Université Joseph Fourier

U.F.R Informatique
&
Mathématiques Appliquées

Institut National Polytechnique de Grenoble

ENSIMAG

I . M . A . G .

ECOLE DOCTORALE MATHEMATIQUES ET INFORMATIQUE

DEA D'INFORMATIQUE : SYSTEMES ET COMMUNICATIONS

Projet présenté par :
Gaëtan Rey

Systèmes interactifs sensibles au contexte

Effectué au laboratoire :

CLIPS-IMAG
Communication Langagière et Interaction Personne Système

Equipe IIHM
Ingénierie de l'Interaction Homme Machine

Jury
Gilles Privat, FT&RD Meylan
Patrick Reignier, GRAVIR
Yves Demazeau, LEIBNIZ
Joelle Coutaz, CLIPS

Date : le 19 juin 2001

Remerciements

Je tiens à remercier toutes les personnes qui ont permis que cette année de DEA se passe de la meilleure façon qui soit. Plus précisément tous mes remerciements à Joëlle qui a su m'encadrer et me mettre dans des bonnes conditions de travail, à Gaëlle pour avoir trouvé le temps de suppléer Joëlle quand elle n'est pas disponible et Jean Christophe qui a ouvert la voie sur le contexte durant sa deuxième année de magistère et m'a passé le flambeau pour mon D.E.A.

Je remercie aussi toutes les personnes qui de près ou de loin m'ont suivi pendant mon travail, notamment toute l'équipe IIHM, dans laquelle l'ambiance a toujours été au beau fixe, à ma famille qui a supporté mon manque de présence et aux membres de #jdr qui m'ont permis d'évacuer le stress accumulé le long de l'année.

Merci aussi à Bruno, Cédric, Frank, Jean-Philippe, Jérôme, Martial, Sébastien et Thomas pour m'avoir soutenu et conseillé le long de ces années.

Enfin, un remerciement spécial à *Christian Hardenberg*, Christophe et Philippe pour leur aide.

Table des Matières

Table des figures

Chapitre II :

Chapitre III:

Introduction

1 Le sujet

Cette étude porte sur les *systèmes interactifs sensibles au contexte*.

Si l'on se réfère aux ouvrages de référence, on entend par contexte :

?? « l'ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours » [Encyclopédie Larousse],

?? ou encore : «l'ensemble des circonstances dans lesquelles se produit un événement, se situe une action » [Encyclopédie Larousse],

?? mais aussi : « l'ensemble des éléments qui entourent un fait et permettent de le comprendre [© 2001 Hachette Multimédia / Hachette Livre].

Ces définitions font référence au discours ou à l'action. Les unes évoquent la situation en tant que telle, d'autres mentionnent l'impact du contexte sur la compréhension. Dans le cadre de cette étude, je fais le choix de privilégier l'action et de couvrir aussi bien la situation que son effet sur la compréhension. Le discours, et notamment les énoncés en langue naturelle, sortent de la portée de cette étude. Ce choix fait, j'appréhende *la notion de contexte en tant qu'ensemble de circonstances dans lesquelles se situe une action, ou qui entourent un fait et permettent de le comprendre*. A son tour, comprendre est une condition nécessaire à l'adaptation autre que la réaction-réflexe.

Déclinant cette définition pour l'Interaction Homme-Machine, je dirai qu'*un système interactif est sensible au contexte lorsqu'il est capable d'identifier les circonstances qui entourent l'action (et notamment celle de l'utilisateur) en vue d'offrir un service adapté*. L'adaptation peut être directement perçue par l'utilisateur. Mais elle peut aussi agir sur l'efficacité et la robustesse du système, servant alors l'utilisateur de manière indirecte. Par exemple, la détection d'un changement de lumière ambiante peut entraîner l'adaptation automatique de la luminosité de l'écran. Cette migration de tâche vers le système évite à l'utilisateur d'interrompre son activité centrale. De même, pour un système de surveillance médicale à distance, la détection d'une anomalie chez le patient est signalée au centre de surveillance. Ici, la tâche d'alerte qui ne peut être accomplie par l'utilisateur invalide, est déléguée au système. Dans le cas d'un système de suivi de doigt par vision par ordinateur, le changement des conditions lumineuses entraîne une adaptation des modèles établis à l'initialisation. Cette adaptation, nécessaire à la robustesse du système, assure la continuité du service de suivi. Elle sert l'utilisateur de manière indirecte.

Ayant introduit mon sujet dans sa généralité, il convient de préciser la problématique qui motive mes objectifs et la méthode adoptée pour les atteindre. Ce chapitre introductif s'achève avec la présentation de la structure du rapport.

2 Motivations

L'étude des systèmes sensibles au contexte trouve trois sources de justification : la transposition à l'interaction personne-système de l'apport de l'implicite dans la communication interpersonnelle ; l'importance de la notion de contexte en conception de systèmes interactifs mais sa dilution au cours du processus de développement ; la diversité des dispositifs d'interaction qui appelle de nouveaux usages fondés sur la situation interactionnelle.

2.1 Contexte et communication interpersonnelle

En communication interpersonnelle, la part du contexte est importante. Les gestes, les expressions faciales, les attitudes corporelles, la manipulation d'objets à proximité, les expériences partagées, le vécu journalier, etc. sont autant de facteurs implicites qui participent à la compréhension mutuelle des échanges explicites. En bref, l'utilisation d'information situationnelle implicite améliore la bande passante de la communication explicite. Elle la rend plus efficace.

Actuellement, les systèmes interactifs d'usage courant, sans accès au contexte, n'ont pas les moyens d'élaborer de l'information implicite. Au mieux, en proposent-ils des versions appauvries. La liste des derniers fichiers ou options utilisés des éditeurs est un exemple commun. Il s'agit d'une connaissance implicite puisqu'elle évite à l'utilisateur de spécifier explicitement certains paramètres, mais le service est pour le moins embryonnaire. Donner accès au contexte devrait permettre d'améliorer la bande passante des canaux d'échange entre l'utilisateur et le système en sorte de faire migrer vers le système des tâches et actions subalternes.

2.2 Contexte et processus de conception

En Interaction Homme-Machine, la considération pour la notion de contexte n'est pas un phénomène nouveau. La conception contextuelle (*contextual design*) de Beyer et Holtzblatt, fondé sur la récolte et l'organisation de données de terrain, prône un processus de conception centré sur l'utilisateur [Beyer 98]. L'analyse des données permet d'identifier la façon dont les gens travaillent en situation, de détecter les insuffisances et d'engager une reconception de la pratique professionnelle par le biais de support informatique. Ainsi, un système informatique n'est pas conçu sur la seule intuition de l'informaticien. Il résulte d'une analyse de la pratique de l'utilisateur observé par le biais d'enquêtes contextuelles (*contextual inquiry*).

La psychologie cognitive appliquée à l'Interaction Homme-Machine est encore largement marquée par le modèle GOMS [Card 83] selon lequel l'individu agit de manière rationnelle d'après un plan déterminé de tâches (ou buts) décomposables en sous-tâches (ou sous-buts). Cette approche ne laisse aucune place au comportement opportuniste, c'est-à-dire à l'improvisation, que justifie notamment l'inattendu. Les modèles de l'Action Située (*Situated Action*) [Suchman 87], la Théorie de l'Activité [Bardram 97] ou encore la Cognition Distribuée (*Distributed Cognition*) [Halverson 94] visent autre chose que la décomposition stricte de GOMS en considérant le contexte comme pièce maîtresse.

Si l'importance du contexte est acquise dans les méthodes de conception, un système reste conçu pour une situation d'usage donnée. Ainsi, on concevra un système de réservation de billet de train pour Internet et un second système de réservation pour les bornes interactives des gares ferroviaires. Si la base de données est commune, l'interface utilisateur (dialogue et présentation) est spécifique à chaque situation, conduisant à des surcoûts de développement et de maintenance. Inversement, un système de réservation sensible au contexte serait capable d'identifier les conditions

d'usage et d'adapter l'interaction en conséquence. Typiquement, pour un accès sur Internet à domicile, le système se permettrait quelques digressions en offrant la possibilité de consulter les promotions, tandis qu'à la gare, il s'agirait de conduire efficacement l'utilisateur à son but et sans erreur malgré les conditions de stress (plusieurs personnes attendent et le train part dans un quart d'heure !).

Au bilan, si le contexte est une notion admise en conception de systèmes interactifs, il n'est utilisé qu'en phase amont du processus de développement. Tout ce qui relève du contexte se dilue progressivement au cours du processus de développement et le triangle classique en IHM « utilisateur-tâche-machine » ne fonctionne que pour un contexte figé prévu à l'avance. Or, la technologie aidant, l'utilisateur est mobile, les dispositifs d'interaction (téléphones et assistants numériques personnels) aussi. De facto, le contexte d'utilisation d'un système change. Dans ces conditions, un système doit intégrer des capacités d'adaptation à ces changements.

2.3 Contexte et diversité des dispositifs d'interaction

Les dispositifs d'interaction se diversifient par la forme et la finalité. L'*ordinateur tout usage* de type calculateur de bureau se voit prolongé d'*appareils dédiés* comme les assistants personnels de bureau (PDA) et les téléphones mobiles. Inversement, avec la convergence télévision-informatique, un objet à finalité bornée devient un dispositif tout usage. Les PDA, comme des Legos, incluent progressivement les services de téléphonie et inversement, les fabricants de téléphone font du « portable » un véritable PDA. Avec l'ordinateur évanescent [Weiser 93], l'espace et les objets de la vie courante deviennent des entités d'interaction.

Il en résulte un foisonnement de solutions techniques qui correspondent chacune à des usages prévus ou émergents. Dans cet espace du possible, on observe une constante : l'utilisateur veut avoir le choix. Dès lors, il convient d'envisager l'utilisation mixte du gros calculateur comme du petit dispositif, du fixe comme du mobile, de l'ordinateur palpable à l'ordinateur évanescent. On assiste progressivement à la création d'un tissu informationnel et computationnel extrêmement malléable. Mais cette plasticité [Thevenin 99] dépend de l'accès au contexte. Voici quelques exemples simples à imaginer ou sur le point d'être disponibles :

?? Un PDA, orienté vers un mur, absorbe l'information qui y est projetée. Celle-ci est transformée afin d'en conserver la lisibilité sur le petit écran.
?? Le touriste passant à proximité d'un site historique est informé des événements pertinents [Cheverst 01].
?? Chez soi, le PDA sert de télécommande universelle qui s'adapte automatiquement à l'objet le plus proche mais ne fonctionne pas lorsqu'il est actionné par un jeune enfant : dans le salon, il reçoit la description du fonctionnement de la platine DVD et permet de la contrôler. Il en va de même pour la lumière et le chauffage. Dans la cuisine, des recettes originales sont suggérées en fonction du contenu du réfrigérateur et du nombre de personnes vivant au domicile.
?? En réunion, la sonnerie du téléphone passe automatiquement en mode vibration ou au contraire utilise une sonnerie repérable dans un milieu bruyant.

En résumé, tous nos exemples démontrent la nécessité pour le système d'identifier la situation afin de proposer un service adapté, qu'il s'agisse de renseigner l'utilisateur à

bon escient, de déléguer au système les tâches subalternes, de migrer sans discontinuité entre dispositifs d'interaction. L'objectif général vise une meilleure efficacité interactionnelle, une amélioration de la disponibilité du système, un élargissement de l'éventail des choix, en bref un progrès dans l'utilisabilité des systèmes. Nos objectifs s'inscrivent dans cette ligne de pensée.

