

L'adaptation dans les systèmes d'information sensibles au contexte d'utilisation: approche et modèles

Le projet SECAS : Simple Environment for Context Aware Systems

Tarak Chaari, Frédérique Laforest
LIRIS
INSA de Lyon, France
{tarak.chaari, frederique.laforest}@liris.cnrs.fr

Résumé

Le projet SECAS (Simple Environment for Context Aware Systems) s'intéresse à l'adaptation des applications au contexte d'utilisation (préférences et environnement de l'utilisateur, terminal utilisé...). Notre objectif est de rendre les applications adaptables aux différents contextes d'utilisation sur leurs trois composantes : données, services et présentation. Dans cet article, nous présentons l'architecture de notre plateforme SECAS, l'approche d'adaptation que nous avons élaborée et les modèles nécessaires à la conception d'une application adaptable au contexte d'utilisation.

Mots clés

Sensibilité au contexte, adaptation de comportement, services Web

1. Introduction

De nos jours, de nouveaux besoins en systèmes d'information sont apparus. L'utilisateur d'une application veut avoir l'information n'importe où qu'il soit et à tout moment. Ceci a incité les développeurs à intégrer les terminaux mobiles dans leurs applications, donnant ainsi naissance à de nouveaux systèmes d'information dits pervasifs ou ubiquitaires [10].

Dans ce genre de système, une adaptation de l'application à un ensemble de paramètres (type de terminal, état de connexion, utilisateur connecté...) doit être assurée pour garantir une utilisation confortable. Tous ces paramètres forment un contexte d'utilisation particulier. Dans différents contextes, les utilisateurs accèdent aux mêmes données et aux mêmes services mais reçoivent des réponses qui peuvent être différentes ou présentées différemment ou à des niveaux de détails différents. Par exemple, un docteur

consulte le dossier médical d'un patient à l'hôpital à l'aide d'un ordinateur de bureau. Dans un autre contexte, il consulte le même dossier mais chez le patient à l'aide d'un ordinateur de poche. Il peut aussi avoir une synthèse vocale du même dossier avant une opération chirurgicale.

Ces systèmes (dit sensibles au contexte) doivent avoir la possibilité de percevoir la situation de l'utilisateur dans son environnement et d'adapter par conséquent tout le comportement du système à la situation en question : ses services, ses données et son interface utilisateur.

Le développement d'une application sensible au contexte nécessite de répondre à deux questions principales : (1) Comment concevoir une architecture garantissant l'adaptation au contexte dynamiquement au cours de l'exécution? (2) Comment concevoir l'application elle-même pour qu'elle s'adapte au contexte?

Le but de notre projet SECAS est de proposer une plateforme de conception et déploiement d'applications sensibles au contexte d'utilisation.

Après la présentation de l'état de l'art relatif à notre travail dans la section 2, nous proposons une nouvelle définition du contexte dans la section 3. Dans la section 4 nous présentons notre architecture de gestion de contexte. Dans la section 5 nous détaillons notre approche d'adaptation en présentant les modèles applicatifs que nous avons utilisés.

2. Etat de l'art

Les chercheurs dans le domaine de la sensibilité au contexte ont commencé par résoudre le problème de la mobilité de l'utilisateur (par exemple le projet Mobile IP [4]). Ensuite, ils ont approfondi leur recherche dans le domaine de sensibilité à la localisation de l'utilisateur. Teleporting [8] et Active Map [3] sont des exemples

d'applications sensibles à l'emplacement géographique de l'utilisateur. Ces systèmes utilisent une petite partie du contexte dont la portée dépasse de loin la simple localisation de l'utilisateur. Dey [2] est l'un des premiers chercheurs à avoir généralisé la notion de contexte. Il a spécifié 3 étapes nécessaires pour la sensibilité au contexte. En premier lieu, on doit capturer le contexte. Ensuite, on doit effectuer une interprétation du contexte pour passer à une représentation de haut niveau plus exploitable pour l'application. Finalement, on doit fournir cette information interprétée à l'application. Le Context toolkit de Dey (figure 1) est l'une des premières architectures qui prend en considération ces 3 étapes. Les senseurs capturent le contexte et le présentent aux widgets. Ces derniers utilisent des interpréteurs pour transformer les données capturées et les passer au serveur. L'application se connecte au serveur pour accéder aux données contextuelles.

