputs) Path 1

test id	Input					
	Consumer Contract			Producer Contract		
	getHas_consumer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getQoc_requirement() .getHas_qoc() .setId()	getHas_producer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getQoc_guarantee() .getHas_qoc() .setId()
1X1_02	deny_overrides	deny	QoC:1	deny_overrides	permit	QoC:1
1X1_04	permit_overrides	permit	QoC:1	deny_overrides	permit	QoC:1
1X1_06	deny_overrides	deny	QoC:1	deny_overrides	permit	QoC:1
1X1_08	permit_overrides	permit	QoC:1	deny_overrides	permit	QoC:1
1X1_18	deny_overrides	deny	QoC:1	permit_overrides	permit	QoC:1
1X1_20	permit_overrides	permit	QoC:1	permit_overrides	permit	QoC:1
1X1_22	deny_overrides	deny	QoC:1	permit_overrides	permit	QoC:1
1X1_24	permit_overrides	permit	QoC:1	permit_overrides	permit	QoC:1

test id	Output			
	qEvaluation	cClauseVerified	cDecision	forward
1X1_02	true	true	permit	true
1X1_04	true	true	permit	true
1X1_06	true	true	permit	true
1X1_08	true	true	permit	true
1X1_18	true	true	permit	true
1X1_20	true	true	permit	true
1X1_22	true	true	permit	true
1X1_24	true	true	permit	true

Table D.2: Consumer Requirement Validation Test (Inputs and Outputs) Path 2

test id	Input					
	Consumer Contract			Producer Contract		
	getHas_consumer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getPrivacy_guarantee() .getHas_purpose() .setId()	getHas_producer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getPrivacy_requirement() .getHas_purpose() .setId()
1X1p_10	deny_overrides	deny	Purpose:1	deny_overrides	permit	Purpose:2
1X1p_12	permit_overrides	permit	Purpose:1	deny_overrides	permit	Purpose:2
1X1p_14	deny_overrides	deny	Purpose:1	deny_overrides	permit	Purpose:2
1X1p_16	permit_overrides	permit	Purpose:1	deny_overrides	permit	Purpose:2
1X1p_26	deny_overrides	deny	Purpose:1	permit_overrides	permit	Purpose:2
1X1p_28	permit_overrides	permit	Purpose:1	permit_overrides	permit	Purpose:2
1X1p_30	deny_overrides	deny	Purpose:1	permit_overrides	permit	Purpose:2
1X1p_32	permit_overrides	permit	Purpose:1	permit_overrides	permit	Purpose:2

test id	Output					
	qEvaluation	cClauseEvaluation	cDecision	pClauseEvaluation	pDecision	forward
1X1p_10	true	true	permit	false	deny	false
1X1p_12	true	true	permit	false	deny	false
1X1p_14	true	true	permit	false	deny	false
1X1p_16	true	true	permit	false	deny	false
1X1p_26	true	true	permit	false	deny	false
1X1p_28	true	true	permit	false	deny	false
1X1p_30	true	true	permit	false	deny	false
1X1p_32	true	true	permit	false	deny	false

Table D.3: Producer Requirement Validation Test (Inputs and Outputs) Path 3

test id	Input					
	Consumer Contract			Producer Contract		
	getHas_consumer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getPrivacy_guarantee() .getHas_purpose() .setId()	getHas_producer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getPrivacy_requirement() .getHas_purpose() .setId()
1X1p_02	deny_overrides	deny	Purpose:1	deny_overrides	permit	Purpose:1
1X1p_04	permit_overrides	permit	Purpose:1	deny_overrides	permit	Purpose:1
1X1p_06	deny_overrides	deny	Purpose:1	deny_overrides	permit	Purpose:1
1X1p_08	permit_overrides	permit	Purpose:1	deny_overrides	permit	Purpose:1
1X1p_18	deny_overrides	deny	Purpose:1	permit_overrides	permit	Purpose:1
1X1p_20	permit_overrides	permit	Purpose:1	permit_overrides	permit	Purpose:1
1X1p_22	deny_overrides	deny	Purpose:1	permit_overrides	permit	Purpose:1
1X1p_24	permit_overrides	permit	Purpose:1	permit_overrides	permit	Purpose:1

test id	Output					
	qEvaluation	cClauseVerified	cDecision	pClauseVerified	pDecision	forward
1X1p_02	true	true	permit	true	permit	true
1X1p_04	true	true	permit	true	permit	true
1X1p_06	true	true	permit	true	permit	true
1X1p_08	true	true	permit	true	permit	true
1X1p_18	true	true	permit	true	permit	true
1X1p_20	true	true	permit	true	permit	true
1X1p_22	true	true	permit	true	permit	true
1X1p_24	true	true	permit	true	permit	true

Table D.4: Producer Requirement Validation Test (Inputs and Outputs) Path 4

test id	Input					
	Consumer Contract					
	getHas_consumer_agreement() .setClauseCombiningAlgId()	setAuthorization()	getQoc_requirement() .getHas_qoc() .setId()	getPrivacy_guarantee() .getHas_purpose() .setId()	getPrivacy_guarantee() .getHas_visibility() .setId()	getPrivacy_guarantee() .getHas_retention() .setId()
1X1qpvr_001	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_033	permit_overrides	deny	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_065	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_097	permit_overrides	deny	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_129	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_161	deny_overrides	deny	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_193	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_225	deny_overrides	deny	QoC:1	Purpose:1	Visibility:1	Retention:1

test id	Input					
	Producer Contract					
	setClauseCombiningAlgId()	setAuthorization()	getQoc_guarantee() .getHas_qoc() .setId()	getPrivacy_requirement() .getHas_purpose() .setId()	getPrivacy_requirement() .getHas_visibility() .setId()	getPrivacy_requirement() .getHas_retention() .setId()
1X1qpvr_001	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_033	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_065	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_097	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_129	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_161	permit_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_193	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1
1X1qpvr_225	deny_overrides	permit	QoC:1	Purpose:1	Visibility:1	Retention:1

TestId	Output						
	qEvaluation	cClauseVerified	cDecision	vEvaluation & pEvaluation & rEvaluation	pClauseVerified	pDecision	forward
1X1qpvr_001	true	true	permit	true&true&true	true	permit	true
1X1qpvr_033	true	true	permit	true&true&true	true	permit	true
1X1qpvr_065	true	true	permit	true&true&true	true	permit	true
1X1qpvr_097	true	true	permit	true&true&true	true	permit	true
1X1qpvr_129	true	true	permit	true&true&true	true	permit	true
1X1qpvr_161	true	true	permit	true&true&true	true	permit	true
1X1qpvr_193	true	true	permit	true&true&true	true	permit	true
1X1qpvr_225	true	true	permit	true&true&true	true	permit	true

Table D.5: Consumer and Producer Requirements Validation Test (Inputs and Outputs) Paths 2 and 4

Appendix E

XACML 3.0 Background

In XACML, the core logic related to policy evaluation resides in a software component called "Policy Decision Point" (PDP). XACML PDP takes two inputs and gives a single output. The inputs are "XACML Policies" and "XACML Request". The output per request can be one of the following:

- **Permit** - Request is evaluated against all applicable policies given to XACML PDP and the request is authorized to carry on the operation/actions
- **Deny** - Request is evaluated against all applicable policies given to XACML PDP and the request is not authorized to carry on the operation/actions
- **Not Applicable** - XACML PDP did not find any applicable policy for a given request and the request is not evaluated in XACML PDP.

The XACML PDP can consider multiple policies and evaluate a request against all those policies. The PDP evaluates policies independently. The result of evaluating a single policy is the same as the results stated above (Permit, Deny, or Not Applicable). The "Policy Combining Algorithm" handles the problem of how the outputs from multiple policies are combined together. This algorithm is responsible for combining outcomes from multiple policy evaluations.

- **permit-override** - If at least one policy is evaluated as "permit", the integrated output will also be "permit".
- **deny-override** - If at least one policy is evaluated as "deny", the integrated output will also be "deny".

Figure E.1, depicts the above mentioned behavior.

XACML Request

An XACML request contains information necessary to take authorization decision when evaluated with respect to a given policy. A policy has an XML element known as "Target". The "Target" element decides

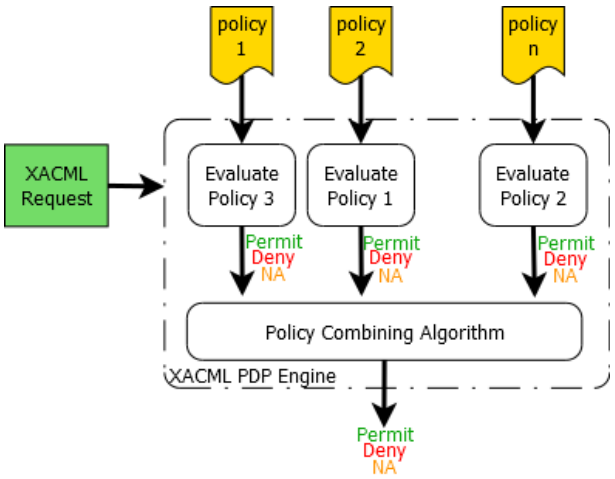


Figure E.1: XACML PDP

whether a particular policy is applicable to a given request [XACMLv3, 2013]. The "Target" element contains 4 sub-elements. They are depicted in Figure E.2. The Target sub-element's attribute values are matched with the incoming request attribute values. If there is match, the XACML PDP decides that the request should be evaluated against the complete policy.

Figure E.2: XACML Target Structure

Let us look at what each of the sub-elements represents.

Subjects - This is a parent element which contains 1 or more "Subject" elements. A Subject element usually represents the identity of the entity, which performs an action. Within the "Subject" element it is necessary to define a matching criterion per policy. For that, another sub-element called "SubjectMatch" is used. Within the "SubjectMatch" the logic of how to match elements is defined. A SubjectMatch element contains 2 parameters:

- Value of the subject attribute

- Name of the subject attribute

We can directly specify how to compare attribute values with the relevant data type (in the policy). When XACML needs to retrieve attribute values from incoming Subject (in request), it uses "AttributeDesignator". In the attribute designator, we specify the fully qualified name of the attribute and its type. Though, this is essential to retrieve attribute values from the request.

Resources - "Resources", is a parent element which represents a collection of "Resource" elements. A "Resource" element usually represents the actual resource which the user is trying to access.

Actions - "Actions", is a parent element which represents a collection of "Action" elements. A "Action" element represents user's activity on a resource (e.g.:- read, write, execute, *etc*).

The process of comparing Resource/Action values in a policy and Resource/Action values in XACML request is quite similar to Subject comparison described above.

Environment- This element is used to define system wide attributes which impact the authorization. E.g. If we want to apply certain policies only on a given domain, we can use this element. In the "Environment" element we can specify the domain name and the domain name will also be sent in the XACML request. We write the policy in such a way that it will be evaluated only if domain names are equivalent (we can use XACML built-in functions to do comparison).

Listing E.1 shows an example of an XACML request in 3.0 versions. This is a request of the Be4ME context contract manager to read the GPS location.