3 Les objectifs

Dans le domaine de l'Interaction Homme-Machine, une analyse, un modèle et plus généralement tout outil pour la pensée, adoptent un point de vue soit centré sur l'utilisateur, soit centré sur le système. Dans cette étude, je m'intéresse aux aspects logiciels de la notion de contexte. Ceci posé, je précise les objectifs, leur cadre d'application et les limites actuelles de l'étude.

3.1 Couverture des objectifs

L'objectif de cette étude comprend deux volets complémentaires. L'un sert l'étape de conception tandis que l'autre intervient dans la mise en œuvre technique de systèmes sensibles au contexte :

1. *Définition de la notion de contexte sous forme d'un cadre conceptuel*. Si l'on veut exploiter le contexte de manière pertinente, il convient d'en comprendre la couverture. Le cadre conceptuel doit permettre au concepteur de systèmes sensibles au contexte d'entrevoir les dimensions du contexte, de se poser les bonnes questions, et d'en extraire les éléments pertinents pour le cas particulier du système à développer. Par exemple, pour un système de visite guidée, la localisation est une dimension essentielle.

2. *Définition d'un modèle et mécanisme opérationnels* qui facilitent la mise en œuvre de systèmes sensibles au contexte et qui respectent le principe de séparation fonctionnelle. En Interaction Homme-Machine, la séparation fonctionnelle est appliquée de la manière suivante : le Noyau Fonctionnel (ou l'application) est un composant distinct de l'Interface Homme-Machine (IHM). A son tour, l'IHM se décompose en Contrôle du Dialogue et Présentation. Le premier est chargé de l'enchaînement des tâches utilisateur (par exemple, la tâche « ouvrir document » doit être effectuée avant d'entreprendre les tâches d'édition). Le second est constitué d'interacteurs (plus communément appelés widgets) qui représentent les concepts applicatifs auprès de l'utilisateur. Le principe de séparation fonctionnelle permet de modifier l'un des composants tout en minimisant les effets de bord sur les autres. Ainsi, changer d'interacteur n'a pas ou peu d'impact sur le Contrôle de Dialogue et le Noyau Fonctionnel. Nous inspirant du savoir-faire en architecture logicielle des systèmes interactifs, nos abstraction et mécanisme logiciels doivent permettre de rendre sensibles au contexte des systèmes qui ne le sont pas, ou de faire évoluer un système sensible au contexte en limitant les modifications à un seul composant.

Les propositions conceptuelles et techniques envisagées se veulent générales mais devront servir en priorité la Plasticité des Interfaces Homme-Machine (IHM), un

problème initié par l'équipe Ingénierie de l'Interaction Homme-Machine du laboratoire CLIPS en 1998.

3.2 Application de l'étude : la plasticité des IHM

La plasticité d'une Interface Homme-Machine dénote sa capacité à s'adapter au contexte d'utilisation dans le respect de son utilisabilité [Thevenin 99]. Dans cette définition :

?? Le contexte est un couple "plate-forme / environnement" où la plate-forme est le support matériel et logiciel sous-tendant l'interaction. Par exemple, un PalmPilot ou un téléphone portable. La taille de l'écran, les dispositifs d'interaction, les capacités de calcul et de communication doivent être modélisés dans le système afin qu'il soit capable de s'adapter à leur variabilité. L'environnement se réfère à l'environnement physique accueillant l'interaction. Il est décrit par un ensemble d'informations, périphériques à la tâche en cours, mais susceptibles de l'influencer. Par exemple, la luminosité, le bruit, la localisation géographique.

?? L'adaptation est une réaction au changement de contexte. Comme le montre la figure 1, elle peut consister en un remodelage de l'interface ou en l'exécution d'une tâche (déclencher le chauffage à l'approche de l'occupant ; masquer les informations confidentielles à l'entrée d'une personne dans une pièce). L'adaptation est faite pour le bien-être de l'utilisateur, mais elle ne cible pas l'utilisateur. L'utilisateur est un utilisateur type défini dans le cahier des charges.

?? L'utilisabilité est évaluée sur la base de propriétés énoncées dans le cahier des charges.

En bref, la plasticité d'une interface est une adaptation résolument ancrée sur la variation des conditions physiques. Elle ne couvre donc pas l'adaptation aux changements d'état mental de l'utilisateur.

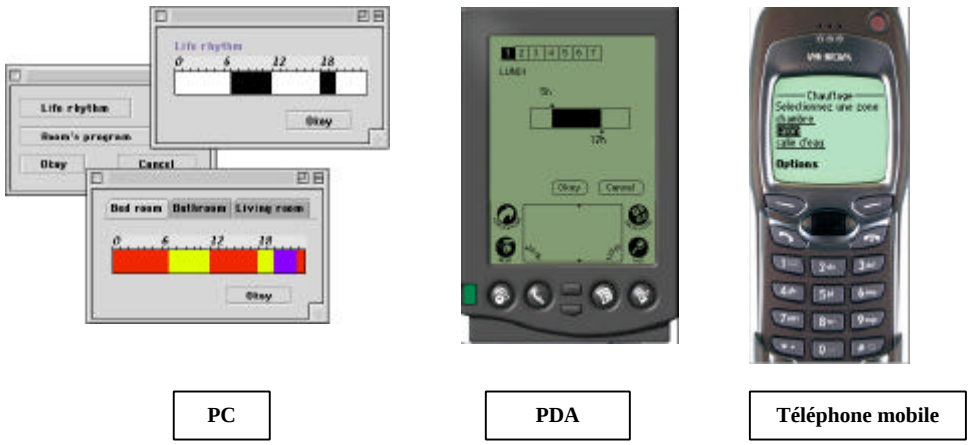

Figure 1 : Exemple d'interface utilisateur plastique au regard du dispositif d'interaction. Ici, l'IHM sert à contrôler le confort du domicile [Thevenin 99]. Cette IHM plastique a été produite selon le processus de la Figure 2. Elle est sensible aux changements de plate-forme mais pas aux changements de l'environnement.

Cette étude de DEA s'inscrit dans le processus de développement d'IHM plastique. La figure 2 en montre le principe. Le processus réifie (concrétise) par étape les modèles initiaux spécifiés par le concepteur (modèle des concepts, tâches, interacteurs, plate-forme, etc.) en modèles transitoires (interface abstraite, interface concrète) jusqu'à l'obtention de l'interface exécutable sensible au contexte [Thevenin 99, Calvary 01]. Les modèles initiaux peuvent être référencés (entièrement ou partiellement) à n'importe quel étape du processus de réification.

L'IHM plastique de la Figure 1 a été produite automatiquement par ARTStudio, un outil qui implémente le processus de la Figure 2 [Thevenin 99, Calvary 01]. Mais cette IHM n'est plastique qu'au regard de la plate-forme. En outre, l'utilisateur ne peut changer de plate-forme à la volée. C'est que ARTstudio, dans sa version actuelle, n'a pas incorporé le modèle d'évolution (voir figure 2). Le modèle d'évolution spécifie les réactions à mettre en œuvre en cas de changement de contexte. Il contient un ensemble de règles de type "Condition? Réaction" où :

?? La Condition exprime la détection et la reconnaissance d'un changement de contexte,
?? La Réaction dénote les mesures à mettre en œuvre pour préserver l'utilisabilité dans la nouvelle situation. La réaction comprend trois étapes : un prologue, la réaction proprement dite et l'épilogue.

Les modèles d'évolution, de plate-forme et d'environnement, spécifiés en amont par le concepteur, alimentent des modèles et mécanismes d'exécution nécessaires à l'adaptation dynamique. La figure 3 montre le modèle et mécanisme exécutable envisagés par [Calvary 01b] mais non encore réalisés. Ce « run time » inclut un superviseur de contexte qui, à partir des modèles exécutables résultant des spécifications amont, applique la réaction qui s'impose. Ce superviseur est sollicité à chaque changement de contexte par un mécanisme d'événement. Un événement est généré par une sonde, sorte d'instrumentation automatique du code applicatif.

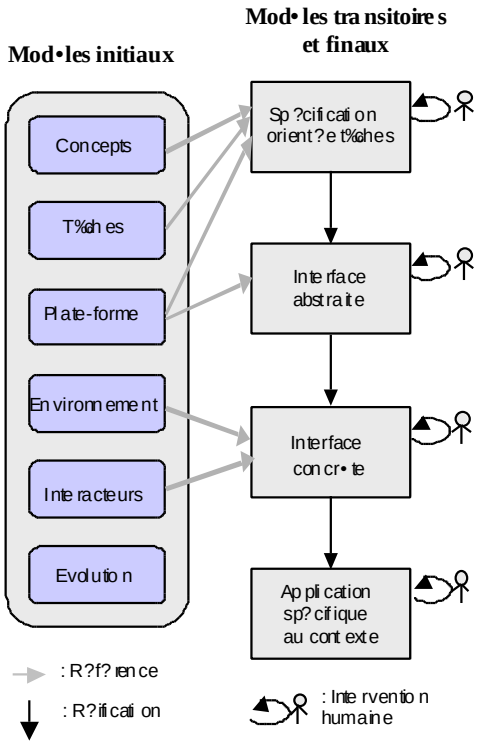

Figure 2 : Processus de développement pour la génération d'interfaces plastiques.

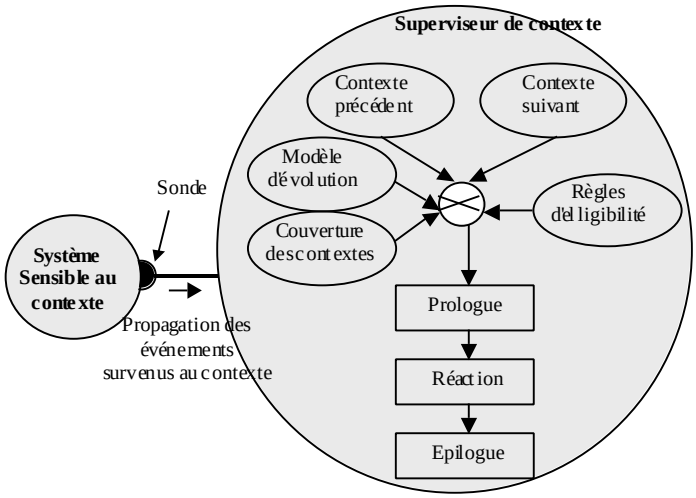

Figure 3 : Modèle et mécanisme pour l'adaptation dynamique aux changements de contexte.

Les retombées de cette étude de DEA doivent permettre de compléter l'outillage conceptuel et logiciel nécessaires aux IHM plastiques. En particulier, je m'intéresse en priorité à la modélisation de l'environnement physique et ceci à deux niveaux :

?? Au niveau de la conception. Il s'agit de définir les attributs de l'environnement que le concepteur doit considérer et spécifier dans le modèle de l'environnement.

?? Au niveau de l'exécution (le run-time). Il s'agit de fournir les abstractions et les mécanismes logiciels idoines permettant la détection et la reconnaissance de changements dans l'environnement. Ces détection et reconnaissance viendront alimenter la sonde qui, à son tour, activera le superviseur de contexte.