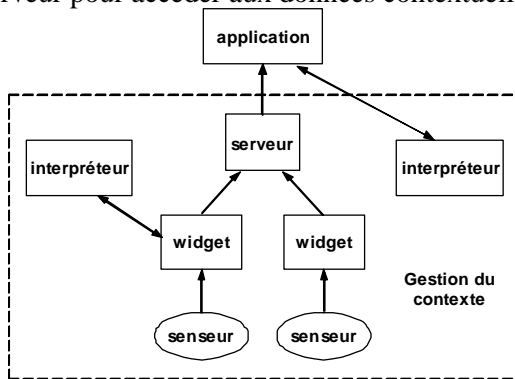


Figure 1. Le context toolkit de Dey [1]

Dans le domaine de la sensibilité au contexte, Dockhorn Costa [15] distingue quatre axes de recherche :

- a- *Les plates-formes conceptuelles* s'intéressent à l'aspect architectural des systèmes sensibles au contexte et fournissent des moyens faciles pour la capture, l'interprétation, et le transfert des données contextuelles à l'application. Context Toolkit [1] et Cooltown [18] sont des exemples de telles plates-formes.
- b- *Les plateformes de services* s'attachent à présenter les services pertinents à l'utilisateur selon le contexte en question. Ces plates-formes fournissent des outils pour la recherche et le déploiement de services. Elles tentent de résoudre les problèmes de passage à l'échelle et de sécurité. M3 [12] et Platform for Adaptive Applications [5] sont des exemples de contri-

butions dans ce domaine.

- c- *Les environnements d'interopérabilité* s'intéressent à la résolution du problème de l'hétérogénéité et de la mobilité de l'utilisateur. Ektara [16] et Universal Information Appliance [13] sont des exemples de projets dans ce domaine.
- d- *les environnements de développement* d'applications pervasives s'intéressent à la conception d'infrastructures physiques et logiques pour développer des systèmes pervasifs. PIMA [9] et Portolano [14] sont des exemples de projets dans cette direction.

Le tableau 1 présente une synthèse de ces axes de recherche et les solutions qu'ils proposent pour la sensibilité au contexte.

	Plate-forme conceptuelle	Plate-forme de services	Env. d'interopérabilité	Env. de développement
Hétérogénéité des terminaux			X	
Mobilité des terminaux			X	
Gestion du contexte	X	X		
Adaptation		X		X
Développement et déploiement		X		X

Tableau 1. Les axes de recherche et leurs contributions pour la sensibilité au contexte

En conclusion, nous pouvons dire que, dans les applications sensibles au contexte existantes, de gros efforts sont fournis pour définir comment capturer le contexte et le disséminer au système mais il n'y a pas une réponse précise pour savoir comment adapter l'application au contexte. C'est dans cette dernière direction que notre travail s'inscrit.

3. Définition du contexte

Les chercheurs dans le domaine de l'adaptation au contexte n'ont pas encore abouti à une définition à la fois générique et pragmatique de la notion de contexte, et plus précisément des paramètres constituant le contexte. En effet, toutes les définitions existantes sont soit très abstraites –ce qui rend la formalisation du contexte très difficile-, soit très spécifiques à un domaine particulier. La définition la plus complète et la plus adoptée par les chercheurs est celle de Dey [2]. Cependant, elle ne permet pas d'identifier les

données appartenant au contexte ni de les distinguer des données propres à l'application. Cette séparation nous semble cependant importante, avant d'entreprendre la conception d'un système sensible au contexte. En effet, le corps fonctionnel de l'application doit être conçu en faisant abstraction des données contextuelle. L'adaptation au contexte constitue une seconde étape qui peut être ajoutée après un développement indépendant du contexte. La séparation entre les données contextuelles et les données de l'application est également importante pour la modélisation du contexte. Pour garantir cette séparation, la limite entre ces deux types de données doit être clairement précisée. Cependant cette limite dépend du domaine de l'application : une donnée définie comme contextuelle dans un domaine peut être une donnée de l'application dans un autre domaine. Par exemple, les coordonnées GPS de l'utilisateur forment une donnée contextuelle dans le domaine de la télémédecine mais peut être une donnée applicative dans un système de régulation de trafic routier. De ce fait, nous proposons une nouvelle définition qui apporte, à notre avis, plus de précision sans toucher à la généralité du contexte d'utilisation.

