Architecture Logicielle Conceptuelle pour la Capture de Contexte

Laurence Nigay[1] et Phil Gray

Computing Science Dept, University of Glasgow
17 Lilibank Gardens, glasgow G128RZ Scotland
{laurence, pdg}@dcs.gla.ac.uk

RESUME
Cet article décrit l'application du modèle d'architecture logicielle PAC-Amodeus pour la conception logicielle de systèmes interactifs qui exploitent le contexte physique de l'utilisateur. Notre architecture s'inspire d'une solution dédiée à l'interaction multimodale. En effet, nous partons du constat de similarité des problèmes à traiter entre l'interaction multimodale et un système interactif sensible au contexte. Nous illustrons l'application de notre solution à deux systèmes et montrons la compatibilité de notre approche architecturale avec celles existantes.

MOTS CLES : Capture de contexte, Architecture logicielle Conceptuelle.

ABSTRACT
This paper describes a PAC-Amodeus software architectural solution for context-aware systems. The software architectural solution is based on an architecture dedicated to multimodal interaction. Indeed we argue in this paper that several software design issues are common between multimodal systems and context-aware systems. Our architectural solution is intentionally generic, intended to serve as the basis for a wide range of possible systems and compatible with the existing architectural approaches.

KEYWORDS : Context-Aware/Context-Sensitive System, Conceptual Software Architecture.

INTRODUCTION
La nécessité de prendre en compte le contexte dans les systèmes interactifs est devenue une évidence du moment que l'utilisateur n'était plus figé à son bureau mais devenait mobile.

Un numéro spécial de la revue HCI [9] consacré aux systèmes interactifs sensibles au contexte dresse un panorama des recherches : nous constatons qu'il n'y a pas une définition consensuelle du terme contexte. Cet article n'ayant pas d'objectif terminologique, nous reprenons notre définition du contexte [4] : ensemble de propriétés de phénomènes physiques qui peuvent être captées.

Dans cet article, nous présentons une solution architecturale pour la conception logicielle de systèmes interactifs sensibles au contexte. Notre solution repose sur le constat que lors de la capture de contexte, des données en provenance de capteurs différents doivent être fusionnées [4]. De plus des capteurs peuvent fournir des informations redondantes qui augmentent la certitude de l'information fusionnée résultante (coefficient de confiance plus élevé), tandis que d'autres fournissent des informations complémentaires. Enfin un capteur peut être assigné à fournir une information particulière du contexte. Fusion, coefficient de confiance, propriétés CARE (Complémentarité, Assignation, Redondance et Equivalence), tant de concepts manipulés pour l'interaction multimodale [5] ! Aussi notre solution est adaptée d'une architecture dédiée à l'interaction multimodale : le modèle d'architecture PAC-Amodeus couplé au moteur de fusion décrit dans [5].

SOLUTION ARCHITECTURALE : PAC-AMODEUS ETENDU
Comme le montre la Figure 1, le modèle PAC-Amodeus préconise une décomposition selon les cinq composants logiciels du modèle Arch [1]. Le Contrôleur de Dialogue (CD), clé de voûte du système interactif, prend en charge l'enchaînement des tâches et gère chaque fil de dialogue au moyen d'une hiérarchie d'agents PAC. Nous étendons le modèle en ajoutant deux composants dédiés à la capture du contexte. Tandis que le composant Contexte Physique est dédié aux traitements qui sont dépendants des capteurs physiques, le composant Contexte Logique contient les traitements d'abstraction des données en provenance des capteurs. L'un des rôles principal du Composant Contexte Logique est de rendre le Contrôleur de Dialogue indépendant des capteurs utilisés, augmentant ainsi la modifiabilité du code. Comme souligné dans [2],

[1] En année sabbatique, de l'Université de Grenoble 1, laboratoire CLIPS-IMAG, équipe IIHM

une forte modifiabilité du code est incontournable car la mise au point de systèmes sensibles au contexte utilisables repose nécessairement sur une approche itérative. Les deux composants, Contexte Physique et Contexte Logique, forment une nouvelle branche. Nous appliquons ici le mécanisme de branche hérité du modèle Arch [1]. Ainsi le Contrôleur de Dialogue, qui à l'origine était à l'écoute des informations en provenance du Noyau Fonctionnel et de l'utilisateur, doit gérer une nouvelle source d'informations, celles en provenance du contexte. Utilisateur, Noyau Fonctionnel et Contexte sont placés au même niveau comme trois sources d'informations pour le Contrôleur de Dialogue.