3.3 Limites et hypothèses de l'étude

Cette étude se borne à offrir un espace de conception et un support à la mise en œuvre de systèmes sensibles à l'environnement physique. Bien qu'importants, les points suivants ne seront pas traités dans le temps imparti au projet : l'éthique, l'évaluation des effets de l'adaptation, les erreurs d'évaluation de la situation par le système.

Ethique. La sensibilité au contexte peut s'appuyer sur la capture du comportement de l'utilisateur. On envisage par exemple, des systèmes de télésurveillance des personnes âgées pour qu'elles puissent rester à leur domicile plutôt que rejoindre une maison de retraite (exemple : le projet IMAG RESID-HIS). A l'évidence, ce type de système fragilise la protection de l'espace privé.

L'évaluation des effets de l'adaptation. L'adaptation automatique n'est pas toujours bonne à prendre.

?? L'adaptation peut avoir un effet disruptif. La propriété de prévisibilité [Gram 96] préconisée en IHM reste valable. Ou encore, si le comportement doit changer radicalement pour répondre à la situation, il faudra d'imaginer des IHM de transition qui permettront à l'utilisateur d'évaluer progressivement l'évolution.

?? L'adaptation sous forme de migration de tâche n'est pas toujours souhaitable : l'utilisateur a le sentiment de ne pas contrôler la situation ou bien perd progressivement son expertise (cas des pilotes d'avion). L'analyse du système GUIDE [Cheverst 01], un guide touristique pour la ville de Lancaster, est à ce titre intéressante. Dans une première version du système, les utilisateurs n'étaient renseignés que sur les monuments proches, interdisant ainsi l'accès à d'autres informations sur la ville. En voulant simplifier la tâche de l'utilisateur (celui-ci n'avait rien à faire sauf se déplacer), les concepteurs ont créé un système beaucoup trop pré-emptif.

Les erreurs d'évaluation de la situation peuvent conduire l'utilisateur à douter du bon fonctionnement du système. Pour ces systèmes où la sécurité des personnes est concernée, ce problème est fondamental. Mon modèle devra prendre en compte la notion d'incertitude, mais je n'aurai pas poussé la validation par l'expérience à partir de capteurs instables, imprécis, et non fiables. Ces problèmes de recherche qui relèvent d'un autre métier, sont supposés traités par d'autres équipes.

4 Méthode et approche de recherche

L'objectif pris dans sa généralité est ambitieux. Afin d'en cerner un contour réaliste :

?? J'ai retenu un cadre applicatif : la plasticité des IHM. Ce cadre reste néanmoins complexe. J'ai donc restreint l'étude au cas de l'environnement physique complétant ainsi les études menées actuellement dans mon équipe d'accueil.

?? J'ai fixé des limites à la couverture de l'étude : je couvrirai les modèles de l'environnement liés aux étapes de conception et d'exécution, mais dans le temps imparti à l'étude, l'évaluation des résultats sera sacrifiée sur trois points : validité éthique des outils, perception de l'adaptation par les utilisateurs, fiabilité des composants de bas niveau d'abstraction.

?? J'ai retenu une modélisation par objet m'en tenant aux approches usuelles du Génie Logiciel familière à la culture scientifique de l'équipe d'accueil. En particulier, on reportera à plus tard une analyse comparative de l'approche retenue avec une mise en œuvre par agents cognitifs au sens de l'Intelligence Artificielle.

Le cadre et les limites étant fixés, l'approche étant identifiée (de la modélisation à l'exécution par objets réactifs), j'étais en mesure d'explorer l'état de l'art :

?? Sur la notion de contexte et leurs applications
?? Sur les solutions logicielles.

Renseigné sur les éléments exploitables de l'état de l'art, j'ai élaboré deux modèles (au niveau conceptuel et au niveau logiciel) validés par plusieurs maquettes :

?? Un évaluateur du niveau d'activité des utilisateurs sur leur station de travail,
?? Un compteur de tasses de café qui envoie un courrier électronique d'avertissement aux membres du laboratoire afin qu'ils pensent à ramener les tasses oubliées dans leur bureau.

5 Organisation du rapport

L'organisation du rapport suit la logique des objectifs :

?? Dans un premier chapitre, nous étudions la notion de contexte en deux volets : une analyse de la littérature sur le sujet suivie de ma propre définition. Conformément aux objectifs, l'étude de l'état de l'art est menée en vue de définir un modèle qui serve la conception et la mise en œuvre de systèmes interactifs.

?? Au chapitre 2, nous étudions les dispositifs faisant le lien entre le monde physique dans lequel évolue l'utilisateur et la représentation numérique de ce monde du point de vue du contexte. Nous analysons, dans un premier temps, les offres relevées dans la littérature proche des préoccupations de l'Interaction Homme-Machine. Puis nous présenterons notre contribution avec la notion centrale de contexteur.

?? Dans un dernier chapitre, nous voyons la dimension technique du modèle conceptuel du contexteur vue dans le chapitre précédent. Ce chapitre se termine avec la description de maquettes d'application sensibles au contexte démontrant la faisabilité de l'approche

Chapitre 1 : Notion de Contexte

Si la notion de contexte est reconnue comme un facteur d'amélioration de la qualité des systèmes interactifs, il n'existe pas en ce jour de consensus sur une définition claire, précise et conduisant à une exploitation opérationnelle. Il faut dire qu'une analyse sérieuse de cette notion est récente. Pour s'en convaincre, je rapporte dans la section 1, les définitions proposées par les concepteurs et réalisateurs de systèmes interactifs sensibles au contexte. En 2, je proposerai une analyse critique avant d'énoncer, en section 3, ma propre définition. Cette proposition introduit la notion de *situation* qui correspond à un état instantané et qui, capitalisée au cours du temps, constitue le *contexte*. Si les objectifs prédisposent à opter pour le point de vue système, je montre en 4, une classification de contexte qui fait le pont avec le point de vue de l'utilisateur. Cette classification permet de donner en section 5 une facture formelle à ma définition de la notion de contexte. Ce chapitre se termine avec des exemples de systèmes sensibles au contexte et analysés selon les termes de cette définition.

1 Définitions du contexte : propositions de la littérature

Cette présentation des définitions tirées de la littérature est organisée selon l'ordre chronologique, démontrant une progression dans la compréhension de la notion de contexte.

Schilit et Theimer introduisent l'expression « context aware » en définissant le contexte comme la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets [Schilit 94, Schilit 94a]. Pour Schilit, étudier le contexte c'est répondre aux *questions quintiliennes* « Où es-tu ? », « Avec qui es-tu ? », « De quelles ressources disposes-tu à proximité ? ». Il définit donc le contexte comme les changements de l'environnement physique, utilisateur et computationnel. Ces idées seront reprises par Pascoe [Pascoe 98] puis par Dey [Dey 99].

Un peu plus tard, Brown restreint le contexte aux éléments de l'environnement de l'utilisateur que le *système informatique connaît* [Brown 96]. Puis, il préfère à sa première définition le fait que le contexte est plutôt « la localisation et l'identité des gens qui entourent l'utilisateur ainsi que l'heure, la saison, la température ... » [Brown 97].

Parallèlement aux travaux de Brown, des définitions émergent avec l'introduction explicite du *temps* et la notion *d'état*. Ryan définit le contexte comme l'environnement, l'identité et la localisation de l'utilisateur ainsi que le temps [Ryan 97]. Ward voit le contexte comme les états des environnements possibles de l'application [Ward 97]. Ces définitions, bien qu'intéressantes avec les notions de temps et d'état, restent trop vagues pour servir efficacement la conception et la mise en œuvre de systèmes interactifs.

Avec la définition de Pascoe en 1998, on retrouve la notion d'état à laquelle s'ajoute celle de *pertinence* : le contexte est « un sous-ensemble des états physiques et conceptuels ayant un intérêt pour une entité particulière » [Pascoe 98]. Puis Dey précise la notion *d'entité* : « le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes» [Dey 99]. En rapport avec l'étude de

la plasticité des IHM, Thevenin et al. aboutissent à une définition assez proche avec la notion de *contexte d'interaction*. Le contexte d'interaction est le couple « plate-forme-environnement » où l'environnement est l'ensemble des entités (objets, personnes et événements) périphériques à la tâche courante mais qui peuvent avoir un impact sur le comportement du système ou de l'utilisateur » [Thevenin 99]. On y trouve l'idée de tâche courante, de périphérie, d'entité, de pertinence et d'impact.

Au-delà de ces définitions, on relève des synonymes au mot contexte tels que *environnement* ou *situation* sans expliquer plus avant ce qu'ils recouvrent, esquivant ainsi subtilement la question.

Au bilan, soit le terme « contexte » est utilisé de manière intuitive ou superficielle, soit des efforts de définition ont été accomplis sans toutefois couvrir des aspects importants pour la conception et la mise en œuvre de systèmes interactifs. Nous poussons plus avant cette dernière remarque dans la section qui suit.

2 Analyse des définitions de la littérature

Les définitions relevées dans la littérature font référence à des éléments intéressants : connaissance par le système, état, temps, entité pertinente et tâche. Aucune d'entre elles ne couvre cependant tous ces aspects.

?? A l'exception de la proposition de Thevenin, aucune définition ne fait référence à la *tâche* utilisateur. Or, l'utilisation d'un système informatique, que ce soit de manière explicite (par exemple, rédiger un article avec Word) ou de manière implicite (cas d'une personne télésurveillée) a lieu en relation avec une activité humaine. Les *entités pertinentes* auxquelles il est fait référence dans certaines définitions ne peuvent se définir qu'en relation avec la tâche.

?? Le *temps* est un facteur décisif et se trouve fortement lié à celui d'*état*. L'état permet de caractériser ce qui est pertinent à un instant donné, mais l'évolution de cet état au cours du temps constitue à son tour une information qui peut être influente. Aucune des définitions ci-dessus ne fait référence à la capitalisation des états au cours du temps.

?? La plupart des définitions ne considère pas la plate-forme d'interaction comme une entité du contexte. Et pourtant, la variabilité des capacités de calcul, de mémorisation, de communication, et la nature des dispositifs d'interaction (absence de clavier ou d'écran, ordinateur évanescent) ont un impact sur la nature de l'interaction aussi bien du point de vue du système que du point de vue de l'utilisateur.

?? Aucun auteur n'a proposé jusqu'ici de formalisme simple pour représenter la notion de contexte et faciliter ainsi le raisonnement des concepteurs de système.

?? Mais tous les auteurs font référence à la localisation, à l'environnement physique démontrant leur statut central dans la notion de contexte.

Cette analyse justifie les éléments de la proposition ci-dessous.

3 Définition du contexte : une nouvelle proposition

3.1 Définition

Rappelons ici les définitions informelles que j'ai formulées dans l'Introduction de ce rapport :

?? Le contexte est l'ensemble des circonstances dans lesquelles se situe une action, ou qui entourent un fait et permettent de le comprendre.

?? Un système interactif est sensible au contexte lorsqu'il est capable d'identifier les circonstances qui entourent l'action (et notamment celle de l'utilisateur) en vue d'offrir un service adapté.

De manière plus formelle, le contexte se définit comme suit :

> Etant donné un utilisateur U, une tâche T et deux instants d'observation t_0 et t, le contexte à l'instant t est le cumul des situations entre les instants t_0 et t pour la réalisation de la tâche T par l'utilisateur U.