D'une façon générale, le contexte décrit la situation de l'utilisateur en termes de localisation, de temps, d'environnement, de terminal utilisé, de profil utilisateur... De plus, les données contextuelles ne sont pas extraites d'un support de stockage de l'application et elles ne sont pas produites par des équipements ou des sources directement liées à l'application. D'un point de vue pratique, une donnée contextuelle n'est pas fournie par l'utilisateur lors de l'invocation des services de l'application.

Plus concrètement, nous définissons le contexte comme l'ensemble des paramètres externes à l'application pouvant influencer sur le comportement d'une application en définissant de nouvelles vues sur ses données et ses services. Ces paramètres ont un aspect dynamique qui leur permet d'évoluer durant le temps d'exécution. Ils ne sont pas significatifs à l'utilisateur final et doivent donc lui être transparents. Une nouvelle instance de ces paramètres caractérise une nouvelle *situation contextuelle* qui ne modifie pas les données de l'application mais qui peut mener à les traiter d'une façon différente. Par exemple, un utilisateur en déplacement veut avoir la liste des restaurants dans son

entourage. Sachant que cet utilisateur a des problèmes de santé et qu'il doit suivre un régime alimentaire précis, la liste des restaurants est filtrée pour ne donner que ceux qui proposent des plats adéquats. La liste des restaurants et les plats qu'ils proposent forment les données de l'application. L'information « l'utilisateur a un problème de santé » est une donnée contextuelle extraite du profil utilisateur.

Dans la suite de cet article, nous utilisons cette nouvelle définition avec ses nouvelles considérations (séparation entre la gestion du contexte et l'application) pour étudier l'influence du contexte sur l'architecture de l'application.

4. Architecture générale d'une application sensible au contexte

Dans ce paragraphe, nous présentons l'architecture générale du projet SECAS à travers la figure 2 et nous spécifions les 5 étapes nécessaires à la sensibilité au contexte. Nous associons un module dans l'architecture de l'application à chacune de ces étapes.

4.1. Capture du contexte

La capture du contexte se fait à l'aide de senseurs physiques qui génèrent des données brutes qui ne sont pas directement exploitable par l'application. Dans notre architecture nous définissons l'entité « context provider » (fournisseur de contexte) pour représenter un système de capture d'un paramètre du contexte.

4.2. Interprétation du contexte

Cette étape consiste à transformer les données contextuelles brutes en paramètres de haut niveau plus faciles à utiliser pour l'application. Par exemple, une adresse postale est beaucoup plus significative que des coordonnées GPS.