Listing E.1: XACML 3.0 Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<Request
            xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
                  http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
            ReturnPolicyIdList="false"
            CombinedDecision="false"
            xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Attributes
            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
            <Attribute
                  IncludeInResult="false"
                  AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
                  <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        >Be4MEContextContractManager
                  </AttributeValue>
            </Attribute>
      </Attributes>
      <Attributes
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
            <Attribute
                  IncludeInResult="false"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
                  <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                        >gpsLocation
                  </AttributeValue>
            </Attribute>
      </Attributes>
      <Attributes
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
            <Attribute
                  IncludeInResult="false"
                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
                  <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        >read
                  </AttributeValue>
            </Attribute>
      </Attributes>
      <Attributes
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" />
</Request>
```

Appendix F

MuContext Contracts to XACML Translation Tables

Context Contract Entity	XACML Entity	Cardinality	Category	AttributeId	DataType
ProducerContextContract	<PolicySet>(E)	1..1	-	-	-
ContextContract.id	PolicySetId (A)	1..1	-	-	anyURI
ContextContract.version	Version (A)	0..1	-	-	string
ContextContract.description	<Description> (E)	0..1	-	-	string
Agreement .clauseCombiningAlgId	PolicyCombiningAlgId (A)	1..1	-	-	-
ContextContract.context_manager.id	<Target> (E)	1..1	urn:poc-qoc:ccm:party	urn:poc-qoc:ccm:context-contract:context-manager:party:id	string
Agreement.Observable.name	<Target> (E)	1..1	urn:poc-qoc:ccm: agreement	urn:poc-qoc:ccm:agreement:has-observable:observable:name	string
ProducerContextContract.producer.id	<Target> (E)	0..1	urn:poc-qoc:ccm:party	“urn:poc-qoc:ccm:context-contract:producer:party:id” “urn:poc-qoc:ccm:context-contract:consumer:party:id”	string
Clause	PolicyIdReference	1..*	-	-	-

Table F.1: ProducerContextContract to XACML Policy Set (“Header”).(E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	**XACML Entity**	**Cardi-nality**	**Category**	**AttributeId**	**Function**	**DataType**	**Note**
Clause (C)	<Policy> (E)	1..*	-	-	-	-	-
Clause.id (P)	PolicyId (A)	1..1	-	-	-	string	-
Clause.authorization (P)	Effect (A)	1..1	-	-	-	-	All rule Effects MUST be equal to Clause.authorization value. And, Rule-CombiningAlgId MUST be set to deny-unless-permit

Table F.2: ProducerClauses to XACML policies (“Body”)- (E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	**XACML Entity**	**Cardinality**	**Category**	**AttributeId**	**Function**	**DataType**	**Note**
PrivacyTerm (C)	Policy <Target> (E)	1..1	-	-	-	-	-
Visibility.id (P)	<AttributeValue> <AttributeDesignator> (E)	0..1	urn:poc-qoc:ccm:privacy-term:visibility	urn:poc-qoc:ccm:privacy-term:visibility:id	-	string	MatchId attribute can be set up according the application needs.
Visibility.description (P)	<AttributeValue> <Description> (E)	0..1	-	-	-	string	optional
Retention.id (P)	<AttributeValue> <AttributeDesignator> (E)	0..1	urn:poc-qoc:ccm:privacy-term:retention	urn:poc-qoc:ccm:privacy-term:retention:id	-	string	MatchId attribute can be set up according the application needs.
Retention.description (P)	<AttributeValue> <Description> (E)	0..1	-	-	-	string	optional
Purpose.id (P)	<AttributeValue> <AttributeDesignator> (E)	0..1	urn:poc-qoc:ccm:privacy-term:purpose	urn:poc-qoc:ccm:privacy-term:purpose:id	-	string	MatchId attribute can be set up according the application needs.
Purpose.description (P)	<AttributeValue> <Description> (E)	0..1	-	-	-	string	optional

Table F.3: PrivacyTerm Requirement to XACML Policy Target (“Body”)- (E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	XACML Entity	Cardi-nality	Category	AttributeId	Function	DataType	Note
QoCTerm (C)	Policy <Target> (E)	1..1	-	-	-	-	-
QoCCondition.id (P)	<AttributeValue> <AttributeDesig-nator> (E)	0..1	urn:poc-qoc:ccm: qoc-term: qoc-condition	urn:poc-qoc:ccm: qoc-term: qoc-condition:id	-	string	MatchId attribute can be set up according the application needs.
QoCIM.Indicator.name (P)	<AttributeValue> <AttributeDesig-nator> (E)	0..1	urn:poc-qoc:ccm: qoc-term: qoc-condition	urn:poc-qoc:ccm: qoc: indicator:name	-	string	MatchId attribute can be set up according the application needs.
QoCCondition. qocLowerBound (P)	<AttributeValue> <AttributeDesig-nator> (E)	0..1	urn:poc-qoc:ccm: qoc-term: qoc-condition	urn:poc-qoc:ccm: qoc-term: qoc:LB	-	string	MatchId attribute can be set up according the application needs.
QoCCondition. qocUpperBound (P)	<AttributeValue> <AttributeDesig-nator> (E)	0..1	urn:poc-qoc:ccm: qoc-term: qoc-condition	urn:poc-qoc:ccm: qoc-term: qoc:UB	-	string	MatchId attribute can be set up according the application needs.

Table F.4: QoCTerm Guarantee to XACML Policy Target (“Body”)- (E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	**XACML Entity**	**Cardi-nality**	**Category**	**AttributeId**	**Function**	**DataType**	**Note**
trustCondition.id (P)	-	0..1	-	-	-	-	-
trustCondition. lowerBound (P)	<Rule RuleId= "urn:poc-qoc: ccm:trust:LB"> <ns:Condition>	0..1	-	-	double-greater-than-or-equal	double	-
trustCondition. upperBound (P)	<Rule RuleId= "urn:poc-qoc: ccm:trust:LB"> <ns:Condition>	0..1	-	-	double-less-than-or-equal	double	-
PredicateCondition.id (P)	RuleId	1..1	-	-	-	string	-
PredicateCondition. predicate (P)	<Condition> (E)	0.*	-	-	urn:poc-qoc: ccm:function: evaluate-predicate	-	boolean evaluate-predicate (predicateFunction(), expectedResult)

Table F.5: Trust Condition to XACML Rule (“Body”)- (E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	XACML Entity	Cardinality	Category	AttributeId	DataType
ConsumerContextContract (C)	<Request> (E)	-	-	-	-
ContextContract.id (P)	<Attribute> (E)	1..1	urn:poc-qoc:ccm: context-contract	urn:poc-qoc:ccm: context-contract:id	string
ContextContract.version (P)	<Attribute> (E)	0..1	urn:poc-qoc:ccm: context-contract	urn:poc-qoc:ccm: context-contract: version	string
ContextContract.description (P)	<Attribute> (E)	0..1	urn:poc-qoc:ccm: context-contract	urn:poc-qoc:ccm: context-contract: description	string
Agreement. clauseCombiningAlgId (P)	-	1..1	-	-	-
ContextContract. context_manager.id (P)	<Attribute> (E)	1..1	urn:poc-qoc:ccm:party	urn:poc-qoc:ccm: context-contract: context_manager: party:id	string
ConsumerContextContract. consumer.id (P)	<Attribute> (E)	0..1	urn:poc-qoc:ccm: party	urn:poc-qoc:ccm: context-contract: consumer:party:id	string
Agreement.Observable.name (P)	<Attribute> (E)	1..1	urn:poc-qoc:ccm: agreement	urn:poc-qoc:ccm: agreement: has-observable: observable:name	string

Table F.6: Consumer MuContext Contract to XACML Requext (“Header”)(E) Element - (A) Attribute - (C)Class - (P)Property

Context Contract Entity	XACML Entity	Cardinality	Category	AttributeId	DataType
Purpose.id (P)	<Attribute> (E)	0..*	urn:poc-qoc:ccm: privacy-term: purpose	urn:poc-qoc:ccm: privacy-term: purpose:id	string
Visibility .id (P)	<Attribute> (E)	0..*	urn:poc-qoc:ccm: privacy-term: visibility	urn:poc-qoc:ccm: privacy-term: visibility:id	string
Retention.id (P)	<Attribute> (E)	0..*	urn:poc-qoc:ccm: privacy-term: retention	urn:poc-qoc:ccm: privacy-term: retention:id	string
QoCCondition.id (P)	<Attributes (E) @id (A)>	1..1	urn:poc-qoc:ccm: observable	urn:oasis:names:tc: xacml:3.0:content-selector	string
QoCIM.Indicator.name (P) , QoCCondition. qocLowerBound (P), QoCCondition. qocUpperBound (P)	<Attribute> (E) <Content> (E) <AttributeValue @XPathCategory (A) > (E)	1..*	XPathCategory	"/contextReport/ observations/ observation/ qocim/indicator/has [@name="Indicator.name" and @value>=qocLowerBound and @value<=qocUpperBound]"	urn:oasis: names:tc: xacml:3.0: data- type: xpathExpre ssion
Clause (C)	<MultiRequests> (E) <RequestReference> (E)	1..*	-	-	-
QoCTerm	<AttributesReference> (E)	0..*	-	-	-
PrivacyTerm	<AttributesReference> (E)	0..*	-	-	-

Table F.7: ConsumerClauses to XACML Attributes (“Body”)(E) Element - (A) Attribute - (C)Class - (P)Property

Appendix G

MuContext Contract Translation into XACML 3.0 Examples

This illustration is split into several sections. The first one (Section G.1) shows the translation of a consumer contract into a XACML multi-request. It shows how the consumer contract is used to select a suitable producer offering the required privacy guarantees. The second and third sections (Sections G.2 and G.3) illustrate how a producer contract is translated into an XACML policy using a contract example with one clause. Step by step, we show how requirements, guarantees and conditions are represented in XACML.

Each illustrating section is structured as described below.

- **Description**: A short description, explaining the intention and highlighting the important items the reader should observe.
- **Context Contract Example**: XMI representation of the muContext Contract with an explanation if required.
- **Equivalent Context Contract in XACML**: XACML representation of the muContext Contract with its transformation explanation.

G.1 Consumer Request Example

G.1.1 Description

A context consumer wants to access an observable for different intentions defined by their respective purposes, visibility levels and retention guarantees. We illustrate below a consumer with two intentions.

- First intention: The consumer wants to read the owner's location with a medium ((e.g., an accuracy about 200 meters horizontally) or high level (e.g., an accuracy about 5 meters horizontally) of QoC, guaranteeing that, his purpose is to help users in a hazardous situation (e.g., heart-attack),

the visibility is restricted to users with a high trust level (e.g., EMS system of the city), the retention of the location will be at most 24 hours.

- Second intention: The consumer wants to read the owner's location with a high QoC level, guaranteeing that, the purpose is to do exercise with other users (e.g., ride bike together) , the visibility is restricted to the owner's family and friends, the retention of the location will be at most one hour.

With this example, we want to demonstrate that a consumer contract with several clauses (i.e., several intention) can be translated into several XACML requests (i.e., translated into an XACML multi-request) and discuss the drawbacks found in this translation approach.

G.1.2 MuContext Contract Example

A consumer contract is represented in XMI in accordance to the meta-model presented in Section 5.3. We extract from the contract three fragments: Listing G.1 corresponds to the header of the contract. Listing G.2 is the privacy guarantee corresponding to the first intention. And Listing G.3 is the QoC requirement for the first intention. The complete contract can be found in Appendix H Listing H.1.

Listing G.1: Consumer muContext Contract (Header)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <poccm:ConsumerRoot
3    xmi:version="2.0"
4    xmlns:xmi="http://www.omg.org/XMI"
5    xmlns:poccm="http://inf.telecom-sudparis.eu/income.poccm">
6    <consumer_contract
7      id="urn:poc-qoc:ccm:context-contract:example:consumer:contract:1"
8      description="This is a consumer muContext contract Example"
9      version="1.0">
10     <consumer
11       id="urn:consumer:id:1"
12       partyType="consumer"/>
13     <context_manager
14       id="urn:context-manager:id:1"
15       partyType="manager"/>
16     <has_consumer__agreement
17         clauseCombiningAlgId="deny_unless_permit">
18       <has_Observable
19         name="urn:observable:name:location"
20         uri=""/>
21 ...
```

In Listing G.1 the most important elements are:

- Line [7] the consumer contract identifier,
- Line [11] the context consumer identifier,
- Line [14] the context manager identifier,
- Line [19] the observable name.

Listing G.2: Consumer muContext Contract (Privacy Guarantee)

```
20 ...
21        <has_consumer_clause
22             id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
23             authorization="permit">
24           <privacy_guarantee>
25             <has_purpose
26              id="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"
27              description="Send help if the user is in danger.">
28              <has_action
29                id="urn:poc-qoc:ccm:privacy-term:purpose:action:heard_attack"
30                description="to find someone when is having a heard attack."/>
31             </has_purpose>
32
33             <has_visibility
34              id="urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person">
35              <has_circle
36                id="urn:poc-qoc:ccm:privacy-term:visibility:id:1">
37                <belongs xsi:type="visibilityMetaModel:Organization"
38                  id="urn:poc-qoc:ccm:privacy-term:visibility:id:EMS_City"/>
39              </has_circle>
40             </has_visibility>
41
42             <has_retention
43              id="urn:poc-qoc:ccm:privacy-term:retention:id:a-day">
44              <has_temporalRetention
45                id="urn:poc-qoc:ccm:privacy-term:retention:id:1"
46                timeFunction="on_day"
47                startTime="00h00"
48                duration="24"/>
49             </has_retention>
50           </privacy_guarantee>
51
52 ...
```

In Listing G.2 the most important elements are:

- Line [22] the clause identifier,
- Line [26] the visibility identifier,
- Line [36] the retention identifier,
- Line [43] the purpose identifier.