Ainsi le contexte n'est pas universel mais concerne un utilisateur donné U impliqué dans une tâche donnée T à un instant t. La notion de *situation* traduit les « circonstances qui entourent l'action » à l'instant t qui peuvent avoir un impact sur la conduite de la tâche T réalisée par U. Le *cumul de situations* traduit la capitalisation des situations au cours du temps à partir d'une *date d'observation de référence* t_0.

Le choix de t_0 est laissé à l'appréciation du concepteur. Il en va de même pour le choix de la granularité de la tâche T, de l'épaisseur du temps, du nombre d'utilisateurs du système :

?? t_0 peut dénoter la date de lancement de l'application, la date du début de la tâche T, une nouvelle journée, ou d'autres événements jugés pertinents pour le système en question.

?? La granularité de la tâche se mesure par sa position dans l'arbre de tâches. On rappelle qu'en Interaction Homme-Machine, une tâche se définit par un couple « but – procédure » où le but définit l'état souhaité et la procédure désigne le ou les plans permettant d'atteindre le but. La granularité d'une tâche se mesure par sa position dans l'arbre. Ma définition du contexte se rapporte à une tâche donnée mais la granularité de cette tâche dépendra des besoins. Si le système doit être sensible au contexte pour toutes les tâches accomplies avec le système, alors T dénotera le sommet de l'arbre.

?? L'épaisseur du temps définit la granularité des intervalles de temps entre deux observations successives de situation. Il dépend de la nature des entités observées, des objectifs qualité attendus, de la tâche T. Par exemple, une observation de l'ordre de la minute est raisonnable s'il s'agit de suivre l'évolution de la lumière pour un

suivi de doigt, mais une observation tous les quarts d'heure est suffisante pour contrôler l'éclairage d'une pièce d'habitation.

?? Le contexte, tel que je le définis, est centré sur un utilisateur donné U. La modélisation est applicable pour chaque utilisateur U du système.

Situation et cumul de situation sont détaillés ci-dessous.

3.2 La notion de situation

En philosophie, la situation est « l'ensemble des relations concrètes qui, à un moment donné, unissent un sujet ou un groupe au milieu et aux circonstances dans lesquels il doit vivre et agir » [Petit Robert]. Autrement dit, on retrouve dans « les circonstances », la notion d'entité pertinente identifiée dans l'analyse précédente, de même, l'idée d'instantané (« à un moment donné ») mais aussi de futur (le sujet ou le groupe en question doit pouvoir « vivre »).

De manière plus formelle, une situation se définit comme suit :

> Etant donné un utilisateur U, une tâche T et un instant t, la situation à l'instant t est l'ensemble des états des variables pouvant influencer T mais périphériques à T, à l'instant t et/ou ultérieurement, et de l'ensemble des relations entre ces variables en t.

Dans cette définition, nous relevons les notions de variable périphérique, d'influence et de relation. Nous les détaillons ci-après.

3.2.1 Variable périphérique de tâche

Les variables périphériques d'une tâche à l'instant t modélisent des entités non centrales à la tâche en t mais susceptibles de l'influencer.

La distinction entre *périphérique* et *central* est assurée dans l'étape amont du processus de conception. Comme nous le disions dans le chapitre d'Introduction, les techniques de type « Contextual Design » ont notamment pour mission d'identifier les entités pertinentes et, parmi ces entités, de distinguer celles qui sont manipulées au cours de la tâche (les entités centrales) des entités qui sont potentiellement influentes (les entités périphériques).

Pour reprendre l'exemple de la billetterie électronique, les notions de date, de place et de tarifs sont centrales pour la tâche de réservation, tandis que le lieu (hall de gare ou chez soi) est périphérique. Néanmoins, le fait que ce lieu puisse être public ou privé présente un impact potentiel, maintenant ou plus tard. (Suite à une mauvaise expérience en un lieu, la tâche sera peut être effectuée autrement la prochaine fois.)

Notons qu'une entité non centrale en t, par exemple, un collègue dans la pièce voisine, peut le devenir en t+?t s'il vient vous rendre visite et s'approche trop près de votre écran (car vous tenez à conserver un caractère privé au texte que vous êtes en train d'écrire).

Afin de faciliter l'analyse d'une situation donnée, je propose d'organiser les variables selon l'espace de la figure 1.1. Dans cet espace, on distingue quatre axes d'analyse :

?? l'environnement physique
?? l'environnement social
?? l'environnement système (ou plate-forme d'exécution)
?? l'environnement utilisateur

L'environnement physique comprend l'ensemble des lieux de l'interaction ainsi que l'ensemble des lieux où l'utilisateur U peut potentiellement intervenir.

L'environnement social se réfère aux personnes présentes dans l'environnement physique.

La plate-forme décrit l'ensemble des dispositifs et systèmes computationnels ainsi que des ressources réseaux et systèmes disponibles.

L'environnement utilisateur décrit le (ou les) utilisateur(s) U du système.

Pour illustrer les différents environnement, prenons l'exemple d'un bureau.

?? Le bureau est le lieu d'interaction il est par conséquent un élément de l'environnement physique. Il possède des propriétés qui sont sa taille, sa localisation … De plus, il contient un grand nombres d'objets, comme des stylos, chaises, tables …, qui sont eux aussi des éléments de l'environnement physique. De même, chacun de ces objets possèdent des propriétés qui lui sont propres (couleurs, taille …)
?? L'environnement utilisateur comprend les informations concernant le « propriétaire » du bureau, comme son agenda, sa fonction, ses goûts, ses habitudes … Si le bureau est partagé par deux personnes, l'environnement utilisateur comprend les informations sur les deux utilisateurs.
?? Les autres personnes pouvant venir occasionnellement dans le bureau voit leurs profiles (informations les concernant) s'ajouter à l'environnement social de cette situation durant leurs séjours dans le bureau.
?? L'environnement système est composé des ordinateurs des utilisateurs et autres personnes se trouvant dans le bureau. Les différentes machines possèdent des propriétés comme la taille du disque dur, la mémoire disponible, la vitesse du processeurs ou encore la bande passante du réseaux.

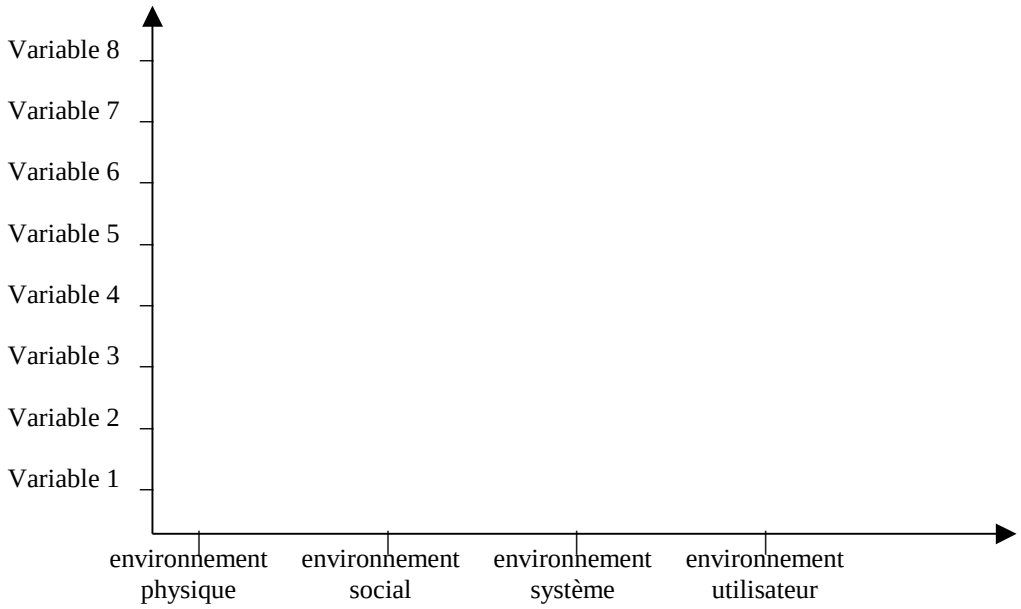

Figure 1. 1 : Espace conceptuel des variables de situation.

3.2.2 Impact de variable périphérique

L'impact d'une variable périphérique sur la tâche en cours peut être :

?? Un abandon définitif de la tâche.

?? Une interruption de la tâche suivie d'une reprise.

?? Une poursuite de la tâche sans interruption, mais avec altération (en mieux ou en moins bien) des propriétés nominales attendues. Par exemple, le temps de réalisation de la tâche, nombre d'erreurs commises, ou affectation de la qualité du produit de la tâche (par exemple, en situation de stress en gare, on ne cherchera pas à obtenir notre place préférée).

3.2.3 Relations entre variables

Les relations entre les variables ont pour but d'apporter des informations complémentaires sur les variables ainsi que de permettre aux concepteurs d'application de mieux cerner les variables qu'ils doivent observer.

On distingue les relations suivantes :

?? *Héritage*. Une variable hérite d'une ou plusieurs autres variables si l'information qu'elle apporte peut être obtenue par une combinaison des autres variables.

?? *Equivalence*. Deux variables sont équivalentes si elles contiennent la même information.

Prenons deux exemples simples pour illustrer ses relations.

?? Dans le premier exemple on s'intéresse seulement à quatre variables du contexte. La première (V_1) nous indique le poids que supporte un certaine chaise, la deuxième (V_2) nous donne la température de la chaise, la troisième (V_3) la température de la salle ou se trouve la chaise et la dernière (V_4) nous indique s'il y a une personne sur la chaise. Si on estime que dire qu'une personne est sur la chaise c'est dire que la chaise supporte un poids conséquent et que sa température est supérieure à celle de la salle. Alors, la dernière variable (V_4) hérite des trois autres variables (V_1, V_2 et V_3).

?? Dans notre deuxième exemple, on s'intéresse à trois variables représentant des températures, à un même temps t, dans des unités différentes. En degrés kelvin pour V_1, en degrés Celsius pour V_2 et en degrés fahrenheit pour V_3 . Puisqu'elles représentent la même information elles sont équivalentes.

3.3 Cumul de situation

Rappelons brièvement qu'une situation S_i est un ensemble composé des états (c'est-à-dire des valeurs) de variables V_j observées au temps t_i. Ainsi, chaque variable V_j peut être observée dans une ou plusieurs situations et peut avoir une nouvelle valeur pour chacune de ces observations.

Le cumul de situation est composé de deux opérations appliquées en séquence : la fusion des ensembles de variable puis la modification des relations entre les variables.

Etape 1 : fusion des ensembles de variables. Pour réaliser cette opération, pour toute situation impliquée dans la fusion, chaque variable est étiquetée par la date t de la situation. Ainsi, chaque variable devient un couple <variable, t>. Le schéma de la figure 1.2 illustre l'effet de l'opération. Dans cet exemple, il s'agit de cumuler deux situations S_i et S_j observées respectivement aux instants t_i et t_j. S_i comprend trois variables : V_1, V_2, V_3 et S_j est constituée des variables V_1, V_2, V_5, V_7. L'ensemble obtenu par fusion inclut V_1 observée en t_i et V_1 observée en t_j, etc. On notera que les relations entres les variables d'une situation sont inchangées. Celles-ci font l'objet de l'étape 2.