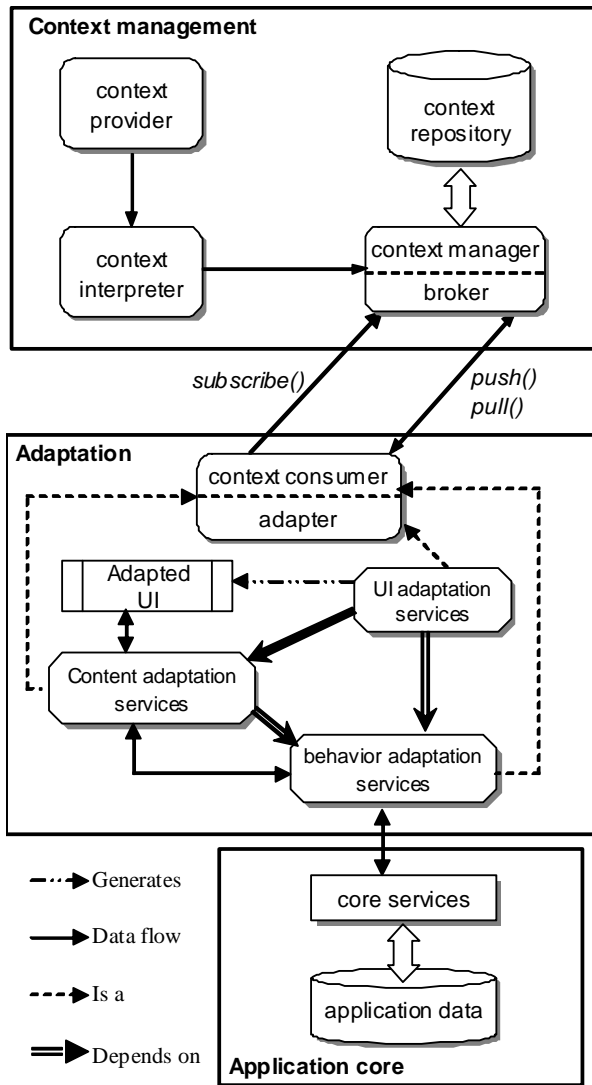


Figure 2. Architecture de la plateforme SECAS

4.3. Stockage du contexte

Nous proposons d'utiliser des représentations XML pour stocker et échanger les valeurs des paramètres contextuels. Nous définissons cinq facettes pour modéliser le contexte (réseau, utilisateur, terminal, méta-données et environnement). Nous définissons un élément XML pour chaque facette. L'ensemble des paramètres constituant chaque facette est défini par le concepteur de l'application en respectant des grammaires standards (comme CC/PP [11] pour les facettes utilisateur et terminal). Le concepteur doit définir la structure de chaque paramètre (ensembles, valeurs atomiques...).

4.4. Dissémination du contexte

Pour qu'elle soit sensible au contexte, l'application doit consommer une partie du contexte. Elle doit s'inscrire au «context broker» (courtier) qui transmet les paramètres contextuels pertinents pour chaque service de l'application. Lors de l'inscription, les services spécifient les paramètres contextuels qui modifient leurs comportements et leurs méthodes de dissémination (synchrone et asynchrone).

4.5. Adaptation au contexte

Après la phase d'abonnement au courtier, les services doivent s'adapter au contexte. Cette adaptation peut concerner trois niveaux différents de l'application : flux de données entre les services et l'utilisateur (adaptation de contenu), interface utilisateur (adaptation de présentation) et fonctionnement des services offerts à l'utilisateur (adaptation de comportement). Dans la suite de cet article, nous détaillons l'adaptation au contexte.

5. Adaptation de l'application au contexte

Pour conduire notre étude nous définissons une *entité conceptuelle* comme un triplet contenant un service, une interface graphique -qui assure l'interaction entre l'utilisateur et le service- et l'ensemble des données affichées ou saisies qu'elle engendre. Une application peut être modélisée par un ensemble de ces entités conceptuelles. L'adaptation de l'application revient alors à adapter les entités conceptuelles qui la composent, ce qui implique l'adaptation des trois composantes de chaque entité : données, présentation et service. Dans la suite de cette section, nous détaillons notre approche de l'adaptation des services puis nous présentons brièvement quelques outils d'adaptation de contenu et le processus d'adaptation de présentation.

5.1. Adaptation de comportement

Les entités conceptuelles présentent des dépendances fonctionnelles. En effet, les entrées d'un service peuvent dépendre des sorties d'un autre service. Nous modélisons un service par une fonction f qui prend en entrée un ensemble de paramètres $X=(x_1,x_2,\dots,x_m)$ et qui renvoie un

ensemble de données $R=(r_1, r_2... r_n)$ chaque paramètre de sortie r_i peut avoir plusieurs instances ($r_{i_a}, r_{i_b}...$). Cette définition générique permet de définir une structure d'échange commune entre fonctions ; ce qui facilite la composition et l'adaptation des services. On utilise la notation suivante : $f1 \rightarrow f2$ pour modéliser la dépendance « $f2$ dépend de $f1$ ». Ainsi, le service $f2$ ne peut être présenté à l'utilisateur que s'il a déjà invoqué le service $f1$. Nous définissons le modèle fonctionnel de l'application par l'ensemble de ses services et des dépendances fonctionnelles présentes entre eux. Ce modèle peut être représenté par un graphe orienté où les sommets sont les services et les arcs représentent les dépendances fonctionnelles. La racine du graphe est le premier service qu'on offre à l'utilisateur. Généralement, c'est un service d'authentification de l'utilisateur.