Listing G.3: Consumer muContext Contract (QoC Requirement)

```
53 ...
54           <qoc_requirement>
55             <has_qoc
56               id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"
57               qocUpperBound="//@consumer_contract/@has_consumer_agreement/@has_consumer_clause.0/
58               @qoc_requirement/@has_qoc/@dependsOn.0/@has.1"
59               qocLowerBound="//@consumer_contract/@has_consumer_agreement/@has_consumer_clause.0
60               /@qoc_requirement/@has_qoc/@dependsOn.0/@has.0">
61               <dependsOn
62                 name="urn:poc-qoc:ccm:qoc:indicator:name:precision"
```

```
63                id="1"
64                isQualifiedBy="//@consumer_contract/@has_consumer_agreement/@has_Observable">
65                <has
66                  id="1"
67                  name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:lb"
68                  value="80"/>
69                <has
70                  id="2"
71                  name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:ub"
72                  value="100"/>
73              </dependsOn>
74            </has_qoc>
75          </qoc_requirement>
76        </has_consumer_clause>
77  ...
```

In Listing G.3 the most important elements are:

- Line [56] the QoC identifier,
- Line [57] the reference to the qocUpperBound value,
- Line [59] the reference to the qocLowerBound value,
- Line [62] the QoC indicator name,
- Line [68] the lower bound value,
- Line [72] the upper bound value.

The qocUpperBound and qocLowerBound values reference to the Indicator's lower and upper bounds in Lines [65] and [69], see Figure 5.10 to observe a graphical example of these relationships.

G.1.3 Translation of the MuContext Contract into XACML

In this section, we show some fragments in order to explain how each consumer contract element is translated into an XACML request. The result of the translation of the consumer contract header is presented in Listing G.4, the parties in Listing G.5, the Agreement in Listing G.6, Privacy Guarantees in Listing G.7 and QoC requirements in G.8. The complete XACML request can be found in Appendix H Listing H.2.

In XACML, a single request MAY represent a request for multiple access control decisions. The syntax and semantics of such requests and responses are specified in the XACML specification [Parducci and Lockhart, 2010]. A XACML Multi-Request is illustrated in Listing G.9.

We use the <Attributes> element to specify new categories to describe the different elements of the muContext contract. For instance, the category "urn:poc-qoc:ccm:context-contract" lists a sequence of <Attribute> elements associated with the consumer contract identifier, version, and description (c.f., Listing G.4).

Listing G.4: XACML request (Consumer muContext Contract Header)

```
1  <?xml version="1.0" encoding="UTF-8"?>
```

```
<ns:Request
   xmlns:ns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
   C:\BALANA\xacml-core-v3-schema-wd-17.xsd"
   ReturnPolicyIdList="true" CombinedDecision="true">
   <!--muContext Contract Header-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:poc-qoc:ccm:context-contract:example:consumer:contract:1"
      Category="urn:poc-qoc:ccm:context-contract" >
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:id"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:context-contract:example:consumer:contract:1
         </ns:AttributeValue>
      </ns:Attribute>
               <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:version"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >1.0
         </ns:AttributeValue>
      </ns:Attribute>
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:description"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >This is a consumer muContext contract Example
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
...
```

Listing G.5 shows the the context consumer identifier and the context manager identifier translation in XACML request context. Note that, although they belong to the same category, they are declared in different <Attributes> elements. This is explained later when the multi request is performed.

Listing G.5: XACML request (Consumer muContext Contract Header - Parties)

```
...
   <!--Consumer-->
   <ns:Attributes
        xmlns:AA="http://www.w3.org/XML/1998/namespace"
        AA:id="urn:consumer:id:1"
           Category="urn:poc-qoc:ccm:party" >
           <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:consumer:party:id"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:consumer:id:1
                  </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
```

```
52    <!--Context Manager-->
53    <ns:Attributes
54       xmlns:AA="http://www.w3.org/XML/1998/namespace"
55       AA:id="urn:context-manager:id:1"
56       Category="urn:poc-qoc:ccm:party" >
57       <ns:Attribute
58          AttributeId="urn:poc-qoc:ccm:context-contract:context_manager:party:id"
59          IncludeInResult="true">
60          <ns:AttributeValue
61             DataType="http://www.w3.org/2001/XMLSchema#string"
62             >urn:context-manager:id:1
63          </ns:AttributeValue>
64       </ns:Attribute>
65    </ns:Attributes>
66 ...
```

Listing G.6 shows the definition of the observable identifier in the XACML request.

Listing G.6: XACML request (Consumer muContext Contract Header - Agreement)

```
65 ...
66    <!--Agreement deny_unless_permit-->
67    <ns:Attributes
68       xmlns:AA="http://www.w3.org/XML/1998/namespace"
69       AA:id="urn:observable:name:location"
70       Category="urn:poc-qoc:ccm:agreement">
71       <!--Observable-->
72       <ns:Attribute
73          AttributeId="urn:poc-qoc:ccm:agreement:has-observable:observable:name"
74          IncludeInResult="true">
75          <ns:AttributeValue
76             DataType="http://www.w3.org/2001/XMLSchema#string"
77             >urn:observable:name:location
78          </ns:AttributeValue>
79       </ns:Attribute>
80    </ns:Attributes>
81 ...
```

In Listing G.7, can be observed two purposes declaration in different <Attributes> elements. This is due to the multi request definition that will be discussed later. Similarly, we did visibility and retention declarations (more details in Appendix H Listing H.2).

Listing G.7: XACML request (consumer muContext Contract Privacy Guarantees)

```
81 ...
82    <!--Purpose 1-->
83    <ns:Attributes
84       xmlns:AA="http://www.w3.org/XML/1998/namespace"
85       AA:id="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"
86       Category="urn:poc-qoc:ccm:privacy-term:purpose">
87       <ns:Attribute
88          AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
89          IncludeInResult="true" >
90          <ns:AttributeValue
91             DataType="http://www.w3.org/2001/XMLSchema#string"
92             >urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger
93             <ns:Description>
94                Send help if the user is in danger.
95             </ns:Description>
```

```
96          </ns:AttributeValue>
97       </ns:Attribute>
98       <ns:Attribute
99          AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:purpose-category"
100         IncludeInResult="true" >
101         <ns:AttributeValue
102            DataType="http://www.w3.org/2001/XMLSchema#string"
103            >urn:poc-qoc:ccm:privacy-term:purpose:purpose-category:single
104            <ns:Description>
105               information can only be used for one specific task and at one given time.
106            </ns:Description>
107         </ns:AttributeValue>
108      </ns:Attribute>
109   </ns:Attributes>
110   <!--Purpose 2-->
111   <ns:Attributes
112      xmlns:AA="http://www.w3.org/XML/1998/namespace"
113      AA:id="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"
114      Category="urn:poc-qoc:ccm:privacy-term:purpose">
115      <ns:Attribute
116         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
117         IncludeInResult="true">
118         <ns:AttributeValue
119            DataType="http://www.w3.org/2001/XMLSchema#string"
120            >urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice
121            <ns:Description>
122               Meet the user to do exercice.
123            </ns:Description>
124         </ns:AttributeValue>
125      </ns:Attribute>
126      <ns:Attribute
127         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:purpose-category"
128         IncludeInResult="true" >
129         <ns:AttributeValue
130            DataType="http://www.w3.org/2001/XMLSchema#string"
131            >urn:poc-qoc:ccm:privacy-term:purpose:purpose-category:single
132            <ns:Description>
133               information can only be used for one specific task and at one given time.
134            </ns:Description>
135         </ns:AttributeValue>
136      </ns:Attribute>
137   </ns:Attributes>
138 ...
```

Listing G.8 describes the QoC condition. The <Attributes> element contains the following elements and attributes:

- Line [227], **xml:id**: This attribute provides a unique identifier for this <Attributes> element. It corresponds to the muContext contract "QoCcondition.id".
- Line [228], **Category**:This attribute indicates which attribute category the contained attributes belong to. Our resource is a context report. The resource attributes are placed in the urn:poc-qoc:ccm:observable attribute category of the <Request> element.
- Line [229-249], **<Content>**: Specifies the observable attributes (QoC meta-data in the context report) in XML document format which can be referenced using <AttributeSelector> elements.

- Line [250-261], **<Attribute>**: The identifier of the context report instance for which access is requested, which is an XPath expression into the <Content> element that selects the data to be accessed. the XPath expression verifies that the context report uses the same QoC indicators and the QOC level value corresponds to the limits established between the QoCCondition.qocLowerBound and QoCCondition.qocUpperBound values.

Listing G.8: XACML request (Consumer muContext Contract QoC requirement)

```
...
   <!--QoC requirement-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"
      Category="urn:poc-qoc:ccm:observable">
      <ns:Content>
         <contextReport
            xmlns:md="urn:example:schemas:context-report"
            xsi:schemaLocation="urn:example:schemas:context-report
            http://www.contextapp.example.com/schemas/context-report.xsd">
            <observations>
               <observation>
                  <value>+48 37 31.54, +2 26 32.36</value>
                  <qocim>
                     <indicator>
                        <has
                           id="1"
                           name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
                           value="00"
                           creationDate="00:00" />
                     </indicator>
                  </qocim>
               </observation>
            </observations>
         </contextReport>
      </ns:Content>
      <ns:Attribute
         IncludeInResult="false"
         AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
         <ns:AttributeValue
            XPathCategory="urn:poc-qoc:ccm:observable"
            DataType=" urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
            >/contextReport/observations/observation/qocim/indicator/has
            [@name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
            and @value>=80
            and @value<=100]
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
...
```

"XACML Multi Requests" feature.

To represent each consumer muContext contract clause, we propose to use the "XACML Multi Requests" feature. As each consumerClause is formed by a set of conditions that forms the consumer's

guarantees and requirements, each consumerClause can be interpreted as an individual request context. To illustrate this, Listing G.9 shows the tree clause of our contract example. Such an XACML request SHALL be interpreted as individual request specified by references to <Attributes> elements representing each consumerClause. The context handler MUST construct a new <Request> element for each <RequestReference> that contains the definitions of privacy and QoC terms of the element contained in the <MultiRequests> element, and process the generated <Request> element.

Each <RequestReference> element contains one or more <AttributesReference> elements, each of which refers to the AA:id XML attribute of one of the <Attributes> elements in the enclosing original <Request> element.

The generated <Request> element MUST be identical to a <Request> element which contains the referenced <Attributes> elements. The result(s) of each such generated <Request> element MUST be included as one or more <Result> elements in the <Response> element corresponding to the original <Request> element. There may be multiple results for a single generated <Request> element when the generated <Request> element makes use of one or more of the other multiple decision request schemes in this profile. There MUST be exactly one <Response> element for the original <Request> element.

Listing G.9: Example request context

```
...
   <ns:MultiRequests>
      <ns:RequestReference>
         <ns:AttributesReference
            ReferenceId="urn:context-manager:id:1"/>
         <ns:AttributesReference
            ReferenceId="urn:observable:name:location"/>
         <ns:AttributesReference
            ReferenceId="urn:consumer:id:1"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:a-day"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"/>
      </ns:RequestReference>
      <ns:RequestReference>
         <ns:AttributesReference
            ReferenceId="urn:context-manager:id:1"/>
         <ns:AttributesReference
            ReferenceId="urn:observable:name:location"/>
         <ns:AttributesReference
            ReferenceId="urn:consumer:id:1"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:family"/>
         <ns:AttributesReference
            ReferenceId="friends"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
         <ns:AttributesReference
            ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"/>
      </ns:RequestReference>
      <ns:RequestReference>
         <ns:AttributesReference
```

```
338             ReferenceId="urn:context-manager:id:1"/>
339          <ns:AttributesReference
340             ReferenceId="urn:observable:name:location"/>
341          <ns:AttributesReference
342             ReferenceId="urn:consumer:id:1"/>
343          <ns:AttributesReference
344             ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
345          <ns:AttributesReference
346             ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:friends"/>
347          <ns:AttributesReference
348             ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
349          <ns:AttributesReference
350             ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"/>
351       </ns:RequestReference>
352    </ns:MultiRequests>
353 </ns:Request>
```

G.2 Producer Policy Set (Contract Header)

G.2.1 Description

A context producer does not want that anyone access its observables for any purpose, any visibility level and any retention guarantee. And for each purpose, the context producer is willing to guarantee specific QoC levels.

With this example, we want to demonstrate that a producer contract with several clauses (i.e., requirements and guarantees by circumstance) can be translated into an XACML policy set, based on how it is evaluated by the PDP. We start with the header of the producer contract. Then, we present one producer clause as illustration.