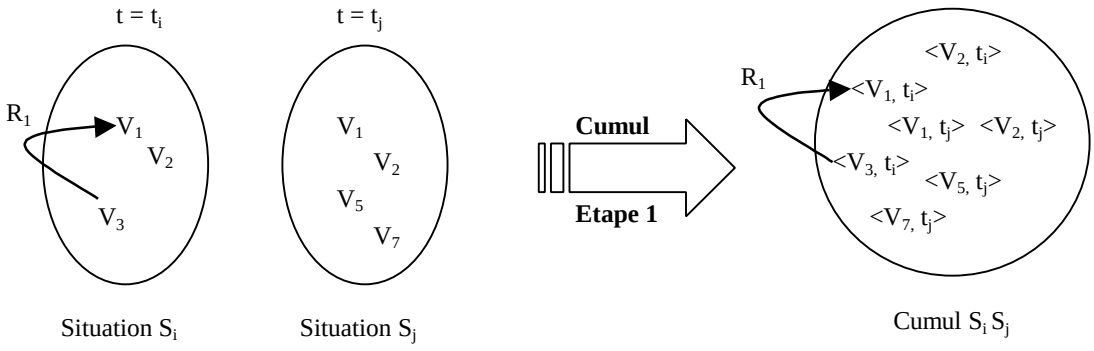

Figure 1. 2: Représentation ensembliste de la première étape de l'opération de cumul.

Etape 2 : modification des relations entre variables. Une modification est soit la suppression d'une relation, soit l'ajout d'une relation, soit les deux. Dans l'exemple de la figure 1.3, la relation R_1 entre V_3 et V_1 qui faisait sens dans la situation S_i, disparaît dans l'opération de cumul mais une nouvelle relation R'_1 apparaît entre V_3 observée en t_i et V_7 observé en t_j.

Prenons un exemple concret illustrant le cas de la figure 1.3. Les variables V_1 et V_3 représente la température d'une pièce durant la même situation. La variable V_3 correspondant à une température en degrés Celsius directement issu d'un thermomètre alors que la variable V_1 est la même température mais en degrés fahrenheit. V_1 est obtenue en convertissant V_3. Donc la relation R_1 entre V_3 et V_1 est à la fois une relation d'héritage et une relation d'équivalence. Durant la situation S_j, le thermomètre utilisé pour mesuré la valeur de V_3 tombe en panne. C'est pourquoi V_3 n'apparaît pas. Cependant, un autre instrument à garder en mémoire la valeur de V_3 durant la situation S_i. Cette valeur est représenté par la variable V_7 . Un nouveau thermomètre permet d'obtenir directement la valeur de la variable V_1. Lors de la deuxième étape de l'opération de cumul l'analyste du contexte ne désire pas faire apparaître les changement de thermomètre et il fait donc disparaître la relation R_1. Cependant note la relation R_1' pour signaler l'équivalence entre les variables V_3 et V_7

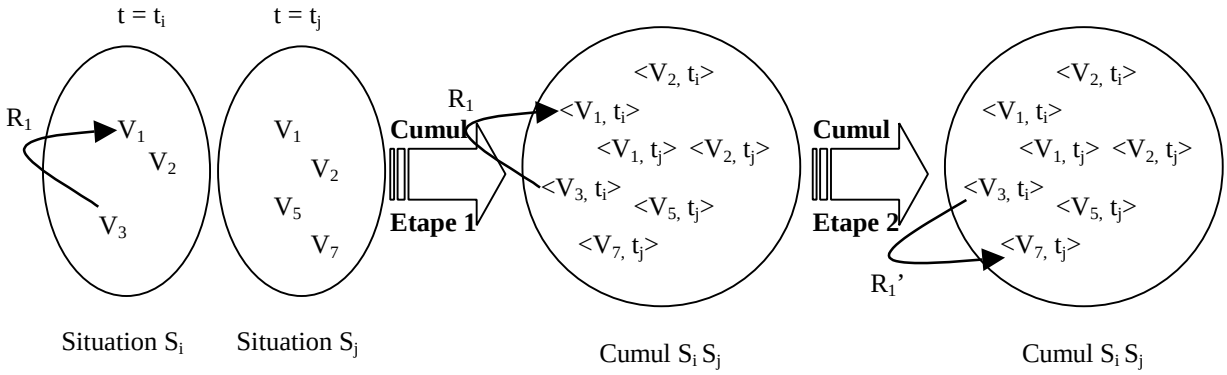

Figure 1. 3 : Représentation ensembliste des deux étapes de l'opération de cumul.

3.3.1 Définition du contexte : résumé

Désormais, par le mot « contexte » on entendra « contexte pour un utilisateur U accomplissant un tâche T à un instant t ». De plus, le « contexte pour U accomplissant T en t », ce n'est pas seulement les variables périphériques en cet instant mais leur évolution ou cumul depuis un instant de référence t_0. Les variables périphériques se répartissent en attributs de l'environnement physique, de la plate-forme logicielle et interactionnelle, de l'environnement social et de l'utilisateur U considéré. L'épaisseur du temps, la date de référence initiale, la granularité de la tâche, le nombre d'utilisateurs sont laissés à l'appréciation du concepteur.

Cette définition du contexte vaut selon plusieurs point de vue que l'on explicite dans la typologie qui suit.

4 Typologie du contexte

La typologie que je propose distingue trois points de vue (omniscient, système, utilisateur) et conformément aux objectifs annoncés, deux étapes essentielles du processus de développement d'un système : l'étape de conception/analyse et l'étape d'exécution.

4.1 Trois points de vue sur le contexte

On rappelle que le mot « contexte » est un raccourci de « contexte pour un utilisateur U accomplissant un tâche T à un instant t ».

Reprenons les trois points de vue annoncés : omniscient, système et utilisateur.

?? On dénomme *contexte brut*, tous les faits universels absolus englobant le réel et le possible en relation avec T et U à l'instant t.

?? Le *contexte système* est la modélisation du contexte par le système en relation avec T et U à l'instant t.

?? Le *contexte utilisateur* est la modélisation du contexte par l'utilisateur U en relation avec sa tâche T à l'instant t.

Le schéma de la figure 1.4 illustre les relations entre ces trois types de contexte (omniscient, système et utilisateur). Par définition, les contextes système et utilisateur sont inclus dans le contexte brut. Mais les contextes système et utilisateur ne se recouvrent pas nécessairement : l'utilisateur a un vécu (les variables qu'il modélise mentalement) que le système ignore (à tort ou à raison). Inversement le système dispose de variables que l'utilisateur ignore.

?? Le *contexte net* correspond à l'intersection des contextes système et utilisateur. Par opposition au contexte brut, le contexte net est le contexte dont on a déduit tout élément étranger et qui est utile au système pour satisfaire les attentes de l'utilisateur U (au regard de sa tâche T à l'instant t depuis t_0, rappelons-le).

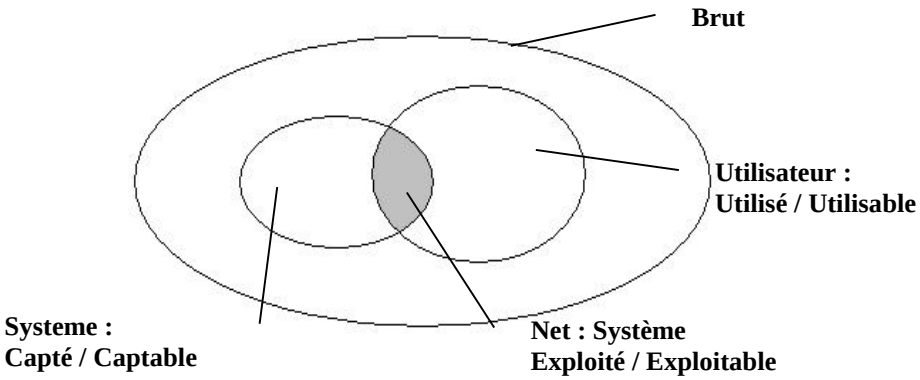

Figure 1. 4 : Relations ensemblistes des contextes Brut, Système, Utilisateur et Net.

4.2 Deux étapes de déploiement du contexte

A l'étape de conception, les analystes et concepteurs identifient les requis et les confrontent aux contraintes techniques et sociales de faisabilité, etc. Il en résulte un ensemble de « souhaitables ».

A l'exécution, les « effectifs » ne sont pas nécessairement conformes aux « souhaitables » : cas des pannes, cas de détournement de la technologie par les utilisateurs, etc. Pour cette raison, on distinguera :

?? Pour le contexte système : le *contexte captable* du *contexte capté*. Le contexte captable dénote le contexte système tel qu'il a été défini à l'étape d'analyse/conception pour la tâche T de U en t. Le contexte capté est le contexte tel qu'il est modélisé en pratique à l'exécution par le système : des capteurs ont pu tomber en panne.

?? Pour le contexte utilisateur : le *contexte utilisable* et le *contexte utilisé*. En analyse amont, les concepteurs ont défini le contexte utilisable : l'ensemble des facteurs (les variables périphériques) auxquels l'utilisateur risque de faire appel dans la conduite de sa tâche T. Dans les faits, ces facteurs n'ont pas eu d'impact sur la tâche utilisateur : contexte utilisé. Inversement, les concepteurs ont pu oublier des entités dans leur analyse qui se révèlent être des facteurs influents dans la conduite de la tâche.

?? A l'intersection des points de vue système et utilisateur : le *contexte net exploitable* et le *contexte net exploité* obtenus respectivement à l'intersection des contextes captable et utilisable, et des contextes capté et utilisé.

La distinction entre le *–able* et le *–é* permet de fournir des bases à l'évaluation de la conception du système comme à ses performances effectives. Pour l'heure, elle nous permet d'aller plus avant dans la formalisation de la notion de contexte.

5 Formalisation de la notion de Contexte

Notons :
?? *situation* ^{U,T} *(t)*, la situation rencontrée par l'utilisateur U, réalisant la tâche T, à l'instant t.
?? *contexte* ^{U,T} *(t)*, le contexte de l'utilisateur U, réalisant la tâche T, à l'instant t.

Indiçons par :
?? *B*, les situations et contextes Bruts,
?? *S*, les situations et contextes Systèmes,
?? *U*, les situations et contextes Utilisateurs,
?? *N*, les situations et contextes Nets.

Par définition, on a :
?? contexte $^{U,T}_{i}$ (0) = situation $^{U,T}_{i}$ (0) avec i ? [B,S,U,N]
?? ? t>0, contexte $^{U,T}_{i}$ (t) = situation $^{U,T}_{i}$ (t) ? contexte $^{U,T}_{i}$ (t-1) où ? est l'opérateur de cumul décrit précédemment et i ? [B,S,U,N]
?? ? t?0, situation $^{U,T}_{N}$ (t) = situation $^{U,T}_{U}$ (t) ? situation $^{U,T}_{S}$ (t)
?? ? t?0, situation $^{U,T}_{N}$ (t) = situation $^{U,T}_{S}$ (t) ? situation $^{U,T}_{U}$ (t) (commutativité de l'intersection)

Par commodité d'écriture, on notera :

$$? \; t?0, \text{contexte}^{U,T}_{B}(t) = \text{situation}^{U,T}_{B}(0) + \int_{0}^{t} ? \; \text{contexte}^{U,T}_{B}(x).dx$$

où, le signe intégrale correspond à une utilisation récursive de l'opération de cumul décrite ci dessus.