Pour garantir l'adaptation des services, nous proposons d'intégrer une couche d'adaptation fonctionnelle sur les services de l'application. Pour ce faire, chaque service doit être polymorphe et doit donc posséder plusieurs versions ou instanciations possibles suivant la situation contextuelle courante. Nous définissons une entité « adaptateur » (adapter) pour chaque service de l'application qui intercepte son invocation et qui applique un opérateur d'adaptation sur le service (choisir l'instance correspondant au contexte, verrouiller l'accès à un paramètre de sortie d'un service, filtrer les paramètres de sortie...). L'adaptateur est un consommateur de contexte (context consumer) et doit s'abonner au courtier dès sa mise en œuvre.

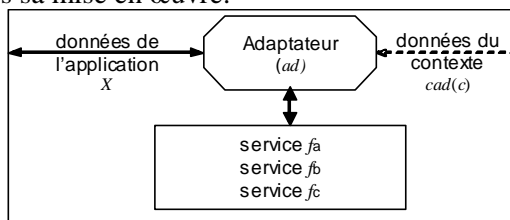


Figure 3. Adaptation de services

Chaque nœud du graphe fonctionnel adapté peut être représenté par la figure 3 et peut être modélisé par la fonction suivante : $ad(X, cad(c))/f_a, f_b...$ où ad est l'adaptateur au contexte, $\{f_a, f_b...\}$ l'ensemble des versions possibles du service initial non adapté, X est l'ensemble des paramètres d'entrée initiaux relatifs au domaine de l'application et $cad(c)$ est la vue du contexte c utile à ad pour effectuer l'adaptation.

5.2. Adaptation de contenu

L'adaptation de contenu consiste à modifier quelques propriétés des données présentées à l'utilisateur intéressé. Ces modifications sont assurées par un ensemble de services Web [6] d'adaptation de contenu. Nous avons défini un ensemble de transformations de contenu pour assurer cette adaptation :

- Adaptation de format : modifier ou complètement changer le format d'une donnée. Par exemple, réduire le nombre de couleurs d'une image.
- Traduction : pour adapter un texte au préférences de l'utilisateur.
- Compression de données : compression brute (par exemple zip), compression multimédia (par exemple jpeg) ou compression sémantique (par exemple résumé d'un long texte).
- décomposition/ agrégation de données : sélectionner une partie d'un objet multimédia ou une agrégation spatiale de données différentes (par exemple image plus texte) ou une agrégation temporelle (agrégation de séquences vidéos).

Pour effectuer l'adaptation de contenu au contexte, nous définissons un modèle XML de méta-données pour décrire toutes les propriétés (format, taille, versions...) de chaque donnée.

5.3. Adaptation de présentation

Pour garantir l'adaptation de la présentation d'une entité conceptuelle, nous proposons de générer automatiquement les interfaces graphiques suivant les caractéristiques réseau, les caractéristiques du terminal et les paramètres d'entrée et de sortie du service de l'entité. Pour chaque service, on génère un modèle logique de présentation décrivant les composants graphiques qui vont engendrer les données d'interaction avec le service. Ensuite, les composants physiques de présentation et leur disposition sur l'écran sont déterminés en utilisant un modèle physique de présentation et de disposition. Nous avons conçu et développé une plateforme de génération automatique d'interfaces graphiques pour des environnements multi-terminaux et multi-utilisateurs [17]. Cette plateforme permet de générer automatiquement une interface graphique pour des services Web.

6. Conclusion

Dans cet article, nous avons présenté notre pla-

teforme SECAS qui assure l'adaptation des applications au contexte d'utilisation. Nous avons proposé une nouvelle définition de la notion de contexte qui sépare les données de l'application des paramètres du contexte. En se basant sur cette définition, nous avons élaboré une architecture garantissant la sensibilité des applications au contexte d'utilisation. Nous avons présenté notre approche d'adaptation sur les trois dimensions des applications (données, présentation et services). Finalement, nous avons proposé de modéliser le corps fonctionnel d'une application par des services Web car ils forment un moyen facile, flexible et réutilisable pour le déploiement d'applications et l'intégration de l'adaptation dans plusieurs niveaux du corps fonctionnel de l'application.

