Contrats de contexte pour la gestion de contexte répartie

Nadia Masmoudi et Denis Conan
Institut Mines-Télécom, Télécom SudParis, UMR CNRS SAMOVAR
9 rue Charles Fourier, 91011 Évry, France
prénom.nom@telecom-sudparis.eu

RÉSUMÉ

Les applications qui s'exécutent dans les environnements ambiants ont besoin d'informations de contexte pour s'adapter aux changements de l'environnement. Un défi majeur qui suscite l'intérêt des chercheurs est de permettre un échange de l'information de contexte. Ce travail s'intéresse particulièrement à la modélisation de la répartition de la gestion de contexte.

La répartition des informations de contexte doit être considérée dès le début de la modélisation. Un modèle de répartition des données de contexte permet l'expression des contraintes de contexte et l'acheminement des informations de contexte sans ambiguïté entre les fournisseurs ou les consommateurs d'informations de contexte, et le service de gestion de contexte, ainsi qu'entre les différents composants du service de gestion de contexte. Cela nous amène à définir un modèle de contrat de contexte. Pour assurer une communication asynchrone entre les fournisseurs et les consommateurs d'informations de contexte, nous nous appuyons sur le modèle d'interaction des Systèmes Répartis à Base d'Événements (SRBE).

ABSTRACT

In ambient computing environments, applications need to monitor their execution context and react to changes in their environment. The distribution of context data poses great challenges to researchers. In this paper, we focus in the modeling of the distribution of context data.

A model of the distribution of context data allows the expression of context constraints and the delivery of unambiguous context information between providers and consumers, and the context management service, and between the various components of the context management service. We argue that the managament of context data distribution can be achieved by establishing context contracts. For this purpose, we propose a model of context contract and use Distributed Events Based Systems (DEBS) as an appropriate interaction model for building asynchronous communication between providers and consumers of context information.

UbiMob 2013, 5–6 juin 2013, Nancy, France.

Categories and Subject Descriptors

Computer systems organization [**Architectures**]: Distributed architectures; Software and its engineering [**Software organization and properties**]: Software system structures—*Distributed systems organizing principles*

Keywords

Middleware, context-awareness, contract, distributed event-based systems, complex event processing

1. INTRODUCTION

La prolifération des appareils mobiles tels que les téléphones et les capteurs embarqués dans les objets que nous utilisons dans notre vie quotidienne, fait naître un nouvel environnement fortement dynamique appelé Internet des objets (en anglais, *Internet of Things*, IoT). Dans un tel environnement, le contexte est « toute information pouvant être utilisée pour caractériser la situation d'une entité (personne, objet physique ou objet virtuel), et plus généralement tout élément pouvant influencer le comportement d'une application » [9].

La notion de sensibilité au contexte concerne l'utilisation du contexte dans les applications. Elle caractérise la capacité d'un système à s'adapter aux changements du contexte. Selon Dey et Abowd, « un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, la pertinence dépendant de la tâche demandée par l'utilisateur » [9].

La complexité de développement et de reconfiguration des applications mobiles sensibles au contexte rend nécessaire le découplage entre la gestion de contexte et la logique de l'application. Selon [7], un gestionnaire de contexte est composé de quatre niveaux d'abstraction. Le niveau le plus bas de l'architecture s'occupe de la collecte des informations de contexte en provenance des capteurs. À ce niveau d'abstraction, les observations sont des grandeurs numériques. Le deuxième niveau s'appuie sur les observations numériques pour produire des observations symboliques. Le troisième niveau identifie les situations de changement pertinent du contexte. Enfin, le plus haut niveau de l'architecture représente l'exploitation des informations de contexte par les composants de l'application.

La problématique de la distribution des informations de

contexte a été résumée dans les synthèses [1, 2]. Dans ce papier, nous nous intéressons particulièrement à la modélisation de la répartition de la gestion de contexte. Un gestionnaire de contexte doit assurer l'échange d'informations entre les fournisseurs et les consommateurs d'informations de contexte. Dans la suite, nous désignons par clients de contexte les composants qui sont connectés au gestionnaire de contexte (fournisseurs ou consommateurs).

Pour assurer une communication sans ambiguïté entre les clients et le gestionnaire de contexte, nous proposons d'introduire des contrats de contexte exprimant des contraintes sur ce que les fournisseurs de contexte tels que les capteurs peuvent fournir au gestionnaire de contexte et ce que les consommateurs ont besoin de recevoir comme informations de contexte en provenance du gestionnaire de contexte. Nous introduisons le contrat de contexte dans la modélisation de contexte dans une approche basée sur l'Ingénierie Dirigée par les Modèles (IDM) avec un méta-modèle de contrat de contexte. Ensuite, nous mettons en œuvre le paradigme publier/souscrire utilisé dans les Systèmes Répartis à Base d'Évenements (SRBE) [12]. Les SRBE [21] assurent une communication asynchrone et permettent le découplage entre les fournisseurs et les consommateurs. Nous faisons une correspondance entre les contrats de contexte introduits dans le service de gestion de contexte et les filtres introduits dans les SRBE. Le contrat de contexte peut alors servir comme souscription du côté consommateur de contexte et comme annonce du côté fournisseur de contexte.