La notion de relation peut aussi être formalisée. De manière générale, on notera la relation R_j de la variable V_i avec les variables V_a ,…,V_z de la manière suivante :

$$V_i = R_j(V_a, \ldots, V_z)$$

On notera H_i les relations d'héritage et E_i les relations d'équivalence.

Alors :

$$V_i = E_k(V_j) \; ? \quad V_j = E_l(V_i)$$

Par exemple, si V_i représente une température en degrés Celsius et que V_j représente la même température en degrés Fahrenheit alors E_k est une fonction de conversion des degrés fahrenheit en degrés Celsius et E_l la fonction inverse.

Puisque les contexte Utilisateur et Système sont des visions restreintes du contexte Brut, l'utilisé / utilisable et le capté / captable peuvent se modéliser comme des filtres f_U et f_C appliqués aux valeurs brutes, à un instant t. Ainsi :

$$? \; t?0, \text{contexte}_{U,T,U}(t) = f_U(\text{contexte}_{U,T,B}(t), t)$$
$$? \; t?0, \text{contexte}_{U,T,C}(t) = f_C(\text{contexte}_{U,T,B}(t), t)$$

Les fonctions, filtres, f_U et f_C dépendent de l'application et de l'ensemble des capteurs disponibles.

Cette définition du contexte et sa typologie sont illustrées dans la section qui suit au moyen d'exemples représentatifs de l'état de l'art.

6 Illustration

Pour justifier ma définition du contexte et pour vous permettre de mieux appréhender les différents points de celle ci concernant les différentes catégories dans lesquelles peuvent être classé les « variables » du contexte, je vais vous présenter certaines applications que j'ai retenues pour illustrer l'ensemble des logiciels déjà existant. Par la suite, je vous montrai les points important du contexte traités par celles ci et enfin je vous présenterais une rapide taxonomie permettant de classer les différentes applications en fonctions des catégories du contextes qu'elles couvrent.

6.1 Contributions représentatives

ContextNotePad est une version sensible au contexte de l'application Notepad (bloc note, mémo …) sur Palm Pilot [Schmidt 00]. La figure 1.5 montre le ContextNotePad à l'état initial.

?? Le dispositif est sensible au mouvement de faible amplitude : il s'allume automatiquement lorsque l'utilisateur le prend dans la main et inversement, s'éteint lorsqu'il est reposé.

Figure 1. 5 : Etat initial de ContextNotePad.

?? Le dispositif est sensible au déplacement, par exemple lorsqu'il est tenu par un utilisateur en marche : la taille des caractères est augmentée en sorte que l'utilisateur puisse lire le texte malgré son déplacement (voir figure 1.6). Le système rétablit la taille normale de la police de caractères à l'arrêt du mouvement.

Figure 1. 6 : Le ContextNotePad grossit les caractères sur détection de mouvement de marche.

?? Si la lumière ambiante baisse de manière significative, l'écran est rétro-éclairé pour faciliter la lecture (voir figure 1.7).

Figure 1. 7 : Le ContextNotePad rétro-éclairé l'écran sur détection de lumière ambiante faible.

?? Si le dispositif n'est pas utilisé, l'écran est masqué sur détection de présence d'étrangers à proximité en sorte de conserver le caractère privé des informations affichées (voir figure 1.8).

Figure 1. 8 : le ContextNotePad en mode masqué.

In / Out Board se présente sous la forme d'un tableau qui indique, pour chaque utilisateur, sa présence ou absence de son bureau avec les heures d'arrivée ou de départ [Salber 99] (voir figure 1.9)

Figure 1. 9 : l'IHM du In / Out Board.

Information Display affiche les informations pertinentes sur la localisation de l'utilisateur [Salber 99]. Cet écran s'allume automatiquement quand quelqu'un approche pour qu'il puisse lire les informations.

DUMMBO Meeting Board est un tableau blanc physique instrumenté pour capturer son contenu ainsi que les informations audio-vidéo provenant de la réunion [Salber 99]. La capture de la réunion démarre automatiquement dès que deux personnes au moins se trouvent à proximité du tableau.

Context-Aware Office Assistant est, comme son nom l'indique, un assistant de secrétaire de bureau [Yan 00]. Il annonce les nouveaux arrivants, les fait patienter, leur propose d'autres rendez-vous en consultant l'agenda de l'arrivant.

Cyberguide, voir figure 1.10, fournit à l'utilisateur sa position et le représente sur une carte [Abowd98]. Il produit automatiquement des informations sur les lieux alentour et permet de réserver et de payer la place de parking. Cette place est indiquée au conducteur en tenant compte des places disponibles et de la position courante de l'utilisateur. Ce système, fondé sur la localisation, est à rapprocher de GUIDE [Cheverst 01] présenté dans notre chapitre d'introduction.

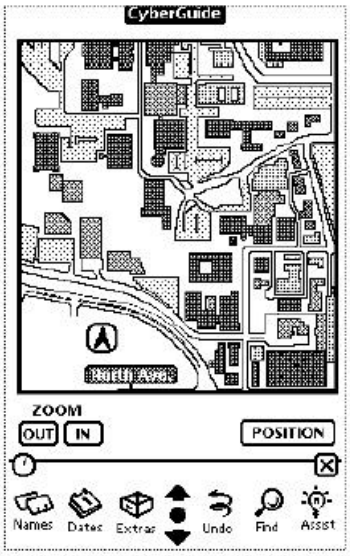

Figure 1. 10 : L'interface de Cyberguide lors de la localisation sur un plan de ville.

CyberDesk utilise les informations du contexte pour faciliter l'intégration d'autres services de bureau comme le mail et le web [Abowd 98]. Par exemple, s'il trouve des informations concernant la home page ou un « lien http » dans le mail que vous êtes entrain de lire, il l'ouvre automatiquement dans un votre navigateur préféré. L'ouverture du navigateur ce fait en tache de fond pour ne pas vous gêner. Si vous changer de mail sans aller voir votre navigateur, il considère que le lien de vous intéressez pas et le ferme.

ClassRoom2000, utilisée pour l'apprentissage de la lecture, capture sous forme audio-vidéo les interactions qui se passent à l'intérieur de la salle de lecture [Abowd 98].

Museum tour application vise à reconnaître la nationalité du visiteur pour lui présenter les informations concernant les œuvres exposées dans sa langue natale [Salber 01].

6.2 Classifications des systèmes sensibles au contexte

Puisque cette étude opte en priorité pour le point de vue système, considérons le contexte système et notamment le contexte qu'il est effectivement capable de traiter : le contexte capté. Dans la classification ci-dessous proposée, on distinguera :

?? La finalité du contexte capté,
?? La classe des variables constituant le contexte capté,
?? Les variables les plus souvent utilisées dans le contexte capté.

6.2.1 Finalité du contexte capté

L'analyse de l'état de l'art révèle trois types de finalité :

?? *Finalité informationnelle* : l'objectif est de fournir des informations situées.

?? *Finalité de présentation*: l'objectif est d'utiliser les informations contextuelles pour adapter l'Interface Homme-Machine automatiquement.
?? *Finalité de service* : l'objectif est de gérer automatiquement certains services.

Le tableau de la figure 1.11 illustre la classification au moyen des systèmes référencés plus haut.

Système	Information	Présentation	Service
ContextNotePad		X	
In / Out Board	X		
Information Display	X		
DUMMBO Meeting Board			X
Context-Aware Office Assistant			X
Cyberguide			X
CyberDesk			X
ClassRoom2000			X
Museum tour application			X

Figure 1. 11 : classification des applications existantes en fonction de leur type de fonctionnalités

Cette classification permet d'appréhender de manière simple et directe l'objectif du système au regard de la sensibilité au contexte.

6.2.2 Classe de variables du contexte capté

Selon ma définition, les classes des variables d'un contexte sont au nombre de quatre et correspondent à (cf. section 3.2.1) :
?? L'environnement physique,
?? L'environnement social,
?? L'environnement système (ou plate-forme),
?? L'environnement utilisateur.

Le tableau de la figure 1.12 permet de comparer à gros grain la couverture de sensibilité des systèmes étudiés. Comparée à la classification précédente, qui est centrée sur les objectifs généraux du système, le concepteur dispose d'une analyse plus précise.

Système	physique	social	système	utilisateur
ContextNotePad	X		X	X
In / Out Board				X
Information Display			X	X
DUMMBO Meeting Board	X			X
Context-Aware Office Assistant	X	X		X
Cyberguide	X			X
CyberDesk		X	X	
ClassRoom2000	X			X
Museum tour application	X	X		X

Figure 1. 12 : Classification de systèmes en fonction des types de variables modélisées dans le contexte capté.

La classification suivante va un cran plus loin en considérant les variables elles-mêmes.

6.2.3 Les variables du contexte capté

Dans le tableau de la figure 1.13, n'ont été retenues que les variables les plus fréquemment référencées dans les exemples de système étudiés. On relève :

?? Dans la classe « environnement utilisateur » : l'identité, la localisation, la présence, le mouvement de l'utilisateur,
?? Dans la classe « environnement social » : la présence de personnes,
?? Dans la classe « environnement physique » : la présence de sons et le niveau de luminosité.

Cette classification permet d'avoir une idée comparative de la richesse du contexte capté. Mais, en l'état, la proposition est loin d'être exhaustive. Il conviendra d'approfondir la nature de chaque variable. Par exemple, les variables « localisation » et « espace » méritent d'être approfondies à travers les 4 catégories d'environnement (utilisateur, société, physique, plate-forme et dispositif d'interaction) : espace privé ou public, espace fermé et espace ouvert, espace (architecture) malléable ou fixe, localisation relative à cet espace ou localisation absolue (de l'utilisateur, du dispositif d'interaction, de l'environnement social, de l'environnement physique).

Système	Identité (U)	Localisation (U)	Présence (U)	Présence (So)	Sons (P)
ContextNotePad				X	
In / Out Board			X		
Information Display			X	X	
DUMMBO Meeting Board			X	X	X
Context-Aware Office Assistant			X	X	
Cyberguide		X			
CyberDesk					
ClassRoom2000					X
Museum tour application	X	X	X	X	X

Figure 1. 13 : Classification des systèmes sensibles au contexte selon les variables les plus fréquemment considérées.

7 Résumé

Dans ce chapitre, nous avons vu une définition de la notion de contexte. Un contexte n'existe pas en tant que tel mais il convient de parler du *contexte de la tâche T réalisée par l'utilisateur U à l'instant t*. Par contexte de la tâche T réalisée par U à l'instant t, il faut entendre *le cumul des situations entre les instants t_0 et t intervenant dans la réalisation de T par U*. La situation à l'instant t est l'ensemble des variables périphériques à la tâche T mais susceptibles de l'influencer. Les variables périphériques se répartissent en attributs de l'environnement physique, de la plate-forme logicielle et interactionnelle, de l'environnement social et de l'utilisateur U considéré. L'épaisseur du temps, la date de référence initiale, la granularité de la tâche, le nombre d'utilisateurs sont laissés à l'appréciation du concepteur.

Cette définition du contexte vaut :

?? selon plusieurs point de vue qui nous amènent à distinguer le contexte brut, le contexte système, le contexte utilisateur et le contexte net (intersection des contextes système et utilisateur) ;
?? et selon les deux étapes essentielles du processus de développement d'un système : l'étape de conception/analyse et l'étape d'exécution, qui nous amènent à distinguer les contextes système captable et capté, les contextes utilisateur utilisables et utilisés et les contextes net exploitable et exploités.