G.2.2 MuContext Contract Example

Listing G.10 corresponds to the XMI representation of a producer context contract. In section G.3, we show the complete definition of a clause. In the meanwhile, we explain line by line what are the elements of this contract header.

Listing G.10: Producer muContext Contract Example

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <poccm:ProducerRoot
3        xmi:version="2.0"
4        xmlns:xmi="http://www.omg.org/XMI"
5        xmlns:poccm="http://inf.telecom-sudparis.eu/income.poccm">
6        <producer_contract
7              id="urn:poc-qoc:ccm:context-contract:example:producer:contract:1"
8              description="This is an example of muContext Contract Header" version="0.1">
9              <producer
10                   id="urn:producer:id:1"/>
11             <context_manager
12                   id="urn:context-manager:id:1"/>
13             <has_producer_agreement
14                   clauseCombiningAlgId="permit_unless_deny">
15                   <has_Observable
```

```
16                        name="urn:observable:name:location"/>
17                  <has_producer_clause
18                        id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
19                        authorization="permit"/>
20                  <has_producer_clause
21                        id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:2"
22                        authorization="permit"/>
23                  <has_producer_clause
24                        id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:3"
25                        authorization="permit"/>
26                  <has_producer_clause
27                        id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:4"
28                        authorization="deny"/>
29            </has_producer_agreement>
30      </producer_contract>
31 </poccm:ProducerRoot>
```

Lines [1]-[5] are a standard XML document tag indicating which version of XML is being used and what the character encoding is. Standard XML namespace declarations are included.

Lines [6] - [30] correspond to the <*producer_contract*> element defining the type of contract. It encloses all the elements of the context contract.

Line [7] is the *id* attribute used for identifying this contract. It SHOULD be unique.

Line [8] is the *description* attribute providing a text description of the context contract and the *version* attribute specifies the version of this contract.

Lines [9]-[10] correspond to the <*producer*> element declaration. It is optional and can be used to give access only to the context owner .

Lines [11]-[12] are the <*context_manager*> element declaration. It is required and used to set up the software entity that controls and provides the knowledge about the trust, context environment, visibility, purpose, retention and QoC.

Lines [13]-[29] are the <*has_producer_agreement*> element. It defines the clause combination algorithm. It encloses a set of <*clause*> elements, which belongs to the producer contract.

Line [14] is the *clauseCombiningAlgId* attribute specifying the algorithm that will be used to resolve the results of the various clauses that may be in the producer contract. The deny_unless_permit clause-combining algorithm specified here as example means that if any rule evaluates to “Deny”, then the policy must return “Deny”. If one rule evaluates to “Permit”, then the policy must return “Permit”.

Lines [15]-[16] correspond to the <*has_Observable*> element. It contains attributes of the observable which the contract will be requested to access.

Lines [17]-[19],[20]-[22],[23]-[25], and [26]-[28] are the <*has_producer_clause*> elements which are the header of each clause declaration. Each one defines its *id* attribute, which is unique by clause. And the *authorization* attribute that determines the decision effect if the clause evaluation returns “True”.

Line [29] closes the agreement.

Line [30] closes the contract.

Line [31] closes the XMI file.

G.2.3 Equivalent MuContext Contract in XACML

Listing G.11 corresponds to the XACML representation of the muContext producer contract presented in Listing G.10. The next paragraph explains line by line how the translation is performed.

Listing G.11: XACML Policy Set (Producer muContext Contract)

```
<?xml version="1.0" encoding="utf-8"?>
<ns:PolicySet
   xmlns:ns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
   C:\BALANA\xacml-core-v3-schema-wd-17.xsd"
   PolicySetId="urn:poc-qoc:ccm:context-contract:example:producer:contract:1"
   Version="1"
   PolicyCombiningAlgId="urn:clause-combining-algorithm:permit_unless_deny">
   <ns:Description>This is an example of muContext Contract Header</ns:Description>
   <ns:Target>
      <ns:AnyOf>
         <ns:AllOf>
            <ns:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <ns:AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  >urn:context-manager:id:1
                              </ns:AttributeValue>
               <ns:AttributeDesignator
                  Category="urn:poc-qoc:ccm:party"
                  AttributeId="urn:poc-qoc:ccm:context-contract:context_manager:party:id"
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  MustBePresent="true"/>
                        </ns:Match>
            <ns:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <ns:AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  >urn:observable:name:location
               </ns:AttributeValue>
               <ns:AttributeDesignator
                  Category="urn:poc-qoc:ccm:agreement"
                  AttributeId="urn:poc-qoc:ccm:agreement:has-observable:observable:name"
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  MustBePresent="true"/>
            </ns:Match>
                  </ns:AllOf>
      </ns:AnyOf>
      <ns:AnyOf>
         <ns:AllOf>
            <ns:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <ns:AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  >urn:producer:id:1
               </ns:AttributeValue>
               <ns:AttributeDesignator
                  Category="urn:poc-qoc:ccm:party"
                  AttributeId="urn:poc-qoc:ccm:context-contract:producer:party:id"
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  MustBePresent="false"/>
            </ns:Match>
         </ns:AllOf>
         <ns:AllOf>
            <ns:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```
54                <ns:AttributeValue
55                   DataType="http://www.w3.org/2001/XMLSchema#string"
56                   >urn:consumer:id:1
57                </ns:AttributeValue>
58                <ns:AttributeDesignator
59                   Category="urn:poc-qoc:ccm:party"
60                   AttributeId="urn:poc-qoc:ccm:context-contract:consumer:party:id"
61                   DataType="http://www.w3.org/2001/XMLSchema#string"
62                   MustBePresent="false"/>
63             </ns:Match>
64          </ns:AllOf>
65       </ns:AnyOf>
66       </ns:Target>
67    <ns:PolicyIdReference Version="0.1">
68       urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1
69       </ns:PolicyIdReference>
70    <ns:PolicyIdReference Version="0.1">
71       urn:poc-qoc:ccm:context-contract:example:contract:1:clause:2
72       </ns:PolicyIdReference>
73    <ns:PolicyIdReference Version="0.1">
74       urn:poc-qoc:ccm:context-contract:example:contract:1:clause:3
75       </ns:PolicyIdReference>
76       <ns:PolicyIdReference Version="0.1">
77       urn:poc-qoc:ccm:context-contract:example:contract:1:clause:4
78       </ns:PolicyIdReference>
79 </ns:PolicySet>
```

Lines [2]-[6] The <PolicySet> element is a top-level element in the XACML policy schema. <PolicySet> is an aggregation of other policies. As we mention it, these policies represent the producer clauses of the muContext contract.

Lines [3]-[6] are the XML namespace declarations.

Line [7], The "PolicySetId" attribute corresponds to the "ContextContract.id" property. It is used for identifying this policy set for possible inclusion in another policy set.

Line [8], The "Version" attribute contains the version number of the policy set. It is Optional. It is alike to the "ContextContract.version" property.

Line [10], The <Description> element contains a free-form description of the <PolicySet>. It has the same as the "ContextContract.description" property.

Line[9], like the "Agreement.clauseCombiningAlgId" property, the policies included in a <PolicySet> element MUST be combined using the algorithm identified by the "PolicyCombiningAlgId" attribute.

Lines [11]-[66] is the <Target> element and identifies the set of decision requests that the parent element is intended to evaluate. Based on that, the context contract's properties: ContextContract.context_manager.id, Agreement.Observable.name, ProducerContextContract.prducer.id, and ConsumerContextContract.consumer.id are required to identify the right XACML request to evaluate.

MuContext Contract Target Evaluation

First we explain how the target is evaluated and then how we use it to evaluate the muContext contracts.

According to the specification of XACML 3.0 [XACMLv3, 2013], the <TARGET> element identifies the set of decision requests that the parent element is intended to evaluate. And is structured as

follows:

- The <Target> element SHALL contain a conjunctive sequence (a sequence of predicates combined using the logical "AND" operation) of <AnyOf> elements.
- The <AnyOf> element SHALL contain a disjunctive sequence (a sequence of predicates combined using the logical "OR" operation) of <AllOf> elements.
- The <AllOf> element SHALL contain a conjunctive sequence of <Match> elements.
- The <Match> element SHALL identify a set of entities by matching attribute values in an <Attributes> element of the request context with the embedded attribute value.
- The <Match> element contains the following attributes and elements:
  - MatchId attribute [Required] specifies a matching function.
  - <AttributeValue> element [Required] embeds an attribute value.
  - <AttributeDesignator> element [Required] MAY be used to identify one or more attribute values in an <Attributes> element of the request context.

The target value SHALL be "Match" if all the "AnyOf" elements specified in the target match values in the request context. Otherwise, if any one of the "AnyOf" elements s specified in the target is "No Match", then the target SHALL be "No Match". Otherwise, the target SHALL be "Indeterminate".

To determine which producer contract (Policy Set) to apply, we define a target with two <AnyOf> elements. The first one, lines [12]-[37] is requested and encloses one <AllOf> element, which encloses two <Match> elements. One to verify if the decision request corresponds to the right observable (lines [28]-[35]) and other to check if the consumer party is using the same context manager (lines[14]-[24]).

We can represent this evaluation with the folowing predicate.

$$(target.context\_manager.id = request.context\_manager.id) \\ \wedge (target.observable.name = request.observable.name)$$

The second <AnyOf> element, lines [38]-[65], is optional and encloses two <AllOf> elements, which enclose one <Match> element each. The aim is verify the identity of the consumer or producer in case it is necessary.

$$[[target.producer.id = request.producer.id] \\ \vee [target.consumer.id = request.consumer.id]]$$

Lines [67]-[78] are references to the policies that contain the context contract clauses. The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element.

G.3 Producer Policy (Clause Example)

G.3.1 Description

The clause represented in Listing G.12 states, in English:

"If the owner is in danger, then any user with a high trust level is allowed to read the owner's location with the aim of helping him/her. If the previous condition is respected, the producer is willing to deliver the owner's location with a high level of QoC (i.e., between 80% and 100% of precision) (e.g., an accuracy less than 5 meters horizontally)."

With this example, we want to confirm that a muContext producer clause can be translated into an XACML policy, based on the way the PDP evaluates the XACML targets.

G.3.2 MuContext Contract Example

Listing G.12: Producer muContext Contract Clause Example

```
...
<has_producer_clause
  id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
  authorization="permit">
  <has_condition>
    <trustCondition
      levelId="urn:poc-qoc:ccm:condition:trust-condition:trust:id:1"
      lowerBound="0.5"
      upperBound="1.0"/>
    <predicateCondition
      id="urn:poc-qoc:ccm:condition:trust-condition:predicate:id:1"
      predicate="EmergencySituation = true"/>
  </has_condition>
  <privacy_requirement>
    <has_purpose
      id="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"
      description="Send help if the user is in danger."/>
  </privacy_requirement>
  <qoc_guarantee>
    <has_qoc
       id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:1"
      qocUpperBound="//@producer_contract/@has_producer_agreement/@has_producer_clause.0
      /@qoc_guarantee/@has_qoc/@dependsOn.0/@has.1"
      qocLowerBound="//@producer_contract/@has_producer_agreement/@has_producer_clause.0
      /@qoc_guarantee/@has_qoc/@dependsOn.0/@has.0">
      <dependsOn
        name="urn:qocim:indicator:id:precision-indicator"
        isQuqlifiedBy="//@producer_contract/@has_producer_agreement/@has_Observable">
        <has
          id="1"
          name="urnqocim:metric-value:id:precision-value:LB"
          value="80"
          definition="//@producer_contract/@has_producer_agreement/@has_producer_clause.0
          /@qoc_guarantee/@has_qoc/@dependsOn.0/@isDefinedBy/@contains.0"/>
        <has
          id="2"
          name="urnqocim:metric-value:id:precision-value:UB"
          value="100"
          definition="//@producer_contract/@has_producer_agreement/@has_producer_clause.0
```