Le papier est organisé comme suit. Nous présentons nos motivations et nos objectifs dans la section 2. Nous parcourons l'état de l'art dans la section 3. Nous décrivons le contrat de contexte dans la section 4. Nous présentons l'architecture de gestion de contexte répartie à base de contrats de contexte dans la section 5. Enfin, nous concluons et fournissons des perspectives pour nos travaux futurs dans la section 6.

2. MOTIVATIONS ET OBJECTIFS

Pour illustrer nos motivations et nos objectifs, nous décrivons le scénario application de transport multimodal en ville dans la section 2.1. Dans les sections 2.2 et 2.3, nous présentons la gestion de contexte orientée processus puis les systèmes répartis à base d'événements et le modèle de traitement complexe d'événements (CEP). Ensuite, nous motivons l'utilisation de la conception par contrat et l'approche IDM dans la section 2.4.

2.1 Application de transport multimodal en ville

Le scénario « application de transport multimodal en ville » décrit dans cette section est inspiré du projet INCOME [17]. En hiver, une vague de froid s'abat en milieu de journée. Une grande quantité de neige tombe et des plaques de verglas se forment. Jérôme décide de rentrer chez lui. Il utilise l'application d'assistance au déplacement multimodal installée sur son téléphone mobile géolocalisé. Une fois connecté, Jérôme envoie les critères de recherche pour son itinéraire de retour en précisant ses préférences : adapter son trajet en fonction des prévisions météo et partager son trajet avec des amis afin de s'entraider. Il reçoit des informations sur les perturbations du réseau des transports en commun et des informations sur ses amis qui peuvent partager un parcours avec lui. L'application répond à sa demande et lui affiche une liste d'itinéraires disponibles en prenant en considération les diverses informations reçues des différents services. Jérôme choisit l'un de ces itinéraires. L'itinéraire est ensuite adapté en continu pendant le déplacement de Jérôme et selon l'évolution du contexte.

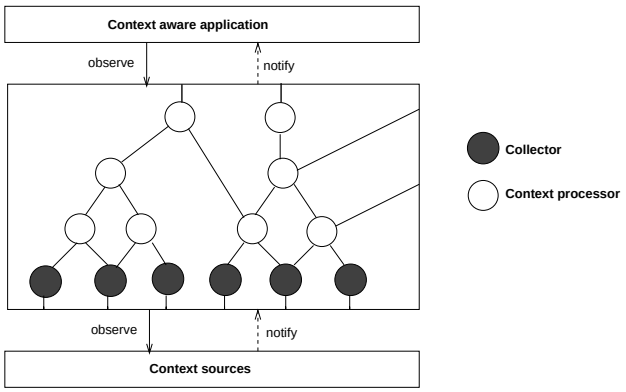

Figure 1: COSMOS PCM

L'application de transport multimodal en ville est une application sensible au contexte. Elle dépend des préférences utilisateur, des informations du trafic des transports en commun, des prévisions météo, de la disponibilité des places de parking et des vélos en libre service ainsi que des utilisateurs connectés à l'application.

2.2 Gestion de contexte orientée processus

Notre gestion de contexte est orientée processus (en anglais, *Process-oriented Context Management*, PCM) [3]. L'orientation processus intègre toutes les acquisitions et transformations d'informations de contexte dans des entités de traitement séparées qui forment un graphe de composants pour la gestion et la distribution des informations de contexte. Les nœuds feuilles du graphe représentent les collecteurs qui collectent des données brutes provenant des sources de contexte. Les autres nœuds du graphe représentent les processeurs de contexte qui filtrent, agrègent et interprètent des informations de contexte de plus haut niveau d'abstraction. Les nœuds racines du graphe fournissent les situations à l'application. Context Toolkit [10], The Contextor [8] et COSMOS [5] sont des exemples de gestionnaires de contexte suivant l'approche PCM. La figure 1 illustre l'approche PCM de COSMOS.

2.3 SRBE et CEP

Les SRBE utilisent le modèle d'interaction publier/souscrire assurant une communication asynchrone et le découplage dans l'espace et dans le temps des fournisseurs et consommateurs [21]. Les SRBE introduisent la notion d'annonce du côté fournisseur pour déclarer l'ensemble des notifications publiées et la notion de souscription du côté consommateur pour décrire l'ensemble des notifications à recevoir. Les annonces et les souscriptions sont des filtres sous forme de fonction à valeur booléenne qui teste la notification et retourne la valeur *true* ou *false*.