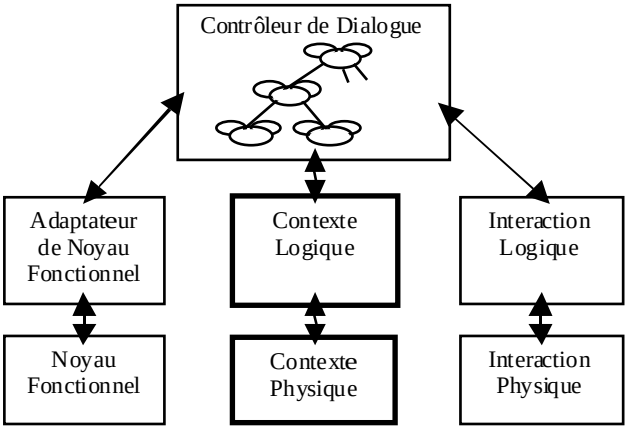

Figure 1 : Modèle PAC-Amodeus étendu pour la capture de contexte.

Exploitation du contexte dans le Contrôleur de Dialogue

Par analogie avec le moteur de fusion dédié à l'interaction multimodale, nous préconisons un Serveur de Contexte. Comme le montre la Figure 2, les services du Serveur de Contexte sont accessibles par la facette Contrôle de chaque agent PAC. (Un agent PAC contient trois facettes, une Abstraction A, compétence de l'agent, une Présentation P, partie perceptible par l'utilisateur et un Contrôle C maintenant les liens entre les deux facettes A et P.) Le Serveur de Contexte maintient les informations sur le contexte en provenance du ou des composants Contexte Logique. Il effectue les fusions nécessaires d'informations issues de capteurs distincts. Adoptant l'approche du moteur de fusion [5], les objets ou unités informationnelles que le Serveur de Contexte manipule, sont génériques : un objet correspond à un ensemble de cases à compléter par fusion. La sémantique attachée à chaque case n'est pas connue du serveur. Seule la composition structurelle doit être déclarée de manière externe au serveur. Cette technique, qui permet au serveur d'engendrer des formats d'objets à façon sans en modifier les traitements, le rend générique et permet d'envisager l'intégration du serveur dans plusieurs applications, chaque agent déclarant son besoin au Serveur de Contexte.

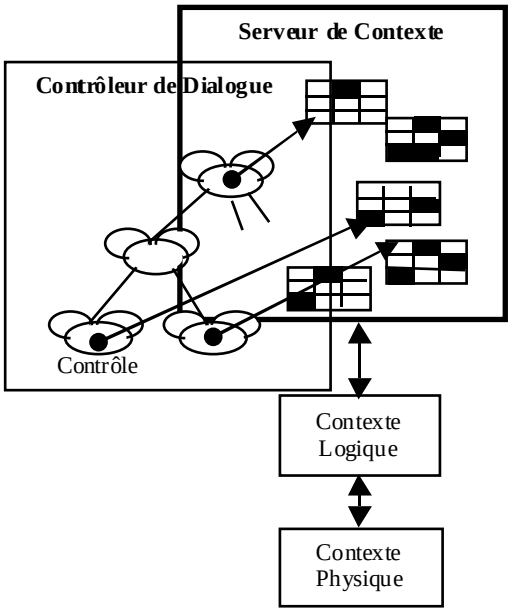

Figure 2 : Point d'ancrage du Serveur de Contexte au sein de l'architecture PAC-Amodeus.