```
55          /@qoc_guarantee/@has_qoc/@dependsOn.0/@isDefinedBy/@contains.0"/>
56        <isDefinedBy
57          name="urn:poc-qoc:ccm:qoc:criterion:name:precision ">
58          <contains
59            name="urnqocim:metric-definition:name:per-cent-precision"
60            value="//@producer_contract/@has_producer_agreement/@has_producer_clause.0
61            /@qoc_guarantee/@has_qoc/@dependsOn.0/@has.1 //@producer_contract
62            /@has_producer_agreement/@has_producer_clause.0/@qoc_guarantee/@has_qoc/
63            @dependsOn.0/@has.0"/>
64        </isDefinedBy>
65      </dependsOn>
66    </has_qoc>
67  </qoc_guarantee>
68 </has_producer_clause>
69 ...
```

Line [18] is the id attribute used for identifying this clause. It should be unique. Line [19] is the authorization attribute that determines the decision effect if the clause evaluation returns "True". This clause has two conditions (Lines [20]-[28]): Firstly in Lines [21]-[24], the <trustCondition> allows to check that the trust of an end user who wants to access the information and checks lies between 0.5 and 1.0. Secondly in Lines [25]-[27], the <predicateCondition> context Owner must declare that he/she is in emergency situation with "EmergencySituation = true".

If both conditions are evaluated to true, then the privacy requirements are applied (Lines [29]-[33]). In this case, there is only one "purpose" requirement. The purpose declared by the end-user must be the same identified in the clause as "urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger" which means, "Send help if the user is in danger.". If all three conditions are satisfied, then the producer is committed to deliver his/her location with a high level of QoC (Lines [34]-[67]) (i.e., deliver with a precision value between 80% and 100%).

G.3.3 Equivalent MuContext Contract in XACML

"An XACML policy consists of header information, an optional text description of the policy, a target, one or more rules and an optional set of obligation expressions." [XACMLv3, 2013]. In this section, to explain how the clause elements are translated into an XACML Policy, we split the policy in several parts:

Listing G.13 describes the policy header, Listing G.14 defines the privacy terms, Listing G.15 defines the function to get trust level of the involved users, Listing G.16 establishes the bounds for the trust condition, and Listing G.17 defines the predicate condition, Listing G.18 defines the QoC terms. The full contract can be found in Appendix H Listing H.3.

In Listing G.13 Line [7] is the PolicyId attribute assigns a name to this policy instance. It corresponds to the muContext contract clause identifier. The name of a policy has to be unique for a given PDP so that there is no ambiguity if one policy is referenced from another policy. In the current muContext contract model all clause conditions and terms must evaluate "true" to apply the clause authorization effect. At Line [9] RuleCombiningAlgId is fixed to the rule combining algorithm "deny-unless-permit", meaning that, all rules must evaluate true to give the access to the observable. Each rule represents a clause condition.

Listing G.13: XACML Policy (Producer Clause Header)

```
<?xml version="1.0" encoding="utf-8"?>
<ns:Policy
  xmlns:ns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
  C:\BALANA\xacml-core-v3-schema-wd-17.xsd"
  PolicyId="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
  Version="0.1"
  RuleCombiningAlgId =
  "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-unless-permit">
  <ns:Description>
    If the owner is in danger. Any user with a high trust
    level is allowed to read the owner's location
    with the aim of help him/her.
  </ns:Description>
...
```

Listing G.14 shows the target of the policy, the target contain the privacy requirements definition that should match the request context privacy guarantees.

The policy <Target> element describes the decision requests to which this policy applies. If the attributes in a decision request do not match the values specified in the policy target, then the remainder of the policy does not need to be evaluated.

The <Match> element specifies a matching function in the MatchId attribute, a literal value of "urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger" and a pointer to a specific purpose identifier attribute "urn:poc-qoc:ccm:privacy-term:purpose:id" in the request context by means of the <AttributeDesignator> element with an attribute category which specifies the purpose "urn:poc-qoc:ccm:privacy-term:purpose". The matching function will be used to compare the literal value with the value of the purpose identifier attribute "urn:poc-qoc:ccm:privacy-term:purpose:id". Only if the match returns "True" will this rule apply to a particular decision request. If the match returns "False", then this rule will return a value of "NotApplicable".

Listing G.14: XACML Policy (Producer Clause - Privacy Terms)

```
...
  <!--Privacy Requirements Definition-->
  <ns:Target>
    <ns:AnyOf>
      <ns:AllOf>
        <!--Purpose Validation-->
        <ns:Match
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger
            <ns:Description>
              Send help if the user is in danger.
            </ns:Description>
          </ns:AttributeValue>
          <ns:AttributeDesignator
            Category="urn:poc-qoc:ccm:privacy-term:purpose"
            AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true" />
```

```
35          </ns:Match>
36        </ns:AllOf>
37      </ns:AnyOf>
38    </ns:Target>
39  ...
```

Listing G.15 illustrates the use of the <VariableDefinition> element to define a function to get the end-user trust level, that is be used throughout the rules that define the trust condition.

This is a non-standar function. It SHALL take one argument, that SHALL be of data-type "http://www.w3.org/2001/XMLSchema♯string", with the consumer identity. It SHALL return a result of "http://www.w3.org/2001/XMLSchema♯double". This function SHALL return the trust level value by queering the context manager.

Listing G.15: XACML Policy (Producer Clause - Trust Condition)

```
38  ...
39    <!-- New function to search the consumer trust level evaluation -->
40    <ns:VariableDefinition VariableId="get-user-trust-level">
41      <ns:Apply
42        FunctionId=
43          "urn:poc-qoc:ccm:function:double-get-user-trust-level">
44        <ns:Apply
45          FunctionId=
46            "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
47          <ns:AttributeDesignator
48            Category="urn:poc-qoc:ccm:party"
49            AttributeId="urn:poc-qoc:ccm:context-contract:consumer:party:id"
50            DataType="http://www.w3.org/2001/XMLSchema#string"
51            MustBePresent="true"/>
52        </ns:Apply>
53      </ns:Apply>
54    </ns:VariableDefinition>
55  ...
```

In Listing G.16 show two rules that evaluate if the consumer trust level correspond to the boundaries established in the contract. Both condition must evaluate to "True" for the rule to be applicable. This condition contains a reference to a variable definition defined elsewhere in the policy.

When Trust condition is translate from the producer muContext contract the effect attribute and RuleId attributes are fixed as follow. The RuleId attributes are set up to "urn:poc-qoc:ccm:trust:LB" and "urn:poc-qoc:ccm:trust:LB" respectively. And, the effect in both rules is "Permit". Finally, the target section says the rule is applicable to any decision request.

Listing G.16: XACML Policy (Producer Clause - Trust Condition Bounds)

```
54  ...
55    <!-- Trust Lower Bound Validation-->
56    <ns:Rule
57      RuleId="urn:poc-qoc:ccm:trust:LB"
58      Effect="Permit">
59      <ns:Target />
60      <ns:Condition>
61        <ns:Apply
62          FunctionId=
63            "urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal">
64          <ns:AttributeValue
```

```
            DataType="http://www.w3.org/2001/XMLSchema#double"
            >0.5
          </ns:AttributeValue>
          <ns:VariableReference VariableId="get-user-trust-level" />
        </ns:Apply>
      </ns:Condition>
    </ns:Rule>
    <!-- Trust Upper Bound Validation-->
    <ns:Rule
      RuleId="urn:poc-qoc:ccm:trust:UB"
      Effect="Permit">
      <ns:Target />
      <ns:Condition>
        <ns:Apply
          FunctionId=
          "urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal">
          <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#double"
            >1.0
          </ns:AttributeValue>
          <ns:VariableReference VariableId="get-user-trust-level" />
        </ns:Apply>
      </ns:Condition>
    </ns:Rule>
...
```

The final rule in this policy evaluate the condition if the owner is in emergency-situation. the RuleId attribute corresponds to the predicate condition identifier. The Effect attribute is fixed to “Permit”. And the <Condition> element apply a non-standar function “urn:poc-qoc:ccm:function:evaluate-predicate”, that evaluates the predicate.

This function applies a Boolean function between the predicate and the expected result. The expression SHALL be “True” if and only if the supplied predicate and the expected result are equals. This function SHALL take two arguments. The first argument SHALL be an <Function> element that names a Boolean function that takes one or more arguments, among them the owner’s identifier. The second argument SHALL be a Boolean data-type, indicating the expected result of the function give as argument.

Listing G.17: XACML Policy (Producer Clause - Predicate Condition)

```
...
  <!--Is the context owner is on emergency situation?-->
  <ns:Rule
    RuleId="urn:poc-qoc:ccm:condition:predicate:id:emergency-situation"
    Effect="Permit">
    <ns:Condition>
      <ns:Apply
        FunctionId="urn:poc-qoc:ccm:function:evaluate-predicate">
        <ns:Apply
           FunctionId=
            "urn:poc-qoc:ccm:condition:predicate:function:emergency-situation">
           <ns:AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string"
              >urn:producer:id:1
           </ns:AttributeValue>
        </ns:Apply>
        <ns:AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#boolean"
```

```
106             >true
107           </ns:AttributeValue>
108         </ns:Apply>
109       </ns:Condition>
110     </ns:Rule>
111 ...
```

Listing G.18 shows the translation of the QoC guarante into XACML policy obligation. The Obligations are a set of operations that must be performed by the PEP in conjunction with an authorization decision [XACMLv3, 2013]. In this example, the obligation must be fulfilled when the policy authorization decision evaluates "Permit" (see Line [116], the FulfillOn attribute).

Lines [114] - [151] The <ObligationExpression> element consists of the ObligationId attribute, which identifies the obligation. In this case, the Context Manager is required to send or produce context reports (context data container) with a high level of QoC (i.e., between 80% and 100% of precision). Lines [117] - [126] The first parameter indicates to the Context manager to filter context reports with the QoC level indicated. The PDP will evaluate the <AttributeSelector> and return the result to the context manager inside the resulting obligation. Lines [128] - [134] The second parameter contains literal text for the metric value name used to evaluate the QoC. Finally, Lines [136] - [150] The third and fourth parameters indicate the lower and upper bounds values that the context manager will use to filter or obfuscate the context reports content.

Listing G.18: XACML Policy (Producer Clause - QoC Terms)

```
111 ...
112 <!--QoC Guarantee Definition-->
113   <ObligationExpressions>
114     <ObligationExpression
115     ObligationId="urn:poc-qoc:ccm:qoc-term:qoc-condition:obligation1"
116     FulfillOn="Permit">
117     <AttributeAssignmentExpression
118       <AttributeSelector
119       MustBePresent="true"
120       Category="urn:poc-qoc:ccm:observable"
121       Path="/contextReport/observations/observation/qocim/indicator/has
122       @name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
123       and @value>=80
124       and @value<=100"
125       DataType="http://www.w3.org/2001/XMLSchema#string"/>
126     </AttributeAssignmentExpression>
127
128     <AttributeAssignmentExpression
129       AttributeId="urn:poc-qoc:ccm:qoc:metric-value:name">
130       <AttributeValue
131       DataType="http://www.w3.org/2001/XMLSchema#string"
132       >urn:poc-qoc:ccm:qoc:metric-value:name:precision
133       </AttributeValue>
134     </AttributeAssignmentExpression>
135
136     <AttributeAssignmentExpression
137       AttributeId="urn:poc-qoc:ccm:qoc-term:qoc:LB">
138       <AttributeValue
139       DataType="http://www.w3.org/2001/XMLSchema#string"
140       >80
141       </AttributeValue>
142     </AttributeAssignmentExpression>
```

```
143
144     <AttributeAssignmentExpression
145       AttributeId="urn:poc-qoc:ccm:qoc-term:qoc:UB">
146       <AttributeValue
147       DataType="http://www.w3.org/2001/XMLSchema#string"
148       >100
149       </AttributeValue>
150     </AttributeAssignmentExpression>
151     </ObligationExpression>
152   </ObligationExpressions>
153 </ns:Policy>
```

Appendix H

muContext Contract Full Listings

Listing H.1: Example Consumer muContext Contract

```
<?xml version="1.0" encoding="UTF-8"?>
<poccm:ConsumerRoot
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:poccm="http://inf.telecom-sudparis.eu/income.poccm">
  <consumer_contract
    id="urn:poc-qoc:ccm:context-contract:example:consumer:contract:1"
    description="This is a consumer muContext contract Example"
    version="1.0">
    <consumer
      id="urn:consumer:id:1"
      partyType="consumer"/>
    <context_manager
      id="urn:context-manager:id:1"
      partyType="manager"/>
    <has_consumer__agreement
        clauseCombiningAlgId="deny_unless_permit">
      <has_Observable
        name="urn:observable:name:location"
        uri=""/>

      <has_consumer_clause
          id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
          authorization="permit">

        <privacy_guarantee>
          <has_purpose
            id="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"
            description="Send help if the user is in danger.">
            <has_action
              id="urn:poc-qoc:ccm:privacy-term:purpose:action:heard_attack"
              description="to find someone when is having a heard attack."/>
          </has_purpose>