Dans un environement ambiant, les (dé)connexions des four-

nisseurs et des consommateurs d'informations de contexte peuvent se faire à tout moment. Les informations de contexte produites par les sources de contexte ne sont pas destinées à des consommateurs particuliers. Les interactions synchrones avec connaissance des pariticpants ne sont pas appropriées dans un tel environement. Nous utilisons les SRBE dans nos travaux pour la gestion et la répartition des informations de contexte. Dans la figure 1, les composants communiquent entre eux via un SRBE.

Le modèle de traitement complexe d'événements (en anglais, *Complex Event Processing*, CEP) est un des modèles de données et de traitement de flots de données dans les « Information Processing Flow » (IFP) [19]. Un processeur d'événements procède à différents types d'opérations sur un événement ou sur un ensemble d'événements : filtrage, agrégation, détection de patrons d'événements. L'adjectif « complexe » du terme CEP correspond à la nature de ces traitements. Dans [11], les auteurs présentent une classification de processeurs d'événements. Un CEP est construit au-dessus d'un SRBE. Dans la figure 1, le CEP correspond à l'architecture dessinée en considérant les collecteurs comme sources d'événements et les processeurs de contexte comme des processeurs d'événements.

2.4 Contrat de contexte et IDM

Notre solution PCM repose sur la définition de contrats de contexte. Nous concevons un contrat de contexte pour les raisons suivantes. Tout d'abord, un contrat de contexte permet d'exprimer sans ambiguïté, d'une part, les informations de contexte désirées par un consommateur de contexte, et d'autre part, les informations de contexte produites par un fournisseur de contexte. C'est ainsi l'élément partagé entre les clients de contexte et le service de gestion de contexte. Le contrat de contexte est une formalisation des contraintes de contexte permettant de restreindre les flots de données échangés. Le contrat de contexte sert en outre à l'auto-adaptation en permettant non seulement la mise à jour des tables de routage en cas de déconnexion ou reconnexion d'un client de contexte mais aussi la gestion du placement des composants de traitement complexe d'événements au niveau du gestionnaire de contexte. Cependant, l'auto-adaptation n'est pas étudiée dans ce papier, mais fait partie de futurs travaux.

La définition d'un modèle de contexte doit s'intégrer dans le processus de conception de l'application sensible au contexte. Ces dernières années, la modélisation des applications sensibles au contexte met l'accent sur l'ingénierie dirigée par les modèles (IDM). L'IDM considère les modèles comme les éléments principaux du développement et ajoute un niveau d'abstraction par rapport au modèle de contexte en définissant un méta-modèle de contexte. L'approche IDM assure la conformité du modèle de contexte avec le méta-modèle et offre des techniques de transformation de modèles et de génération de code. Nous présentons une approche de modélisation des contrats de contexte selon l'approche IDM.

3. ÉTAT DE L'ART

Notre étude porte sur deux sujets distincts : la gestion de contexte répartie et la conception par contrat. Un aperçu des travaux de recherche sur les gestionnaires de contexte est présenté dans la section 3.1 et un aperçu des approches de conception par contrat est présenté dans la section 3.2.

3.1 Gestion de contexte répartie

Dans [2], les auteurs énoncent des exigences pour la distribution des informations de contexte : découplage des fournisseurs et consommateurs, adaptation en environnement mobile et hétérogène, distribution basée sur la qualité de contexte, et gestion du cycle de vie des informations de contexte. Ils présentent une architecture en trois couches : la gestion de contexte comprenant la représentation et le traitement des informations de contexte, la livraison des informations de contexte comprenant la dissémination et le routage, et l'adaptation pendant l'exécution tout au long du processus de distribution.

Le système de gestion de contexte PACE [15] est composé d'un ensemble réparti de répertoires d'informations de contexte. Chaque répertoire gère une collection de modèles de contexte définissant des situations de contexte. Les composants sensibles au contexte découvrent ces répertoires dynamiquement. Cette approche ne prend pas en compte le passage à l'échelle et ne traite pas la tolérance aux fautes.

Les auteurs de [24] décrivent le service de gestion de contexte de l'intergiciel MADAM. Leur approche est basée sur la diffusion de messages périodiques qui servent à la formation de groupes et à la diffusion des informations de contexte. L'un des membres du groupe est le gestionnaire du groupe ; il s'occupe de la diffusion des messages périodiques et interagit avec les gestionnaires des autres groupes pour la distribution des informations de contexte.

Dans SALES [6], les auteurs exploitent les communications dans une infrastructure fixe et dans une infrastructure *ad hoc*. Les nœuds sont organisés selon une architecture hiérarchique. Ils considèrent un arbre à trois niveaux pour la distribution des informations de contexte.

Les auteurs de [25] ont recours aux réseaux logiques pair-à-pair pour la distribution des informations de contexte. Une entité centralisée est utilisée comme registre pour les localisations des sources de contexte et comme composant de résolution des requêtes de contexte. Cette entité reste un nœud central à partir duquel les clients de contexte initient leurs requêtes.