Propriétés

Une architecture n'est jamais intrinsèquement bonne ou mauvaise, mais sa qualité se juge à la lumière de propriétés identifiées par avance. La solution architecturale préconise trois modules propres à la capture et gestion du contexte. Cette modularité augmente la modifiabilité du code et réduit la complexité de programmation. En particulier le composant Contexte Logique a un rôle d'abstraction ou d'interface logicielle entre le composant Contexte Physique, dépendant des capteurs, et les autres composants indépendants des capteurs. Ainsi lors du changement d'un capteur physique, notre solution architecturale localise les modifications au sein du composant Contexte Physique uniquement. De plus le Serveur de Contexte bien que propre à chaque application repose sur un mécanisme générique, les applications déclarant leurs besoins de façon externe au serveur. Notons que ce serveur est aussi indépendant des capteurs physiques grâce au composant Contexte Logique. Enfin la modularité concourt à une meilleure réutilisabilité : réutilisabilité d'un composant ou d'une partie comme celle dédiée à capturer les données d'un capteur physique dans le composant Contexte Physique.

DU MODELE A LA REALITE

La solution architecturale exposée ci-dessus a été appliquée à la conception logicielle de deux systèmes : CASPER et MAGIC. Pour les deux architectures, nous décrivons ici que les points saillants liés au contexte.

Architecture de CASPER

CASPER (*Computer Assisted PERicardial puncture*) est un système de chirurgie augmentée [3]. Le chirurgien réalise une ponction péricardique au moyen d'une aiguille de ponction alors que le système fournit des informations

de guidage en affichant la trajectoire optimale calculée avant l'opération dans un casque semi-transparent porté par le chirurgien. Aussi la direction du regard du chirurgien, l'aiguille ainsi que le patient sont localisés dans l'espace. L'architecture logicielle du système concernant la capture de contexte est présentée à la Figure 3.

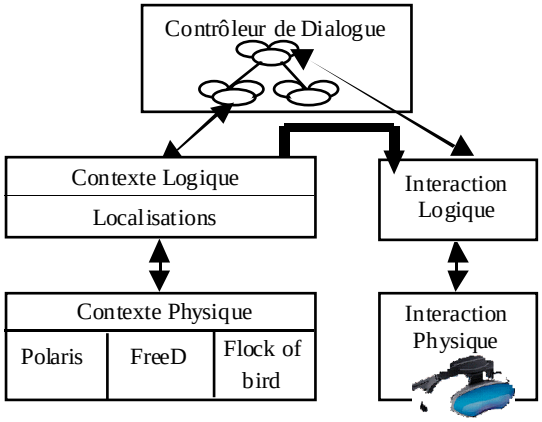

Figure 3 : Capture de contexte dans CASPER.

Le contexte se réduisant aux positions du regard du chirurgien, du patient et de l'aiguille directement acquises par le composant Contexte Logique, aucun traitement supplémentaire (comme la fusion) n'est nécessaire. Aussi, nous n'avons pas développé un composant Serveur de Contexte : les informations du composant Contexte Logique sont directement exploitées par les agents du Contrôleur Dialogue. L'application de notre solution architecturale a permis de facilement tester plusieurs localisateurs 3D (Polaris, FreeD et Flock of Bird) : seul le composant Contexte Physique a été modifié pour changer de localisateur. De plus un flux d'informations entre les composants Contexte Logique et Interaction Logique a été identifié. Il s'agit d'informations en provenance du contexte qui ne modifient pas l'état du dialogue de l'application, mais qui sont utiles pour un retour d'information du niveau lexical. Selon la position courante du regard du chirurgien, l'affichage de la trajectoire affichée dans le casque est modifiée sans que l'information transite par le Contrôleur de Dialogue. Notons que ce flux horizontal a déjà été identifié dans le cas d'interaction multimodale, par exemple, pour afficher la phrase dictée reconnue avant traitement par le moteur de fusion [5].