          <has_visibility
            id="urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person">
            <has_circle
              id="urn:poc-qoc:ccm:privacy-term:visibility:id:1">
              <belongs xsi:type="visibilityMetaModel:Organization"
```

```
40                id="urn:poc-qoc:ccm:privacy-term:visibility:id:EMS_City"/>
41              </has_circle>
42            </has_visibility>
43
44            <has_retention
45              id="urn:poc-qoc:ccm:privacy-term:retention:id:a-day">
46              <has_temporalRetention
47                id="urn:poc-qoc:ccm:privacy-term:retention:id:1"
48                timeFunction="on_day"
49                startTime="00h00"
50                duration="24"/>
51            </has_retention>
52          </privacy_guarantee>
53
54          <qoc_requirement>
55            <has_qoc
56              id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"
57              qocUpperBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.0/
58              @qoc_requirement/@has_qoc/@dependsOn.0/@has.1"
59              qocLowerBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.0
60              /@qoc_requirement/@has_qoc/@dependsOn.0/@has.0">
61              <dependsOn
62                name="urn:poc-qoc:ccm:qoc:indicator:name:precision"
63                id="1"
64                isQuqlifiedBy="//@consumer_contract/@has_consumer__agreement/@has_Observable">
65                <has
66                  id="1"
67                  name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:lb"
68                  value="80"/>
69                <has
70                  id="2"
71                  name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:ub"
72                  value="100"/>
73              </dependsOn>
74            </has_qoc>
75          </qoc_requirement>
76        </has_consumer_clause>
77        <has_consumer_clause
78          id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:2"
79          authorization="permit">
80          <privacy_guarantee>
81            <has_visibility
82              id="urn:poc-qoc:ccm:privacy-term:visibility:id:family"/>
83            <has_retention
84              id="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
85            <has_purpose
86              id="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
87          </privacy_guarantee>
88          <qoc_requirement>
89            <has_qoc
90              id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"
91              qocUpperBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.1/
92              @qoc_requirement/@has_qoc/@dependsOn.0/@has.1"
93              qocLowerBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.1/
94              @qoc_requirement/@has_qoc/@dependsOn.0/@has.0">
95              <dependsOn
96                id="1"
97                name="urn:poc-qoc:ccm:qoc:indicator:name:precision"
98                isQuqlifiedBy="//@consumer_contract/@has_consumer__agreement/@has_Observable">
99                <has
```

```
                    id="1"
                    name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:lb"
                    value="40"/>
                  <has
                    id="2"
                    name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:ub"
                    value="100"/>
                </dependsOn>
              </has_qoc>
            </qoc_requirement>
          </has_consumer_clause>
          <has_consumer_clause
            id="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:2"
            authorization="permit">
            <privacy_guarantee>
              <has_visibility
                id="urn:poc-qoc:ccm:privacy-term:visibility:id:friends"/>
              <has_retention
                id="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
              <has_purpose
                id="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
            </privacy_guarantee>
            <qoc_requirement>
              <has_qoc
                id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"
                qocUpperBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.2/
                @qoc_requirement/@has_qoc/@dependsOn.0/@has.1"
                qocLowerBound="//@consumer_contract/@has_consumer__agreement/@has_consumer_clause.2/
                @qoc_requirement/@has_qoc/@dependsOn.0/@has.0">
                <dependsOn
                  name="urn:poc-qoc:ccm:qoc:indicator:name:precision"
                  id="1"
                  isQuqlifiedBy="//@consumer_contract/@has_consumer__agreement/@has_Observable">
                  <has
                    id="1"
                    name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:lb"
                    value="40"/>
                  <has
                    id="2"
                    name="urn:poc-qoc:ccm:qoc:metric-value:id:precision:ub"
                    value="100"/>
                </dependsOn>
              </has_qoc>
            </qoc_requirement>
          </has_consumer_clause>
        </has_consumer__agreement>
      </consumer_contract>
</poccm:ConsumerRoot>
```

Listing H.2: Example request context

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:Request
   xmlns:ns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
   C:\BALANA\xacml-core-v3-schema-wd-17.xsd"
   ReturnPolicyIdList="true" CombinedDecision="true">
   <!--muContext Contract Header-->
```

```
      <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:poc-qoc:ccm:context-contract:example:consumer:contract:1"
      Category="urn:poc-qoc:ccm:context-contract" >
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:id"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:context-contract:example:consumer:contract:1
         </ns:AttributeValue>
      </ns:Attribute>
            <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:version"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >1.0
         </ns:AttributeValue>
      </ns:Attribute>
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:description"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >This is a consumer muContext contract Example
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
   <!--Consumer-->
   <ns:Attributes
        xmlns:AA="http://www.w3.org/XML/1998/namespace"
        AA:id="urn:consumer:id:1"
           Category="urn:poc-qoc:ccm:party" >
           <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:consumer:party:id"
         IncludeInResult="false">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:consumer:id:1
                  </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
   <!--Context Manager-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:context-manager:id:1"
      Category="urn:poc-qoc:ccm:party" >
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:context-contract:context_manager:party:id"
         IncludeInResult="true">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:context-manager:id:1
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
   <!--Agreement permit_unless_deny-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
```

```
      AA:id="urn:observable:name:location"
      Category="urn:poc-qoc:ccm:agreement">
      <!--Observable-->
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:agreement:has-observable:observable:name"
         IncludeInResult="true">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:observable:name:location
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
   <!--Privacy Guarantees-->
   <!--Purpose 1-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"
      Category="urn:poc-qoc:ccm:privacy-term:purpose">
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
         IncludeInResult="true" >
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger
            <ns:Description>
               Send help if the user is in danger.
            </ns:Description>
         </ns:AttributeValue>
      </ns:Attribute>
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:purpose-category"
         IncludeInResult="true" >
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:privacy-term:purpose:purpose-category:single
            <ns:Description>
               information can only be used for one specific task and at one given time.
            </ns:Description>
         </ns:AttributeValue>
      </ns:Attribute>
   </ns:Attributes>
   <!--Purpose 2-->
   <ns:Attributes
      xmlns:AA="http://www.w3.org/XML/1998/namespace"
      AA:id="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"
      Category="urn:poc-qoc:ccm:privacy-term:purpose">
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
         IncludeInResult="true">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice
            <ns:Description>
               Meet the user to do exercice.
            </ns:Description>
         </ns:AttributeValue>
      </ns:Attribute>
      <ns:Attribute
         AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:purpose-category"
         IncludeInResult="true" >
```

```
129         <ns:AttributeValue
130            DataType="http://www.w3.org/2001/XMLSchema#string"
131            >urn:poc-qoc:ccm:privacy-term:purpose:purpose-category:single
132            <ns:Description>
133               information can only be used for one specific task and at one given time.
134            </ns:Description>
135         </ns:AttributeValue>
136      </ns:Attribute>
137   </ns:Attributes>
138   <!--Visibility 1-->
139   <ns:Attributes
140      xmlns:AA="http://www.w3.org/XML/1998/namespace"
141      AA:id="urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person"
142      Category="urn:poc-qoc:ccm:privacy-term:visibility">
143      <ns:Attribute
144         AttributeId="urn:poc-qoc:ccm:privacy-term:visibility:id"
145         IncludeInResult="true">
146         <ns:AttributeValue
147            DataType="http://www.w3.org/2001/XMLSchema#string"
148            >urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person
149            <ns:Description>
150               Any person register on the system
151               with high trust level.
152            </ns:Description>
153         </ns:AttributeValue>
154      </ns:Attribute>
155   </ns:Attributes>
156   <!--Visibility 2-->
157   <ns:Attributes
158      xmlns:AA="http://www.w3.org/XML/1998/namespace"
159      AA:id="urn:poc-qoc:ccm:privacy-term:visibility:id:family"
160      Category="urn:poc-qoc:ccm:privacy-term:visibility">
161      <ns:Attribute
162         AttributeId="urn:poc-qoc:ccm:privacy-term:visibility:id"
163         IncludeInResult="true">
164         <ns:AttributeValue
165            DataType="http://www.w3.org/2001/XMLSchema#string"
166            >urn:poc-qoc:ccm:privacy-term:visibility:id:family
167            <ns:Description>
168               User's family members register on the system.
169            </ns:Description>
170         </ns:AttributeValue>
171      </ns:Attribute>
172   </ns:Attributes>
173   <!--Visibility 3-->
174   <ns:Attributes
175      xmlns:AA="http://www.w3.org/XML/1998/namespace"
176      AA:id="urn:poc-qoc:ccm:privacy-term:visibility:id:friends"
177      Category="urn:poc-qoc:ccm:privacy-term:visibility">
178      <ns:Attribute
179         AttributeId="urn:poc-qoc:ccm:privacy-term:visibility:id"
180         IncludeInResult="true">
181         <ns:AttributeValue
182            DataType="http://www.w3.org/2001/XMLSchema#string"
183            >urn:poc-qoc:ccm:privacy-term:visibility:id:friends
184            <ns:Description>
185               User's friends register on the system.
186            </ns:Description>
187         </ns:AttributeValue>
188      </ns:Attribute>
```

```
</ns:Attributes>
<!--Retention 1-->
<ns:Attributes
   xmlns:AA="http://www.w3.org/XML/1998/namespace"
   AA:id="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"
   Category="urn:poc-qoc:ccm:privacy-term:retention">
   <ns:Attribute
      AttributeId="urn:poc-qoc:ccm:privacy-term:retention:id"
      IncludeInResult="true">
      <ns:AttributeValue
         DataType="http://www.w3.org/2001/XMLSchema#string"
         >urn:poc-qoc:ccm:privacy-term:retention:id:an-hour
         <ns:Description>
            Store the information only for an hour
         </ns:Description>
      </ns:AttributeValue>
   </ns:Attribute>
</ns:Attributes>
<!--Retention 2-->
<ns:Attributes
   xmlns:AA="http://www.w3.org/XML/1998/namespace"
   AA:id="urn:poc-qoc:ccm:privacy-term:retention:id:a-day"
   Category="urn:poc-qoc:ccm:privacy-term:retention">
   <ns:Attribute
      AttributeId="urn:poc-qoc:ccm:privacy-term:retention:id"
      IncludeInResult="true">
      <ns:AttributeValue
         DataType="http://www.w3.org/2001/XMLSchema#string"
         >urn:poc-qoc:ccm:privacy-term:retention:id:a-day
         <ns:Description>
            Store the information only for 24 hours
         </ns:Description>
      </ns:AttributeValue>
   </ns:Attribute>
</ns:Attributes>
<!--QoC requirement-->
<ns:Attributes
   xmlns:AA="http://www.w3.org/XML/1998/namespace"
   AA:id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"
   Category="urn:poc-qoc:ccm:observable">
   <ns:Content>
      <contextReport
         xmlns:md="urn:example:schemas:context-report"
         xsi:schemaLocation="urn:example:schemas:context-report
         http://www.contextapp.example.com/schemas/context-report.xsd">
         <observations>
            <observation>
               <value>+48 37 31.54, +2 26 32.36</value>
               <qocim>
                  <indicator>
                     <has
                        id="1"
                        name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
                        value="00"
                        creationDate="00:00" />
                  </indicator>
               </qocim>
            </observation>
         </observations>
      </contextReport>
```

```
249         </ns:Content>
250         <ns:Attribute
251            IncludeInResult="false"
252            AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
253            <ns:AttributeValue
254               XPathCategory="urn:poc-qoc:ccm:observable"
255               DataType=" urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">
256               /contextReport/observations/observation/qocim/indicator/has
257               [@name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
258               and @value>=80
259               and @value<=100]
260            </ns:AttributeValue>
261         </ns:Attribute>
262      </ns:Attributes>
263      <ns:Attributes
264         xmlns:AA="http://www.w3.org/XML/1998/namespace"
265         AA:id="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"
266         Category="urn:poc-qoc:ccm:observable">
267         <ns:Content>
268            <contextReport
269               xmlns:md="urn:example:schemas:context-report"
270               xsi:schemaLocation="urn:example:schemas:context-report
271               http://www.contextapp.example.com/schemas/context-report.xsd">
272               <observations>
273                  <observation>
274                     <value>+48 37 31.54, +2 26 32.36</value>
275                     <qocim>
276                        <indicator>
277                           <has
278                              id="1"
279                              name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
280                              value="00"
281                              creationDate="00:00" />
282                        </indicator>
283                     </qocim>
284                  </observation>
285               </observations>
286            </contextReport>
287         </ns:Content>
288         <ns:Attribute
289            IncludeInResult="false"
290            AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
291            <ns:AttributeValue
292               XPathCategory="urn:poc-qoc:ccm:observable"
293               DataType=" urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">
294               /contextReport/observations/observation/qocim/indicator/has
295               [@name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
296               and @value>=40
297               and @value<=100]
298            </ns:AttributeValue>
299         </ns:Attribute>
300      </ns:Attributes>
301      <ns:MultiRequests>
302         <ns:RequestReference>
303            <ns:AttributesReference
304               ReferenceId="urn:context-manager:id:1"/>
305            <ns:AttributesReference
306               ReferenceId="urn:observable:name:location"/>
307            <ns:AttributesReference
308               ReferenceId="urn:consumer:id:1"/>
```

```
309            <ns:AttributesReference
310               ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger"/>
311            <ns:AttributesReference
312               ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:reliable-person"/>
313            <ns:AttributesReference
314               ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:a-day"/>
315            <ns:AttributesReference
316               ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:1"/>
317         </ns:RequestReference>
318         <ns:RequestReference>
319            <ns:AttributesReference
320               ReferenceId="urn:context-manager:id:1"/>
321            <ns:AttributesReference
322               ReferenceId="urn:observable:name:location"/>
323            <ns:AttributesReference
324               ReferenceId="urn:consumer:id:1"/>
325            <ns:AttributesReference
326               ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
327            <ns:AttributesReference
328               ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:family"/>
329            <ns:AttributesReference
330               ReferenceId="friends"/>
331            <ns:AttributesReference
332               ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
333            <ns:AttributesReference
334               ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"/>
335         </ns:RequestReference>
336         <ns:RequestReference>
337            <ns:AttributesReference
338               ReferenceId="urn:context-manager:id:1"/>
339            <ns:AttributesReference
340               ReferenceId="urn:observable:name:location"/>
341            <ns:AttributesReference
342               ReferenceId="urn:consumer:id:1"/>
343            <ns:AttributesReference
344               ReferenceId="urn:poc-qoc:ccm:privacy-term:purpose:id:do-execice"/>
345            <ns:AttributesReference
346               ReferenceId="urn:poc-qoc:ccm:privacy-term:visibility:id:friends"/>
347            <ns:AttributesReference
348               ReferenceId="urn:poc-qoc:ccm:privacy-term:retention:id:an-hour"/>
349            <ns:AttributesReference
350               ReferenceId="urn:poc-qoc:ccm:qoc-term:qoc-condition:id:precision-value:2"/>
351         </ns:RequestReference>
352      </ns:MultiRequests>
353   </ns:Request>
```