Les auteurs de [23] proposent une approche de gestion de contexte multi-domaine. Un domaine local couvre une zone géographique et un domaine applicatif gère les composants ayant des critères dépendants de l'application. Chaque domaine est administré par un serveur, et la coordination et la distribution des informations de contexte entre les domaines est effectuée par un ensemble de services Web.

Aucun des travaux précités ne définit de formalisme qui permet de modéliser des contrats de contexte au niveau du service de gestion de contexte.

3.2 Conception par contrat

Dans cette section, nous présentons un ensemble de modèles et de formalismes de contrats utilisés dans des domaines différents de la gestion de contexte. Le premier contrat logiciel a été défini par Meyer dans [20] et est inspiré de la logique de Hoare [16] et des ADT (en anglais, *Abstract Data type*) [13]. Meyer considère l'objet comme une structure de données et associe à chaque opération une pré- et une post-

condition contraignant les paramètres et la valeur de retour. Ce contrat présente l'intérêt d'exprimer les conditions d'utilisation d'un objet en clarifiant les obligations et les garanties. C'est l'approche dite de « conception par contrat ».

Dans [14], les auteurs considèrent que le système est composé de groupes d'objets. Ils définissent des contrats de collaboration comme des ensembles d'objets participants et de leurs obligations. Deux sortes de contraintes sont exprimées : le participant doit offrir certaines variables et interfaces et le participant doit exécuter une séquence d'actions ordonnées causalement. Les contrats définissent des pré-conditions et des invariants. C'est l'approche « conception orientée interaction ».

Les auteurs de [18] proposent une approche à base de contrats pour la gestion dynamique de services. Le contrat définit les participants, la description du service et des garanties de qualité de service. Les auteurs présentent un modèle orienté objet de contrat qui autorise l'ajout, la modification ou le retrait de certaines garanties afin qu'elles soient conformes aux paramètres de QoS courants. C'est l'approche « accord de niveau de service ».

Dans [22], l'auteur présente un méta-modèle de contrat composé de clauses et d'un accord. Chaque clause est associée à un participant et définit une spécification organisée selon le paradigme hypothèse–garantie. L'accord exprime la compatibilité des spécifications contenues dans les clauses.

Dans une approche IDM, les invariants sont des contraintes sur les éléments définissant un méta-modèle et les opérations à spécifier sont les opérations d'interaction entre les éléments du modèle. Les auteurs de [4] définissent des contrats en utilisant OCL (en anglais, *Object Constraint Language*).

Toutes les approches à base de contrats présentées ci-dessus consistent à spécifier le contrat par la définition des invariants sur les entités logicielles et des pré- et des post-conditions à leurs opérations. Le but est de spécifier ce que font ces éléments, comment les utiliser correctement et s'assurer qu'ils fonctionnement correctement. Dans le reste du papier, nous présentons notre modélisation de contrat de contexte et notre architecture de gestion de contexte répartie.

4. CONTRAT DE CONTEXTE

L'approche de modélisation de gestion de contexte par contrats permet d'établir des interactions entre les clients et le service de gestion de contexte, sans ambiguïté. Nous détaillons notre solution de contrat de contexte dans la section 4.1. Dans la section 4.2, nous présentons le méta-modèle de contexte prenant en compte le contrat de contexte.

4.1 Méta-modèle de contrat de contexte

Notre définition de contrat de contexte ainsi que notre méta-modèle de contrat sont inspirés de [22]. Nous définissons le contrat de contexte comme étant « la spécification d'expressions d'échanges d'informations de contexte, ces expressions étant décrites dans un langage de description et vérifiées pendant l'exécution pour garantir la propriété de compatibilité lors de l'appariement entre informations de contexte produites et contraintes de contextes exigées ».