Nous avons ensuite vu une définition formelle des éléments de cette typologie avant de clore le chapitre par une illustration des concepts introduits au moyen de quelques exemples représentatifs de systèmes sensibles au contexte.

Ayant posé les bases conceptuelles de la notion de contexte en général, dans la suite de ce rapport, nous nous intéressons au contexte système et notamment à la mise en œuvre de contexte capté avec la notion de *contexteur*.

Chapitre II : Contexteur : un modèle logiciel du contexte

Pour que les ordinateurs puissent élaborer un contexte capté, ils doivent acquérir les informations contextuelles relatives aux tâches auxquelles ils participent. Pour ce faire, ils ont besoin de dispositifs faisant le lien entre le monde physique dans lequel évolue l'utilisateur et la représentation numérique qu'ils se font d'un sous-ensemble du contexte brut.

Quel est la forme d'un tel dispositif ? Quels en sont les propriétés ? Quelle architecture ? etc. sont autant de questions essentielles que se posent aujourd'hui les concepteurs logiciels de système sensible au contexte.

Ce chapitre est structuré comme suit :

?? Dans une première partie de ce chapitre, nous analysons les offres relevées dans la littérature proche des préoccupations de l'Interaction Homme-Machine. Nous savons que la Robotique se heurte à des problèmes similaires mais dans le cadre et le temps imparti à cette étude, on a souhaité s'affranchir des bas niveaux d'abstraction traités en fusion de données. Pour les mêmes raisons (durée du projet), nous n'avons pas étudié l'apport des agents cognitifs, donnant la priorité à l'approche Génie Logiciel de l'Ingénierie de l'Interaction Homme-Machine.

?? Le reste du chapitre présente ma contribution avec la notion centrale de contexteur. Ce modèle sera ensuite illustré par la mise en œuvre au chapitre suivant.

1 Modèles logiciels du contexte : état de l'art

On relèvera trois études significatives sur les architectures logicielles traitant de capture du contexte : le Context-toolkit [Dey 99], l'architecture de Schmidt [Schmidt 99] ainsi que le Context-handling component de Salber [Salber 00]. Nous les passons successivement en revue en fournissant pour chacune de ces contributions une analyse de leur atouts et de leurs limites.

1.1 Le context-toolkit

1.1.1 Description

Le nom de cet outil, Context-toolkit, vient de l'analogie avec les bibliothèques graphiques de widgets. Comme pour les « toolkit » graphiques qui offrent un modèle computationnel et des widgets d'utilité publique (menu, formulaire, bouton), l'objectif du Contex-Toolkit est d'offrir un modèle d'exécution et des composants de base réutilisables.

Comme le montre la figure 2.1, le constituant de base de l'architecture est un « context-widget ». Deux autres types de composants complètent l'architecture : les « interpréteurs » et les « serveurs ».

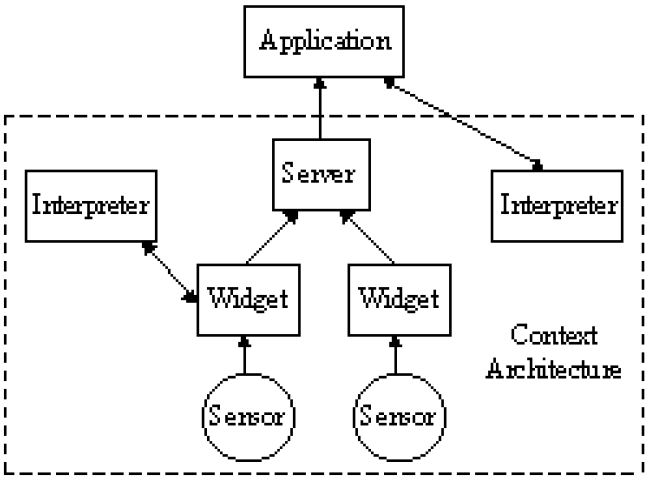

Figure 2. 1 : Architecture et composants du Context-toolkit

?? Un « context widget », ou widget de contexte, est un composant autonome qui encapsule un capteur physique. Son rôle est de communiquer à un (ou plusieurs) serveur(s) ou interpréteur(s) les informations perçues en fonction des données reçues par le capteur. Il a aussi pour rôle de constituer un historique qui lui est propre. Ce stockage de signaux peut être désactivé pour des fonctions qui n'en présentent pas l'intérêt.

?? Les interpréteurs ont pour objectif de donner une sémantique aux signaux qui leur sont transmis par les « contexts widgets » et ceci au bon niveau d'abstraction. Par exemple, un interpréteur de localisation de personne peut fournir comme informations :

- Personne A dans la salle B204
- Personne A dans le Bâtiment B de l'IMAG
- Personne A dans le Campus de Grenoble

?? Les serveurs ont pour pour objectif principal de collecter l'ensemble des signaux émis par les autres composants et de faire le lien entre les applications et les « contexts-widgets ». Les serveurs ont également la charge de synthétiser les informations en informations de niveau d'abstraction supérieur. Cependant le mécanisme de synthèse n'est pas décrit et revient au programmeur.

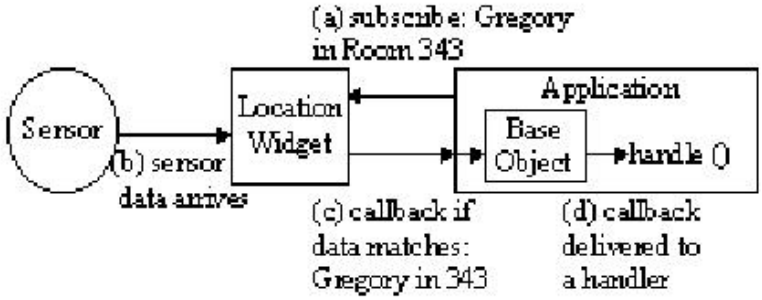

Figure 2. 2 : Exemple d'interaction entre une application et un widget de contexte.

Sur le plan technique, le système de communication entre les context-widgets et l'application se fait par souscriptions et la réaction aux souscriptions s'effectue sous forme de call-back (réaction). La figure 2.2. montre un exemple d'interaction entre l'application et un widget de contexte.

1.1.2 Atouts et handicaps

En première analyse, le modèle est clair et introduit la notion de niveau d'abstraction. Les context-widgets permettent un accès simple au capteur. Les interpréteurs donnent un sens au contexte capturé, et les serveurs font le lien entre les context-widgets et les applications sensibles au contexte.

Cependant, les context-widgets ne sont que de simple pilotes logiciels pour les capteurs auxquels ils sont liés. Ils ne fournissent pas de sens aux informations qu'ils transmettent et ont besoin pour cela des interpréteurs.

Les serveurs, quant à eux, regroupent le gros du travail. Ils doivent synthétiser de nouvelles informations avec celles que leur fournissent les context-widgets et faire le lien entre les applications et les context-widgets.

Au bilan,
?? Une architecture simple à comprendre,
?? Mais une architecture qui répartit mal les fonctions ;
?? Enfin, un modèle sans mesure de la qualité des informations transmises à l'application qui doit donc faire confiance ou pratiquer ces calculs elle-même.

1.2 L'architecture de Schmidt

1.2.1 Description

Comme le montre la figure 2.3, l'architecture de Schmidt s'appuie sur un modèle en couches.

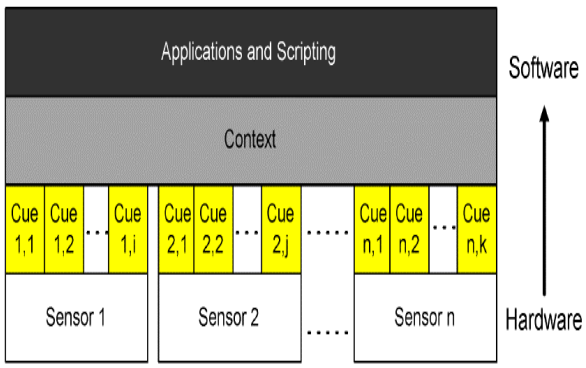

Figure 2. 3 : Architecture en couches de Schmidt.

La couche du niveau le plus bas est composée des capteurs considérés dans l'application. Chaque capteur i est modélisé par une fonction Si ayant pour unique variable le temps t. Le résultat est un signal Xi appartenant à l'ensemble des signaux pouvant être émis par i, soit :

$$S_i : t \; ? \quad X_i$$

Le niveau intermédiaire est composé d'entités appelées « indice ». Les indices sont des abstractions des capteurs physiques de la couche inférieure. Il peut y avoir plusieurs indices pour un seul capteur. Nous retrouvons ici la notion d'interpréteur du « Context-toolkit ». Chaque indice j est modélisé par une fonction Cj ayant pour paramètre l'ensemble des signaux Si émis entre l'instant t-n et l'instant t par le capteur i associé à j. Cette fonction renvoie un résultat Y_{ij} rattaché à une sémantique spécifique. Soit :

$$C_j : S_i(t) \; ? \; S_i(t \; ? \; 1) \; ? \; ... \; ? \; S_i(t \; ? \; n) \; ? \quad Y_{ij}$$

La strate de niveau supérieur regroupe l'ensemble des indices et est capable de déterminer un contexte. Cette strate se modélise aussi par une fonction mathématique T ayant pour paramètre l'ensemble des indices efficients à l'instant t. Le résultat fourni est un vecteur h (v,p) composé d'une valeur v symbolique d'une situation et d'une valeur p indiquant la valeur de confiance associée à v. Soit :

$$T : C_0(S_0,t) \; ? \; C_1(S_0,t) \; ? \; ... \; ? \; C_k(S_0,t) .. C_0(S_i,t) \; ? \; C_1(S_i,t) \; ? \; ... \; ? \; C_m(S_i,t) \; ? \quad h$$

Le vecteur h permet de définir une surface, fonction du temps, qui est ensuite comparée à des surfaces caractéristiques de situation comme représenté sur la figure 2.4. Nous avons trois axes représentant respectivement la valeur de v symbole de l'environnement, un axe p graduant une échelle de confiance et un axe t représentant le temps. Cette dernière strate a également la charge de notifier les applications des contextes en cours (voir figure 2.4).

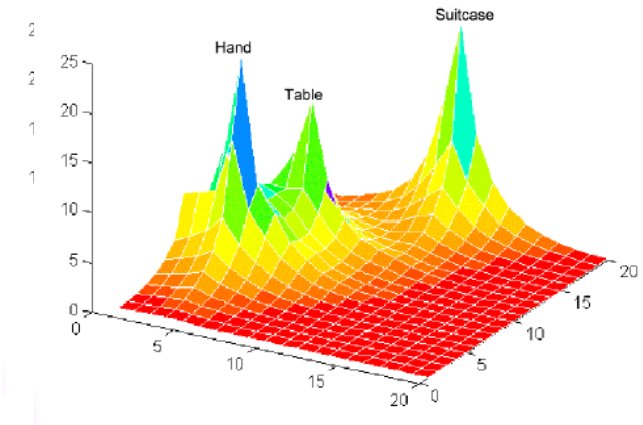

Figure 2. 4 : Fonction T dans l'architecture de Schmidt. Les pics représentent se qui a été identifier. Ici, une table, une main et une valise.