Listing H.3: Producer Policy Clause 1 Example

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ns:Policy
3    xmlns:ns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
4    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5    xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
6    C:\BALANA\xacml-core-v3-schema-wd-17.xsd"
7    PolicyId="urn:poc-qoc:ccm:context-contract:example:contract:1:clause:1"
8    Version="0.1"
9    RuleCombiningAlgId =
10   "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-unless-permit">
11   <ns:Description>
```

```
    If the owner is in danger. Any user with a high trust
    level is allowed to read the owner's location
    with the aim of help him/her.
  </ns:Description>
  <!--Privacy Requirements Definition-->
  <ns:Target>
    <ns:AnyOf>
      <ns:AllOf>
        <!--Purpose Validation-->
        <ns:Match
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:poc-qoc:ccm:privacy-term:purpose:id:help-in-danger
            <ns:Description>
              Send help if the user is in danger.
            </ns:Description>
          </ns:AttributeValue>
          <ns:AttributeDesignator
            Category="urn:poc-qoc:ccm:privacy-term:purpose"
            AttributeId="urn:poc-qoc:ccm:privacy-term:purpose:id"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true" />
        </ns:Match>
      </ns:AllOf>
    </ns:AnyOf>
  </ns:Target>
  <!-- New function to search the consumer trust level evaluation -->
  <ns:VariableDefinition VariableId="get-user-trust-level">
    <ns:Apply
      FunctionId=
        "urn:poc-qoc:ccm:function:double-get-user-trust-level">
      <ns:Apply
        FunctionId=
          "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <ns:AttributeDesignator
          Category="urn:poc-qoc:ccm:party"
          AttributeId="urn:poc-qoc:ccm:context-contract:consumer:party:id"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          MustBePresent="true"/>
      </ns:Apply>
    </ns:Apply>
  </ns:VariableDefinition>
  <!-- Trust Lower Bound Validation-->
  <ns:Rule
    RuleId="urn:poc-qoc:ccm:trust:LB"
    Effect="Permit">
    <ns:Target />
    <ns:Condition>
      <ns:Apply
        FunctionId=
          "urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal">
        <ns:AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#double"
          >0.5
        </ns:AttributeValue>
        <ns:VariableReference VariableId="get-user-trust-level" />
      </ns:Apply>
    </ns:Condition>
  </ns:Rule>
```

```
<!-- Trust Upper Bound Validation-->
<ns:Rule
  RuleId="urn:poc-qoc:ccm:trust:UB"
  Effect="Permit">
  <ns:Target />
  <ns:Condition>
    <ns:Apply
      FunctionId=
      "urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal">
      <ns:AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#double"
        >1.0
      </ns:AttributeValue>
      <ns:VariableReference VariableId="get-user-trust-level" />
    </ns:Apply>
  </ns:Condition>
</ns:Rule>
<!--Is the context owner is on emergency situation?-->
<ns:Rule
  RuleId="urn:poc-qoc:ccm:condition:predicate:id:emergency-situation"
  Effect="Permit">
  <ns:Condition>
    <ns:Apply
      FunctionId="urn:poc-qoc:ccm:function:evaluate-predicate">
      <ns:Apply
         FunctionId=
          "urn:poc-qoc:ccm:condition:predicate:function:emergency-situation">
         <ns:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:producer:id:1
         </ns:AttributeValue>
      </ns:Apply>
      <ns:AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >true
      </ns:AttributeValue>
    </ns:Apply>
  </ns:Condition>
</ns:Rule>
<ObligationExpressions>
 <ObligationExpression
 ObligationId="urn:poc-qoc:ccm:qoc-term:qoc-condition:obligation1"
 FulfillOn="Permit">
 <AttributeAssignmentExpression
   <AttributeSelector
   MustBePresent="true"
   Category="urn:poc-qoc:ccm:observable"
   Path="/contextReport/observations/observation/qocim/indicator/has
   @name="urn:poc-qoc:ccm:qoc:metric-value:name:precision"
   and @value>=80
   and @value<=100"
   DataType="http://www.w3.org/2001/XMLSchema#string"/>
 </AttributeAssignmentExpression>

 <AttributeAssignmentExpression
   AttributeId="urn:poc-qoc:ccm:qoc:metric-value:name">
   <AttributeValue
   DataType="http://www.w3.org/2001/XMLSchema#string"
   >urn:poc-qoc:ccm:qoc:metric-value:name:precision
   </AttributeValue>
```