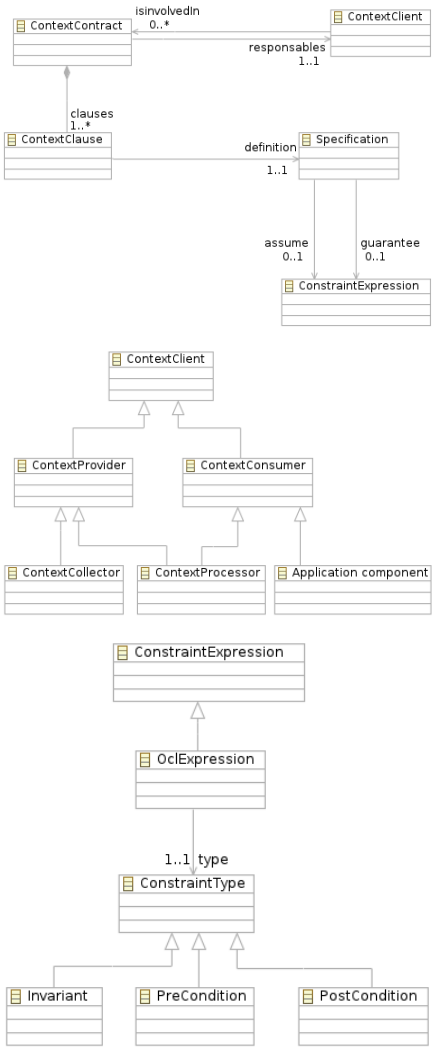

Figure 2: Méta-modèle du contrat de contexte

La figure 2 montre le méta-modèle de contrat de contexte. Un contrat de contexte est constitué de clauses de contexte, chacune relative à une spécification organisée selon le paradigme hypothèse–garantie. Les responsables du contrat de contexte sont les clients de contexte qui peuvent être des fournisseurs ou des consommateurs de contexte. Un fournisseur de contexte est un collecteur de contexte ou un processeur de contexte. Un consommateur de contexte est un processeur de contexte ou un composant de l'application sensible au contexte.

Nous mettons en œuvre le contrat de contexte durant la phase de conception. Les développeurs de l'application sensible au contexte établissent les contrats de contexte pendant le processus de conception ; sont particulièrement concernés les rôles « context specifier » et « context-awareness designer » [22]. Nous utilisons le modèle de contraintes définies par OCL pour la modélisation du contrat de contexte. Nous avons choisi d'utiliser le standard OCL comme langage d'expression de contrats pour plusieurs raisons. Tout d'abord, OCL est un langage d'expression de contraintes et un contrat est un ensemble de contraintes. Ensuite, notre but est de définir un contrat, indépendamment des plates-formes et des outils de modélisation : OCL est un standard utilisé sur diverses sortes de modèles telles que UML, MOF et pour la plateforme Eclipse / EMF. OCL est également utilisé comme langage de requête.

4.2 Méta-modèle de contexte

Nous montrons dans la figure 3 les principaux concepts du méta-modèle de contexte proposé dans [26] que nous reprenons. Une entité observable est une entité physique ou logique qui peut être observée. Pour une entité observable, plusieurs observables peuvent donner lieu à des observations. Un observable est une abstraction qui définit quelque chose à observer. Un observable interprété est obtenu par interprétation à partir d'un ou de plusieurs observables. Dans l'application de transport multimodal en ville, des exemples d'entités observables sont l'utilisateur, les téléphones mobiles et la météo. Ces différentes entités observables sont en relation. Par exemple, l'utilisateur Jérôme est en rendez-vous avec un autre utilisateur de l'application. Jérôme est équipé d'un téléphone mobile géolocalisé. Des observables associés à un utilisateur sont sa localisation, sa destination et ses préférences pour le calcul de l'itinéraire (partage de son trajet avec ces amis, prise en compte des prévisions météo, etc.).

Comme nous le montrons dans les sections 4.1, nous nous référons à OCL pour exprimer les expressions de contraintes de contexte des contrat de contexte. Ces contraintes de contexte sont associées à des instances d'observables. Ces associations sont nommées dans la figure 3 RelatesTox. La figure 4 montre des exemples de contrat de contexte. ContratUi1 et ContratUi2 sont des contrats de consommateurs. Ils contiennent les informations sur les différents observables de l'entité observable de la catégorie Utilisateur tels que : la localisation actuelle, la destination désirée, la préférence pour le partage de parcours avec d'autres utilisateurs et la préférence pour la prise en compte des conditions météorologiques. ContratMETEO est un exemple de contrat fournisseur. Il contient les informations de prévision météorologique de l'entité observable Météo.

5. ARCHITECTURE DE GESTION DE CONTEXTE RÉPARTIE À BASE DE CONTRATS DE CONTEXTE

La figure 5 montre l'architecture répartie du service de gestion de contexte. Les fournisseurs de contexte sont des collecteurs ou des processeurs de contexte. Ils exécutent deux types d'opérations : advertise(ContextContract) pour annoncer un contrat de contexte, et publish(ContextData) pour publier leurs informations de contexte. Comme visualisé dans la figure 5, le service réseau social, le service météo, les services de gestion de trafic, le téléphone géolocalisé, les services de gestion de parkings et les services de gestion de parcs de vélos sont des fournisseurs d'informations de contexte.

Les routeurs de contexte ou courtiers, notés Bx dans la figure 5, distribuent les contrats de contexte vers les autres courtiers pour mettre à jour les tables de routage et propagent les informations de contexte vers les autres routeurs. Les courtiers de bordure du gestionnaire de contexte transmettent les informations de contexte aux courtiers locaux pour notifier les consommateurs intéressés avec l'opération notify(ContextData). Ils procèdent donc à deux types d'appariement pour assurer que les contrats seront respectés. Le premier appariement est établi entre le contrat de contexte du consommateur et le contrat de contexte du fournisseur ; le second appariement est établi entre les contrats de contexte stockés au niveau des tables de routage et les événements (informations de contexte) publiées.