Dans la suite de notre travail, nous envisageons de développer nos services d'adaptation en utilisant les patrons de conception [7] au niveau des adaptateurs. Ensuite, nous souhaitons valider notre plateforme en l'appliquant au domaine médical.

7. Références

- [1] A. K. Dey, D. Salber and G. D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction Journal* 16(2-4), pp. 97-166, 2001.
- [2] A. K. Dey, G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, The Hague, The Netherlands, April 2000.
- [3] B. N. Schilit and M. M. Theimer. Disseminating Active Map Information to Mobile Hosts. *IEEE Network* 8(5) pp. 22-32, 1994.
- [4] C. E. Perkins and D. B. Johnson. Mobility Support in IPv6. *Proc. 2nd Annual International Conference on Mobile Computing and Networking*, pp. 27-37, White Plains, NY, November 1996.
- [5] C. Efstathiou, K. Cheverst, N. Davies and A. Friday. An Architecture for the Effective Support of Adaptive Context-Aware Applications. *Proc. 2nd Int. Conf. in Mobile Data Management (MDM'01)*, Hong Kong, pp. 15-26, January 2001.
- [6] D. Austin, A. Barbir, C. Ferris, S. Garg. Web Services Architecture Requirements, W3C Working Draft, 19 August 2002, <http://www.w3.org/TR/2002/WD-wsaregs20020819>.
- [7] E. Gamma, R. Helm, R. Johnson and J. Vlissied. *Design Patterns*, Addison Wesley, 1995.
- [8] F. Bennett, T. Richardson and A. Harter. Teleporting - Making Applications Mobile. *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pp. 82-84, Santa Cruz, California, December 1994.
- [9] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman and D. Zukowski. Challenges: An Application Model for Pervasive Computing. *Proc. 6th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom 2000)*, pp. 266-274, Massachusetts, USA, August 2000.
- [10] J. Birnbaum, Pervasive Information Systems, *Communications of the ACM* 40 no 2, february 1997.
- [11] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henriksen. Experiences in Using CC/PP in Context-Aware Systems. *Proc. 4th Intl. Conf. on Mobile Data Management*, Melbourne, Australia, pp. 247-261, January 2003.
- [12] J. Indulska, S. W. Loke, A. Rakotonirainy, V. Witana, A. Zaslavski. An Open Architecture for Pervasive Systems. *Proc. 3rd Intl. Work. Conf. on Distributed Applications and Interoperable Systems (DAIS 2001)*, Kraków, Poland, pp. 175-188, 2001.
- [13] K. Eustice, T. J. Lehman, A. Morales, M. C. Munson, S. Edlund and M. Guillen. A Universal Information Appliance. *IBM Systems Journal*, 38(4), pp. 575-601, 1999.
- [14] M. Esler, J. Hightower, T. Anderson, and G. Borriello. Next Century Challenges: Data-Centric Networking for Invisible Computing. *Proc. 5th Annual Intl. Conference on Mobile Computing Networking (MobiCom'99)*, August 1999.
- [15] P. Dockhorn Costa. *Towards a Services Platform for Context-Aware Applications*. Master Thesis. University of Twente, The Netherlands, 2003.
- [16] R. W. DeVaul, A. S. Pentland. *The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications*. MIT Technical Report, 2000.
- [17] T. Chaari and F. Laforest. Génération et adaptation automatiques des interfaces utilisateurs pour des environnements multi-terminaux. *Revue Ingénierie des Systèmes d'Information, n° spécial Systèmes d'information pervasifs*, vol 2, 2004.
- [18] T. Kindberg and J. Barton. A Web-Based Nomadic Computing System. *Computer Networks*, Elsevier, 35(4), pp. 443-456, 2001.

8. Remerciements

Nous tenons à remercier le professeur Augusto Celentano pour sa collaboration et sa participation active dans l'élaboration du projet SECAS.