```
132     </AttributeAssignmentExpression>
133
134     <AttributeAssignmentExpression
135       AttributeId="urn:poc-qoc:ccm:qoc-term:qoc:LB">
136       <AttributeValue
137       DataType="http://www.w3.org/2001/XMLSchema#string"
138       >80
139       </AttributeValue>
140     </AttributeAssignmentExpression>
141
142     <AttributeAssignmentExpression
143       AttributeId="urn:poc-qoc:ccm:qoc-term:qoc:UB">
144       <AttributeValue
145       DataType="http://www.w3.org/2001/XMLSchema#string"
146       >100
147       </AttributeValue>
148     </AttributeAssignmentExpression>
149     </ObligationExpression>
150   </ObligationExpressions>
151 </ns:Policy>
```

Bibliography

[Abid and Chabridon, 2011] Abid, Z. and Chabridon, S. (2011). A fine-grain approach for evaluating the quality of context. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 444 –449. 56

[Agrawal et al., 2002] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. (2002). Hippocratic databases. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 143–154. VLDB Endowment. 25, 26, 27, 28, 29, 56

[Agre and Rotenberg, 1998] Agre, P. and Rotenberg, M. (1998). *Technology and privacy: The new landscape*. The MIT Press. 1, 20

[Anciaux et al., 2006] Anciaux, N., van Heerde, H., Feng, L., and Apers, P. M. (2006). Implanting life-cycle privacy policies in a context database. 99

[Arcangeli et al., 2012] Arcangeli, J.-P., Boujbel, R., Bouzeghoub, A., Camps, V., Chabridon, S., Conan, D., Desprats, T., Guivarch, V., Laborde, R., Lavinal, E., Leriche, S., Oglaza, A., Péninou, A., Rottenberg, S., Taconet, C., and Zaraté, P. (2012). Multi-scale context management: Concepts, Definitions, and State of the art. ANR INFRA INCOME Multi-Scale Context Management for the Internet of Things deliverable Task 2.1. 13, 68

[Atkinson and Kühne, 2003] Atkinson, C. and Kühne, T. (2003). Model-driven development: A metamodeling foundation. *IEEE Softw.*, 20(5):36–41. 31

[Atzori et al., 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805. v, x, xxi, 11, 12

[Bagüés et al., 2007] Bagüés, A. S., Zeidler, A., Fernandez Valdivielso, C., and Matias, I. (2007). Towards personal privacy control. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, volume 4806 of *Lecture Notes in Computer Science*, pages 886–895. Springer Berlin Heidelberg. 46

[Bagüés et al., 2010] Bagüés, S. A., Zeidler, A., Klein, C., Valdivielso, C. F., and Matias, I. R. (2010). Enabling personal privacy for pervasive computing environments. *Journal of Universal Computer Science*, 16(3):341–371. xxi, 45, 46, 47, 57

[Barker et al., 2009] Barker, K., Askari, M., Banerjee, M., Ghazinour, K., Mackas, B., Majedi, M., Pun, S., and Williams, A. (2009). A data privacy taxonomy. In *Proceedings of the 26th British National Conference on Databases: Dataspace: The Final Frontier*, BNCOD 26, pages 42–54, Berlin, Heidelberg. Springer-Verlag. xxi, 22, 23, 24, 37, 56

[Bass et al., 2003] Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice, 2nd Edition*. Addison-Wesley. 63, 66

[Bisdikian et al., 2012] Bisdikian, C., Sensoy, M., Norman, T. J., and Srivastava, M. B. (2012). Trust and Obfuscation Principles for Quality of Information in Emerging Pervasive Environments. In *IEEE PerCom Workshop Proc., Lugano, Switzerland*, pages 44–49. 29, 83

[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 HPOVUA*, Geneva. 2, 13

[Buckley, 2006] Buckley, J. (2006). From RFID to the Internet of Things: Pervasive Networked Systems. In *Final Report Conf. DG InfSo, Brussels*. European Commission. 1

[Chabridon et al., 2013] Chabridon, S., DESPRATS, T., LABORDE, R., MARQUEZ, S. M., MARIE, P., OGLAZA, A., TACONET, C., and ZARATE, P. (2013). Models of quality and protection of multiscalable context information, livrable 4.1, income project. xxii, 81, 164

[Chakraborty et al., 2011] Chakraborty, S., Choi, H., and Srivastava, M. (2011). Demystifying privacy in sensory data: A QoI based approach. In *PERCOM IQ2S Workshop*. xxi, 51, 52, 54

[Chakraborty et al., 2012] Chakraborty, S., Raghavan, K., Srivastava, M., Bisdikian, C., and Kaplan, L. (2012). An Obfuscation Framework for Controlling Value of Information during Sharing. In *IEEE Statistical Signal Processing Workshop*. xxi, 51, 52, 53, 57, 99

[Cheaito et al., 2011] Cheaito, M., Laborde, R., Barrérre, F., and Benzekri, A. (2011). A deployment framework for self-contained policies (regular paper). In *IEEE/IFIP International Conference on Network and Service Management (CNSM), Niagara Falls - CANADA, 25/10/2010-29/10/2010*, pages 88–95, http://www.comsoc.org. IEEE Communications Society. 27

[Choi et al., 2011] Choi, H., Chakraborty, S., Charbiwala, Z., and Srivastava, M. (2011). Sensorsafe: A framework for privacy-preserving management of personal sensory information. In Jonker, W. and Petkovic, M., editors, *Secure Data Management*, volume 6933 of *Lecture Notes in Computer Science*, pages 85–100. Springer. 51, 54

[Clarke, 2006] Clarke, R. (2006). Introduction to dataveillance and information privacy, and definitions of terms. 26

[Clarke, 2013] Clarke, R. (2013). Introduction to dataveillance and information privacy, and definitions and terms. xxi, 19, 20, 27

[Clearinghouse, 2013] Clearinghouse, P. R. (2013). Privacy today: A review of current issues. 21

[Clements et al., 2011] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2011). *Documenting Software Architecture: Views and Beyond, 2nd Edition*. Addison-Wesley. 69

[Conan et al., 2013] Conan, D., Li, X., Lim, L., and Zhu, J. (2013). Architecture of multiscalable context management, livrable 3.1.a.1, income project. xxii, 69, 72, 73

[Coutaz et al., 2005] Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005). Context is key. *Commun. ACM*, 48(3):49–53. v, x

[Cuppens and Miège, 2003] Cuppens, F. and Miège, A. (2003). Modelling Contexts in the Or-BAC Model. In *19th Annual Computer Security Applications Conference (ACSAC '03)*. 96, 97, 98

[Czarnecki and Helsen, 2003] Czarnecki, K. and Helsen, S. (2003). Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, volume 45, pages 1–17. USA. 147

[Dey et al., 2001] Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16:97–166. 20, 56

[Duta and Barker, 2008] Duta, A. C. and Barker, K. (2008). P4a: A new privacy model for xml. In *DBSec*, pages 65–80. 56

[Eclipse Foundation, 2010] Eclipse Foundation (2010). Eclipse Modeling Framework (EMF). http://www.eclipse.org/modeling/emf/. 102

[EU, 1995] EU (1995). EU Directive 95/46/ec on the protection of individuals with regard to the processing of personal data and the free movement of such data. *Official Journal of the European Communities*. vi, 4, 21

[EU, 2002] EU (2002). EU Directive 2002/58/ec on the processing of personal data and the protection of privacy in the electronic communications sector. *Official Journal of the European Communities*. vi, 4, 21

[Filho, 2010] Filho, J. B. (2010). *A Family of Context-Based Access Control Models for Pervasive Environments*. PhD thesis, University of Grenoble Joseph Fourier, France, Univ. of Grenoble, France. v, x, xxi, 13, 48, 49, 56, 57

[Foundation, 2014] Foundation, E. (2014). "atl" a model transformation technology. 145

[Gambetta, 2000] Gambetta, D. (2000). Can We Trust Trust? *Trust: Making and Breaking Cooperative Relations, electronic edition, Department of Sociology, University of Oxford*, pages 213–237. 17

[Ghazinour et al., 2009a] Ghazinour, K., Majedi, M., and Barker, K. (2009a). A lattice-based privacy aware access control model. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 03*, CSE '09, pages 154–159, Washington, DC, USA. IEEE Computer Society. 22, 23, 24, 56

[Ghazinour et al., 2009b] Ghazinour, K., Majedi, M., and Barker, K. (2009b). A model for privacy policy visualization. In *Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference*, volume 02 of *COMPSAC '09*, pages 335–340, Washington, DC, USA. IEEE Computer Society. 56

[Giusto et al., 2010] Giusto, D., Iera, A., Morabito, G., and Atzori, L. (2010). *The Internet of Things*. Springer-Verlag. v, x, 10

[Grandison and Sloman, 2000] Grandison, T. and Sloman, M. (2000). A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16. 17

[Greenleaf, 2012] Greenleaf, G. (2012). The influence of european data privacy standards outside europe: Implications for globalisation of convention 108. *International Data Privacy Law*, 2(2):68–92. 21, 22

[Henricksen et al., 2002] Henricksen, K., Indulska, J., and Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, pages 167–180, London, UK, UK. Springer-Verlag. 13

[Hönle et al., 2005] Hönle, N., Käppeler, U.-W., Nicklas, D., Schwarz, T., and Großmann, M. (2005). Benefits of Integrating Meta Data into a Context Model. In *3rd IEEE Conference on Pervasive Computing and Communications Workshops*, pages 25–29, Kauai Island, HI, USA. IEEE Computer Society. 13

[IBM, 2003] IBM (2003). Enterprise privacy authorization language (epal). 33, 37

[INCOME, 2013a] INCOME (2013a). Multi-scale context management:concepts, definitions,and state of the art, livrable 2.1. 17, 26, 27, 29, 99

[INCOME, 2013b] INCOME (2013b). Scénarios et exigences, livrable 2.2. 101, 164

[Indulska et al., 2003] Indulska, J., Robinson, R., Rakotonirainy, A., and Henricksen, K. (2003). Experiences in using cc/pp in context-aware systems. In *In Proc. of the Intl. Conf. on Mobile Data Management (MDM*, pages 247–261. Springer. 13

[INFSO, 2008] INFSO, D. (2008). Networked enterprise & RFID infso g. 2 micro & nanosystems, in co-operation with the working group RFID of the etp eposs, internet of things in 2020, roadmap for the future [r]. *Information Society and Media, Tech. Rep*. 10

[Katasonov et al., 2008] Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., and Terziyan, V. (2008). Smart semantic middleware for the internet of things. In *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics*. 12

[Kosta and Dumortier, 2008] Kosta, E. and Dumortier, J. (2008). Taxonomy. picos deliverable d2.1. 32

[Kumaraguru et al., 2007] Kumaraguru, P., Lobo, J., Cranor, L. F., and Calo, S. B. (2007). S.: A survey of privacy policy languages. In *In: Workshop on Usable IT Security Management (USM 07): Proceedings of the 3rd Symposium on Usable Privacy and Security, ACM*. 36

[Ma, 2011] Ma, H. (2011). Internet of things: Objectives and scientific challenges. *Journal of Computing Science and Technology*, 26(6). 1

[Marie et al., 2013a] Marie, P., Desprats, T., Chabridon, S., and Sibilla, M. (2013a). QoCIM : A Meta-model for Quality of Context. In *CONTEXT, LNCS 8175*. 93, 99, 100, 108

[Marie et al., 2013b] Marie, P., Desprats, T., Chabridon, S., and Sibilla, M. (2013b). Qocim: A meta-model for quality of context. In Brezillon, P., Blackburn, P., and Dapoigny, R., editors, *CONTEXT*, volume 8175 of *Lecture Notes in Computer Science*, pages 302–315. Springer. xxi, 13, 14, 15, 16, 37, 157

[Mbanaso et al., 2009] Mbanaso, U., Cooper, G., Chadwick, D., and Anderson, A. (2009). Obligations of trust for privacy and confidentiality in distributed transactions. *Internet Research*, 19(2):153–173. xxi, 40, 41, 42, 57

[Mbanaso et al., 2007] Mbanaso, U. M., Cooper, G., Chadwick, D., and Anderson, A. (2007). Obligations for privacy and confidentiality in distributed transactions. In *Emerging Directions in Embedded and Ubiquitous Computing*, pages 69–81. Springer. 40, 42

[McKnight and Chervany, 1996] McKnight, D. H. and Chervany, N. L. (1996). The Meanings of Trust. Working Paper. 17

[Meyer, 1992] Meyer, B. (1992). Applying "design by contract". *Computer*, 25(10):40–51. 29

[Neisse, 2012] Neisse, R. (2012). *Trust and Privacy Management Support for Context-Aware Service Platforms*. PhD thesis, CTIT - University of Twente, Univ. of Twente. xxi, 18, 37, 54, 55, 56, 57, 157

[Neisse et al., 2008] Neisse, R., Wegdam, M., and van Sinderen, M. (2008). Trustworthiness and Quality of Context Information. In *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS'2008*, pages 1925–1931, Hunan, China. 54

[OASIS, 2012] OASIS (2012). extensible access control markup language (xacml). 27

[Obeo, 2014a] Obeo (2014a). "acceleo" code generator. 145

[Obeo, 2014b] Obeo (2014b). "obeo designer". 102

[OECD, 1980] OECD (1980). Guidelines on the protection of privacy and transborder flows of personal data. http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_1_1_1_1,00.html. vi, 4, 21

[OMG, 2005] OMG (2005). *Unified Model Language (UML)*. 30

[OMG, 2008] OMG (2008). *Software & Systems Process Engineering Meta-model (SPEM) v2.0*. Object Management Group, formal/2008-04-01 edition. 144

[OMG, 2011a] OMG (2011a). *Meta Object Facility (MOF)*. last access may 2013. 30, 31

[OMG, 2011b] OMG (2011b). *XML Metadata Interchange (XMI)*. 30

[Parducci and Lockhart, 2010] Parducci, B. and Lockhart, H. (2010). *XACML v3.0 Multiple Decision Profile Version 1.0, OASIS Standard*. 198

[Press, 2012] Press, C. U. (2012). Cambridge Dictionaries Online. 29, 30

[PrimeLife, 2009] PrimeLife (2009). Design for Policy Languages and Protocols, 2d Draft, PRIMELife Project Deliverable. http://primelife.ercim.eu/results/documents/120-h532-draft-2nd-design-for-policy-lan 45

[PrimeLife, 2010] PrimeLife (2010). Policy Engine PrimeLife, 2d Release, PRIMELife Project Deliverable. http://primelife.ercim.eu/results/documents/121-532d. xxi, 42, 43, 44, 45, 57

[Ragouzis et al., 2008] Ragouzis, N., Hughes, J., Philpott, R., Maler, E., Madsen, P., and Scavo, T. (2008). Oasis security assertion markup language (saml) v2.0 technical overview. xxi, 34, 37

[Rapps and Weyuker, 1982] Rapps, S. and Weyuker, E. J. (1982). Data flow analysis techniques for test data selection. In *Proceedings of the 6th International Conference on Software Engineering*, ICSE '82, pages 272–278, Los Alamitos, CA, USA. IEEE Computer Society Press. 129

[Rosenberg and Remy, 2004] Rosenberg, J. and Remy, D. (2004). *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Pearson Higher Education. xxi, 36

[Sackmann et al., 2006] Sackmann, S., Str "uker, J., and Accorsi, R. (2006). Personalization in privacy-aware highly dynamic systems. *Communications of the ACM*, 49(9):38. 33

[Skinner, 2006] Skinner, Geoffrey andChang, E. (2006). A conceptual framework for information privacy and security in collaborative

environments. *IJCSNS International Journal of Computer Science and Network Security*, 6:166–172. 25, 26, 27, 28, 29

[Skinner et al., 2006a] Skinner, G., Han, S., and Chang, E. (2006a). A taxonomy for information privacy in virtual collaborations. *WSEAS TRANS on INFORMATION SCIENCE and APPLICATIONS*, 3(6):1108–1115. 20, 56

[Skinner et al., 2006b] Skinner, G., Han, S., and Chang, E. (2006b). An introduction to a taxonomy of information privacy in collaborative environments. In *Proceedings of the 5th WSEAS international conference on Applied computer science*, ACOS'06, pages 980–985, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS). 56, 96

[Standards, 2005] Standards, O. (2005). Security and privacy considerations for the oasis security assertion markup language (saml) v2.0. 35

[Taconet and Kazi-Aoul, 2010] Taconet, C. and Kazi-Aoul, Z. (2010). Building Context-Awareness Models for Mobile Applications. *JDIM*, 8(2):78–87. 86, 88

[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 *OTM '09*, Vilamoura, Portugal. Springer-Verlag. vii, 5

[Ullman, 1988] Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press. 25

[U.S. Legal Inc., 2013] U.S. Legal Inc. (2013). Elements of a Contract. http://contracts.uslegal.com/elements-of-a-contract/. 30

[Violino, 2003] Violino, B. (2003). Benetton to tag 15 million items. RFID Journal. 19, 20

[Völter, 2009] Völter, M. (2009). Best Practices for DSLs and Model-Driven Development. *Journal of Object Technology*, 8(6):79–102. 30

[W3C, 2011] W3C (2011). The platform for privacy preferences project (p3p). http://www.w3.org/P3P/. 27, 37

[Wishart et al., 2005] Wishart, R., Henricksen, K., and Indulska, J. (2005). Context Obfuscation for Privacy via Ontological Descriptions. In Springer-Verlag, editor, *Proceedings of the 1st Int. Workshop on Location and Context-Awareness (LoCA)*, volume 3479 of *lncs*. 99

[WSO2, 2014] WSO2 (2014). "balana" the open source xacml 3.0 implementation. 143, 152

[XACMLv3, 2013] XACMLv3 (2013). *eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard*. 35, 36, 37, 43, 88, 127, 146, 147, 150, 184, 207, 210, 214

[Zimmer et al., 2006] Zimmer, T. et al. (2006). Qoc: Quality of context-improving the performance of context-aware applications. *Advances in Pervasive Computing*, page 209–214. 13