Les consommateurs de contexte sont des composants de l'application ou des processeurs de contexte. Ils expriment leurs requêtes de contexte via l'opération subscribe(ContextContract). Comme montré dans la figure 5, le service réseau social est un fournisseur et un consommateur d'informations de contexte. Les processeurs de contexte, notés PCx, sont des processeurs d'événements d'un CEP, qui utilise le SRBE. Par exemple, un processeur de contexte permet le calcul par série temporelle de la fréquence de passage d'un bus à un arrêt ou encore permet la détection d'incidents.

Par conséquent, notre gestionnaire de contexte est un SRBE formé à la base d'un réseau de courtiers. Au dessus, est construit un réseau de processeurs de contexte réifiés en processeurs d'événements d'un CEP. Ces derniers exécutent des traitements complexes pour inférer des informations de contexte abstraites et des situations. Les sources, les utilisateurs, et les processeurs de contexte interagissent avec le SRBE à travers l'interface publier/souscrire et définissent des contrats de contexte. Comme montré dans la figure 5, le gestionnaire de contexte est donc constitué des courtiers et des processeurs de contexte ; les utilisateurs du service de gestion de contexte sont les sources de contexte et les applications des utilisateurs finaux.

6. CONCLUSION ET PERSPECTIVES

Nous avons développé une solution PCM pour la gestion de contexte répartie. Nous avons présenté un méta-modèle de contrat de contexte qui sert à exprimer les contraintes de contexte partagées entre les clients de contexte et le service de gestion de contexte. Nous montrons aussi que le contrat de contexte sert à la répartition de la gestion de contexte en utilisant un SRBE.