Architecture de MAGIC

MAGIC (*Mobile Augmented reality Group in Context*) est une plate-forme matérielle et logicielle destinée à une activité collaborative en situation de mobilité, et initialement conçue pour la prospection archéologique [6]. Nous sommes en cours de développement d'un jeu basé sur le troc qui repose sur la plate-forme MAGIC. La plate-forme matérielle conçue est un assemblage de dispositifs commercialisés : elle comprend une tablette et un casque semi-transparent. Une caméra est fixée entre les deux écrans du casque. La plate-forme contient aussi un magnétomètre qui détermine l'orientation de la caméra ainsi qu'un GPS qui localise l'utilisateur mobile dans le champ de fouille ou dans le terrain de jeu. La capture de contexte consiste en la position de l'utilisateur et son orientation. Aussi comme dans CASPER, nous n'avons pas développé de Serveur de Contexte. Les informations de localisation en provenance du composant Contexte Logique sont directement exploitées par le Contrôleur de Dialogue. L'architecture résultante de la Figure 4, complètement décrite dans [7] est similaire à celle de CASPER, néanmoins les capteurs de localisation sont différents.

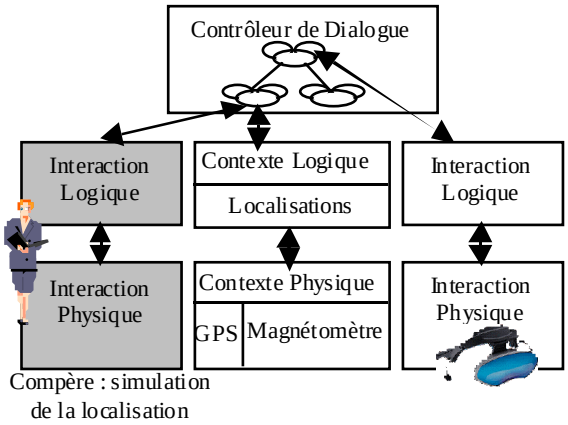

Figure 4 : Capture de contexte dans MAGIC.

Le mécanisme de localisation (GPS) n'étant pas assez précis, nous envisageons d'utiliser un compère humain (technique du magicien d'oz) pour simuler la localisation lors des premiers tests du jeu de troc basé sur la plate-forme MAGIC. Pour cela nous montrons à la Figure 4, comment nous concevons l'architecture : deux composants (en gris) sont ajoutés à la place des deux composants Contexte Logique et Physique. Le reste de l'architecture est inchangé.

ANALYSE DE LA SOLUTION ARCHITECTURALE

La validation d'une architecture logicielle conceptuelle est certes une entreprise difficile. Nous avons avancé un ensemble de propriétés véhiculées par la solution, comme la portabilité vis-à-vis des capteurs physiques utilisés. Néanmoins, l'application de la solution à la réalisation logicielle de deux systèmes n'est pas suffisante pour une validation. De plus les deux systèmes développés exploitent un contexte limité à la position et orientation de l'utilisateur qui n'a pas nécessité le développement du Serveur de Contexte. Une façon complémentaire de valider notre solution architecturale est de montrer qu'elle correspond à une pratique logicielle. Pour cela nous étudions notre solution architecturale au regard des approches de la littérature.

La solution architecturale proposée dans [4] préconise l'ajout d'une branche dédiée à la gestion du contexte au

sein du modèle Arch [1]. Notre solution basée sur le modèle PAC-Amodeus reprend cette approche en ajoutant une branche composée des deux composants Contexte Physique et Logique. Comme dans la solution [4], la connexion entre cette branche et le Serveur de Contexte peut être dynamique. Ainsi si un agent PAC du Contrôleur de Dialogue exprime le besoin d'une information du contexte, le serveur peut dynamiquement chercher cette information. Nous trouvons dans [8] une autre solution architecturale ajoutant une branche au modèle Arch. Les solutions de [4] et [8] proposent des décompositions différentes des deux composants Contexte Physique et Logique. Dans [4], les composants sont peuplés d'un réseau d'unités logicielles dont la structure est compatible avec celle des *widgets* de la boîte à outils pour le contexte [2]. Dans [8], quatre couches sont identifiés au sein de la branche dédiée au contexte. Les deux couches Capture et Transformation correspondent respectivement aux deux composants Contexte Physique et Logique, tandis que les deux couches Identification et Exploitation sont localisées dans notre Serveur de Contexte. Notons que notre solution affine la couche Exploitation en explicitant les liens entre les agents du Contrôleur de Dialogue et le Serveur de Contexte. En synthèse, les deux solutions [4, 8] sont compatibles et complémentaires avec notre solution, en affinant la structuration des deux composants Contexte Physique et Logique.