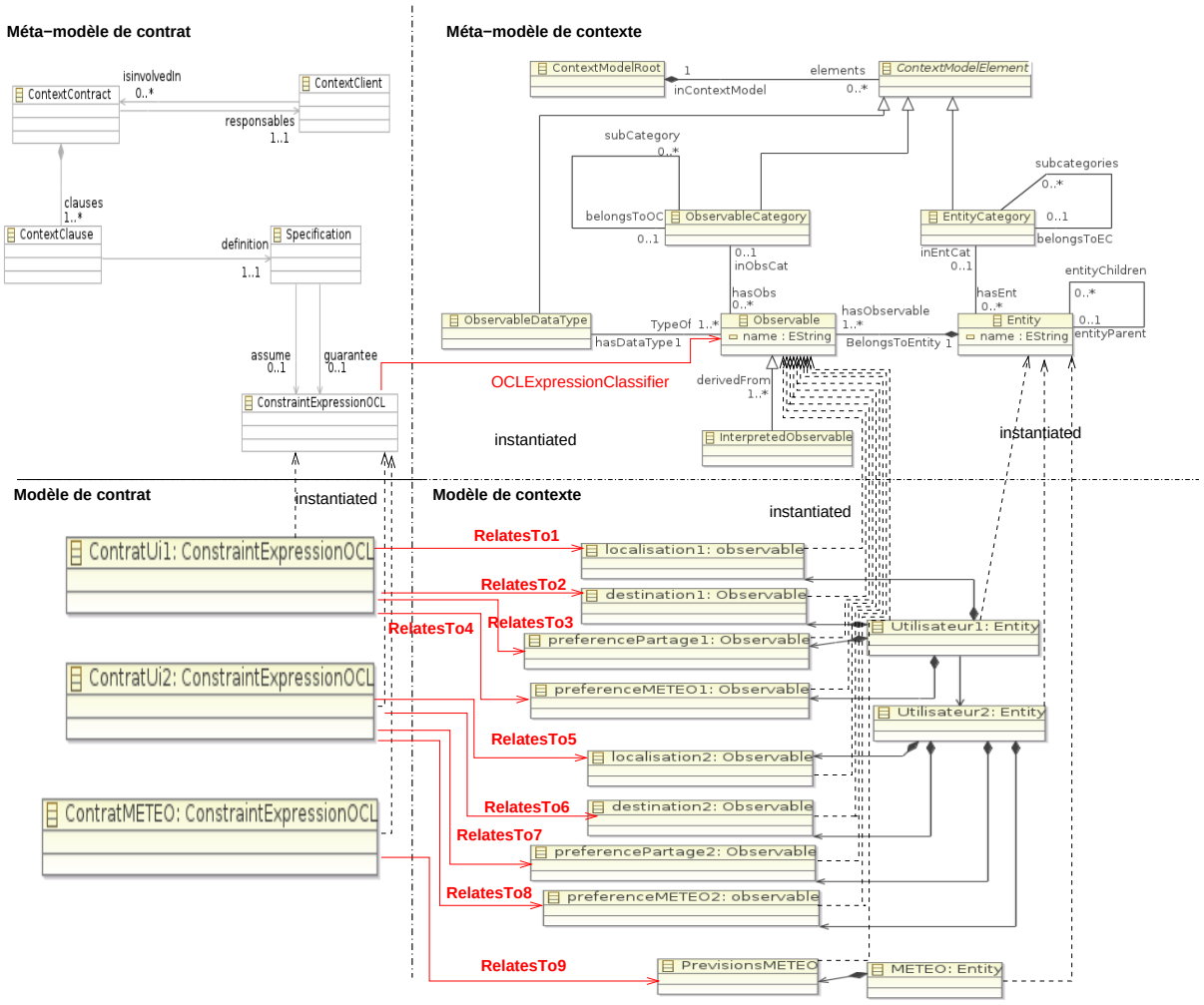

Figure 3: Modélisation contexte et contrat de contexte

Context ContratUi1 **{**

Invariant **self−>includes** (localisation1) **and**
self−>includes (destination1) **and**
self−>includes (preferencePartage1) **and**
self−>includes (preferenceMeteo1)

}

Context ContratMETEO **{**

Invariant **self−>includes** (previsionsMETEO)

}

Context ContratUi2 **{**

Invariant **self−>includes** (localisation2) **and**
self−>includes (destination2) **and**
self−>includes (preferencePartage2) **and**
self−>includes (preferenceMeteo2)

}

Figure 4: Exemples de contrats de contexte

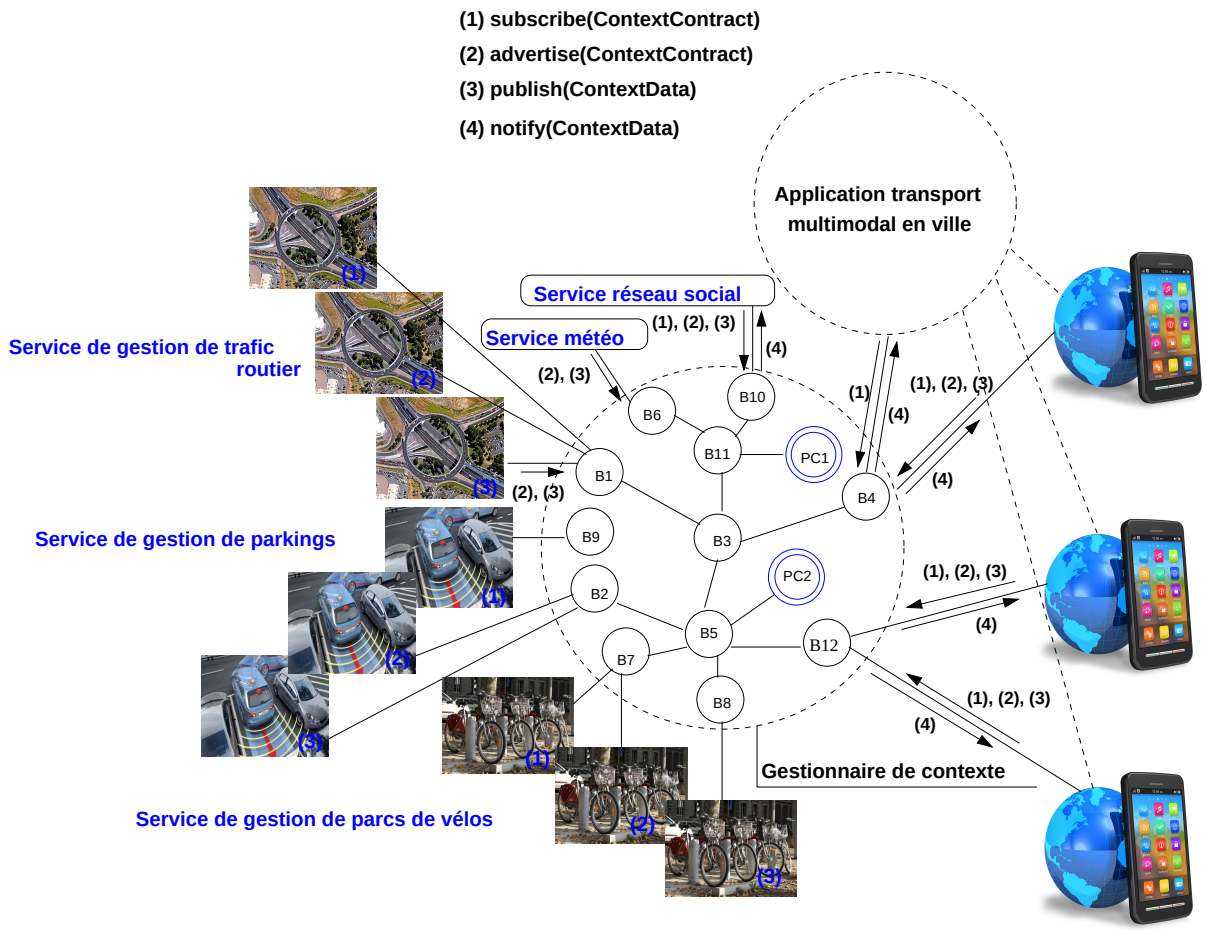

Figure 5: Architecture de contexte répartie à base de contrats de contexte

Dans nos travaux futurs, nous implanterons notre approche avec SRBE et nous utiliserons un CEP pour l'auto-adaptation des contrats. Nous envisageons aussi de mettre en œuvre un mécanisme de gestion de placement des agents de traitement complexe d'événements au sein du service de gestion de contexte.

Remerciements. Ce travail est en partie effectué dans le cadre du projet INCOME (ANR-11-INFR-009, 2012-2015) de l'Agence Nationale de la Recherche (ANR).

7. REFERENCES

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

[2] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv.*, 44(4) :24 :1–24 :45, Sept. 2012.

[3] A. Bouzeghoub, C. Taconet, A. Jarraya, N. Do, and D. Conan. Complementarity of Process-oriented and Ontology-based Context Managers to Identify Situations. In *Proc. ICDIM*, pages 222–229, June 2010.

[4] E. Cariou, C. Ballagny, A. Feugas, and F. Barbier. Contracts for model execution verification. In *Proc. European Conference on Modelling Foundations and Applications*, pages 3–18, 2011.

[5] D. Conan, R. Rouvoy, and L. Seinturier. Scalable Processing of Context Information with COSMOS. In *Proc. IFIP DAIS*, pages 210–224, June 2007.

[6] A. Corradi, M. Fanelli, and L. Foschini. Adaptive context data distribution with guaranteed quality for mobile environments. In *Proc. IEEE ISWPC*, pages 373–380, 2010.

[7] J. Coutaz, J. Crowley, S. Dobson, and D. Garlan. Context is Key. *CACM*, 48(3) :49–53, 2005.

[8] J. Coutaz and G. Rey. Foundations for a Theory of Contextors. In *Proc. 4th Int. Conf. on Computer-Aided Design of User Interfaces*, 2002.

[9] A. Dey and G. Abowd. Towards a better understanding of context and context-awareness. In *Proc. Workshop on the What, Who, Where, When and How of Context-Awareness*, 2000.

[10] A. Dey, G. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *HCI*, 16(2) :97–166, Dec. 2001.

[11] O. Etzion and P. Niblett. *Event Processing in Action*. Manning, 2011.

[12] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, pages 1–22, Mar. 2003.

[13] J. Guttag. Abstract data types and the development of data structures. *CACM*, 20(6) :396–404, June 1977.

[14] R. Helm, I. M. Holland, and D. Gangopadhyay. Contracts : Specifying behavioral compositions in object-oriented systems. In *Proc. ACM OOPSLA*, 1990.

[15] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam. Middleware for Distributed Context-Aware Systems. In *Proc. DOA*, pages 846–863, 2005.

[16] C. Hoare. An axiomatic basis for computer programming. *CACM*, 12(10) :576–580, Oct. 1969.

[17] INCOME. Étude des scénarios et exigences. Document de la Tâche 2.2 du projet INCOME, May 2013.

[18] A. Keller, G. Kar, H. Ludwig, A. Dan, and J. Hellerstein. Managing dynamic services : A contract-based approach to a conceptual architecture. In *Proc. IEEE NOMS*, 2002.

[19] A. Margara and G. Cugola. Processing flows of information : from data stream to complex event processing. In *Proc. 5th ACM DEBS*, pages 359–360, 2011.

[20] B. Meyer. Applying design by contract. *IEEE Computer*, 1992.

[21] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag, 2006.

[22] A. Ozanne. *Interact : un modèle général de contrat pour la garantie des assemblages de composants et services*. PhD thesis, Université Pierre & Marie Curie — Paris VI, Nov. 2007.

[23] F. Paganelli, G. Bianchi, and D. Giuli. A context model for context-aware system design towards the ambient intelligence vision : experiences in the eTourism domain. In *Proc. ERCIM Conf. on User interfaces for all*, pages 173–191, 2007.

[24] N. Paspallis, A. Chimaris, and G. Papadopoulos. Experiences from developing a distributed context management system for enabling adaptivity. In *Proc. IFIP DAIS*, pages 225–238, 2007.

[25] C. Reichert, M. Kleis, and R. Giaffreda. Towards Distributed Context Management in Ambient Networks. In *Proc. IST Mobile & Wireless Communications Summit*, 2005.

[26] C. Taconet and Z. Kazi-Aoul. Building Context-Awareness Models for Mobile Applications. *JDIM*, 8(2) :78–87, Apr. 2010.