Une autre approche proposée par [10] repose sur une architecture *Blackboard*. Une application déclare ses besoins au *Blackboard*. Ce dernier maintient les structures de données déclarées par l'application et un ensemble d'*observers* sont chargés de collecter les données nécessaires. Cette solution, centrée sur les données, correspond à notre approche au niveau du Serveur de Contexte. En effet chaque agent PAC déclare de façon externe au Serveur de Contexte ses besoins. Tandis que la solution *Blackboard* est présentée comme une alternative à l'approche de [2], nous concilions les deux approches au sein notre solution architecturale.

CONCLUSION

Nous avons présenté une architecture conceptuelle pour des systèmes interactifs sensibles au contexte. Nous soulignons le niveau conceptuel de la solution à opposé au niveau implémentationnel. Ainsi le composant Serveur de Contexte du niveau conceptuel n'implique pas une architecture implémentationnelle particulière comme une architecture Client-Serveur ou Peer to Peer. Nous avons illustré l'architecture avec deux systèmes et l'avons située par rapport aux approches existantes. En particulier, nous avons montré que notre architecture permet de concilier deux approches existantes, présentées aujourd'hui comme deux solutions opposées [10]. Enfin notre architecture s'inspire d'une solution conçue pour l'interaction multimodale. En effet, la multimodalité a permis de souligner le rôle des méta données, par exemple le coefficient de certitude ; ces méta données étaient auparavant ignorées, comme lors de la capture d'un clic souris. Les méta données sont centrales dans la capture de contexte comme le soulignent les modèles [4, 8] : elles influencent les traitements au sein du Serveur de Contexte.

BIBLIOGRAPHIE

1. Bass, L., et al. *A Metamodel for the Runtime Architecture of an Interactive System*. SIGCHI Bulletin, ACM Press, 1992, pp. 32-37.
2. Dey, A., Abowd, G., Salber, D. *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*. Human-Computer Interaction (HCI) Journal, Vol. 16, Lawrence Erlbaum, 2001, pp. 96-166.
3. Dubois, E., Nigay, L., Troccaz, J. *Assessing Continuity and Compatibility in Augmented Reality Systems.* International Journal on Universal Access in the Information Society, special issue on Continuous Interaction, Springer-Verlag, 2002, à paraître.
4. Gray, P., Salber, D. *Modelling and Using Sensed Context Information in the Design of Interactive Applications.* Proceedings of EHCI 01, LNCS 2254, Springer-Verlag, 2002, pp. 92-111.
5. Nigay, L., Coutaz, J. *A Generic Platform for Addressing the Multimodal Challenge.* Proceedings of CHI'95, ACM Press, 1995, pp. 98-105.
6. Nigay, L., Salembier, P., Marchand, T., Renevier, P., Pasqualetti, L. *Mobile and Collaborative Augmented Reality: A Scenario based design approach.* Proceedings of Mobile HCI 02, Springer-Verlag, 2002, à paraître.
7. Renevier, P., Nigay, L. *Mobile Collaborative Augmented Reality, the Augmented Stroll.* Proceedings of EHCI 01, LNCS 2254, Springer-Verlag, 2002, pp. 315-334.
8. Rey, G., Coutaz, J. *Le Contexteur : une Abstraction Logicielle pour la Réalisation de Systèmes Interactifs Sensibles au Contexte.* Actes de la Conférence IHM 02, ACM Press, 2002, à paraître.
9. *Special Issue on Context-Aware Computing.* Human-Computer Interaction (HCI) Journal, Vol. 16, Lawrence Erlbaum, 2001.
10. Winograd, T. *Architectures for Context*. Human-Computer Interaction (HCI) Journal, Vol. 16, Lawrence Erlbaum, 2001, pp. 401-419.