1.2.2 Atouts et handicaps

Comme pour le context-toolkit, on retrouve 3 niveaux d'abstraction : Le niveau capteur équivalent au context widget, le niveau indice représentant la notion d'interpréteur et le niveau contexte faisant office d'agrégateur comme le font les serveurs.

Contrairement au context-toolkit, on voit apparaître chez Schmidt deux notions fondamentales :
?? le temps,
?? la qualité des données (valeur de confiance).
Enfin, pour en finir avec les points positifs, Schmidt propose une méthode d'analyse du contexte par comparaison avec des situations de référence.

Inversement,
?? la force du modèle (son fondement mathématique) pourra être perçue comme une faiblesse car éloigné du concepteur logiciel qui s'attend à un modèle plus proche de ses préoccupations de mise en œuvre.
?? On regrettera également que les informations sur la qualité des données fassent leur apparition seulement au niveau 3. Pourquoi ne pas considérer la qualité dès le capteur ?
?? Enfin, la méthode d'analyse du contexte oblige la comparaison des surfaces contextuelles obtenues avec d'autres surfaces caractéristiques de situation. Cela semble assez compliqué et superflu dans certaines situations comme, par exemple, savoir s'il fait chaud.

Au bilan, le modèle de Schmidt
?? présente un fondement mathématique fort,
?? introduit les notions de temps et de qualité de service,
?? structure le problème en trois niveaux d'abstraction étanches,
?? mais peut sembler complexe et lourd pour le traitement de cas simples.

1.3 Le Context-handling component

Sur le plan architecture, ce modèle se positionne d'emblée comme une extension du modèle Arch (voir figure 2.5).

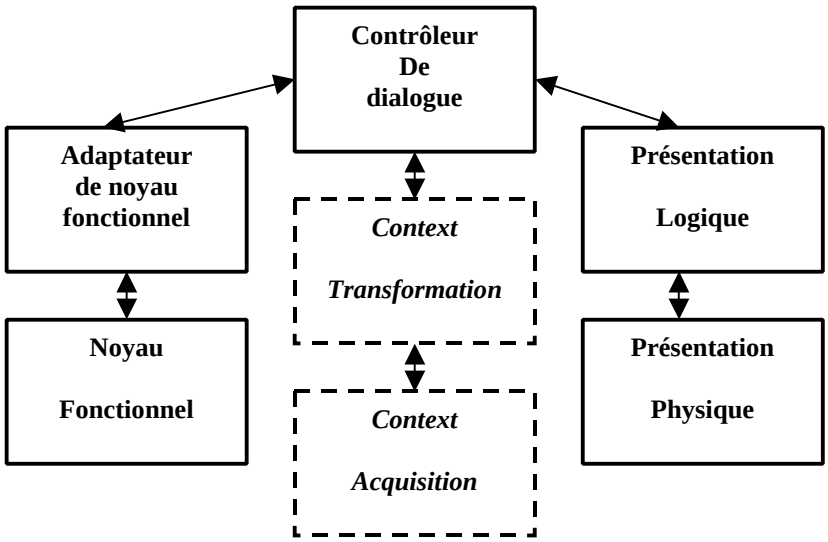

Figure 2. 5 : Extension du modèle Arch avec, en pointillé, le Context-handling Component.

Arch est l'un des modèles d'architecture de référence en Interaction Homme-Machine : aux deux pieds de l'arche, le noyau fonctionnel (ou application) et la présentation physique des concepts et services réalisés dans le noyau fonctionnel. En clé de voûte, le contrôleur de Dialogue, responsable de l'enchaînement des tâches utilisateur. De chaque côté, un adaptateur (l'adaptateur de noyau fonctionnel et le composant de présentation logique) qui minimise les dépendances du Contrôleur avec le noyau fonctionnel et la présentation physique.

Comme le montre la figure 2.5, le modèle de Salber vient se brancher sur le Contrôleur de Dialogue qui instruit à son tour la branche « présentation » et/ou la branche « noyau fonctionnel » de l'arche.

Le modèle de Salber est structuré en deux niveaux d'abstration reprenant ainsi les fondements de Arch (découplage du physique et du logique) :

?? Le niveau Context Acquisition qui prend en charge la capture de l'information dans le monde physique à l'aide de capteurs.
?? Le niveau Context Transformation qui extrait les informations significatives en provenance du Context Acquisition et les transmet au Contrôleur de Dialogue au niveau d'abstration idoine.

A leur tour, ces deux composants sont constitués d'un ensemble de composants appelés « Context handling component » dont l'interface de communication, représentée dans la figure 2.6, comprend 3 flots d'entrée et 3 flots de sortie.

?? Les flots d'entrée incluent un flot de donnée (Data In), un flot de Méta-Donnée (Meta Data In) et un flot de contrôle (Control).
?? Les flots de sortie sont composés d'un flot de donnée (Data Out), d'un flot de Meta Donnée (Meta Data Out) et d'un flot d'action (Action).

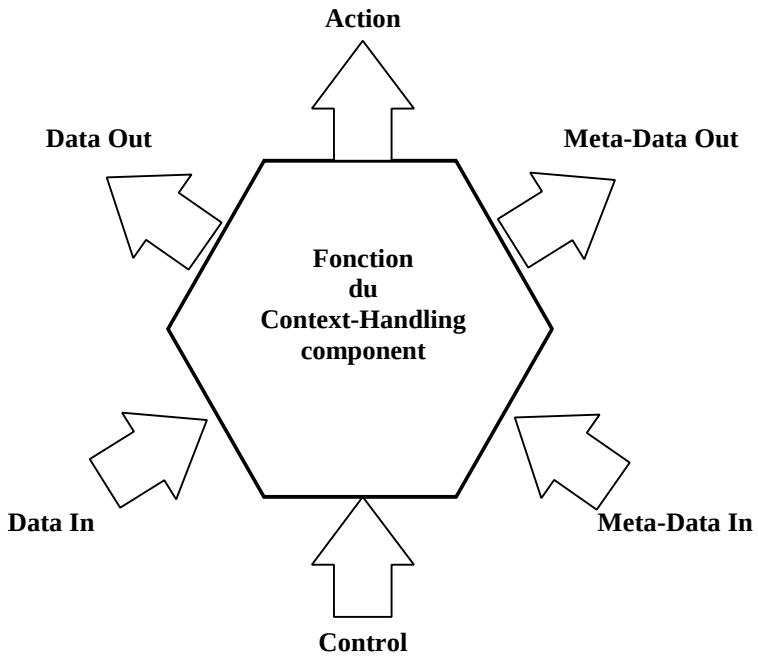

Figure 2. 6 : Interface d'un « Context-handling component ».

?? Les flots *Data* (In ou Out) correspondent aux informations reçues ou émises par le composant.
?? Les flots *Meta-Data* (In ou Out) livrent les mesures de qualité sur les informations reçues ou émises par le composant.
?? Le flot *Control* permet à d'autres composants de modifier le comportement interne du composant (supprimer l'activité de ce composant reconnu non fiable, changer les paramètres de fonctionnement, etc.).
?? Le flot *Action* sert à produire une action sur l'environnement (changer la vitesse d'un moteur, éclairer la lumière …).

Le modèle précise les règles de composition des Context-handling components : les flots *Data* (et Meta data) sont connectables entre eux, c'est-à-dire un flot Out est relié à un flot In. Il en va de même pour un flot *Action* qui vient alimenter un flot *Control*.

1.3.1 Atouts et handicaps

Ce modèle présente des avantages sur les modèles analysés jusqu'ici :
?? Il s'inscrit dans une architecture de référence en Interaction Homme-Machine,
?? Il véhicule la mesure de qualité des données à tous les niveaux d'abstraction (MetaData).

Inversement,
?? La notion de capteur disparaît,
?? Les Meta-data et les Data transitent dans des flots distincts possiblement désynchronisés. Or une donnée et sa méta-donnée sont fortement couplées. Elles doivent donc être communiquées ensemble.
?? Le modèle ne propose pas de classification pour les Context-handling components, ni ne détaille leur composition.

Au bilan, le modèle de Salber correspond le mieux à l'attente mais ses travaux présentent des lacunes que j'ai tentées de combler avec les propositions qui suivent. Celles-ci comprennent 3 volets :

?? Le transducteur, composant interne d'un dispositif de capture,
?? Le capteur, une sorte de transducteur auquel nous donnerons la capacité de fournir des données numériques,
?? Le contexteur, composant de représentation de contexte.

Les sections qui suivent reprennent ces notions une à une.

2 Transducteur

2.1 Définition

Le Comité du langage scientifique définit le transducteur par : «tout système ou dispositif intermédiaire qui, recevant de l'énergie en provenance de plusieurs systèmes ou milieux, fournit à plusieurs autres systèmes ou milieux une énergie de même nature ou de nature différente de l'énergie reçue, en correspondance avec celle-ci». [© 2001 Hachette Multimédia / Hachette Livre]. De cette définition, on peut faire la distinction entre les transducteurs d'entrée qui reçoivent de l'énergie du milieu pour la fournir à un système des transducteurs de sortie qui font l'opération inverse du système vers le milieu.

Dans cette étude, nous sommes concernés par les transducteurs d'entrée. Nous optons pour la définition suivante :

> Un transducteur est un dispositif matériel capable de détecter un phénomène physique en vue de le représenter sous la forme d'un signal analogique.

Les transducteurs sont largement utilisés en électricité et en électronique ou dans des applications de ces techniques dont notamment la famille des cristaux piézoélectriques (dont les quartz font partie) qui transforment un flux d'énergie mécanique en énergie électrique (transducteur électromécanique).

On distingue deux grandes catégorie de transducteurs : les transducteur passifs et les transducteurs actifs.

2.2 Transducteurs passif et actif

Un transducteur est passif si l'énergie qu'il délivre provient exclusivement de l'énergie qu'il reçoit. Ainsi, un filtre électrique formé uniquement de résistances et de condensateurs est un transducteur passif : il délivre une fraction plus ou moins grande de l'énergie électrique qui lui est fournie par le générateur placé entre ses bornes d'entrée.

Un transducteur est actif si l'énergie qu'il délivre provient au moins en partie de sources autres que celles qui lui fournissent de l'énergie. Un amplificateur est un transducteur

actif car il reçoit de l'énergie d'un dispositif relié à ses bornes d'entrée (générateur, capteur, antenne radioélectrique) et il restitue une énergie beaucoup plus grande entre ses bornes de sortie. Ces énergies sont transportées par des courants de même fréquence. L'amplificateur ne produit pas l'énergie qu'il cède, mais il est conçu pour permettre la fourniture de cette énergie par une alimentation incorporée dans l'appareil. C'est la constitution de l'amplificateur qui fait qu'il y a correspondance entre l'énergie de sortie et l'énergie d'entrée.

Cependant la détection d'un phénomène physique ne suffit pas aux systèmes numériques. Les systèmes sensibles au contexte doivent construire une représentation numérique de l'univers.

Aussi, j'introduis la notion d'unité de capture que je définis le capteur comme un unité de capture.

3 Unité de capture et capteur

On trouvera ci-dessous la définition d'une unité de capture suivie de notre acception du terme capteur. En 3.3, on trouvera les propriétés permettant de caractériser les qualités fonctionnelles et opérationnelle d'un capteur.

3.1 Unité de capture : définition

La figure 2.7 montre la structure d'une unité de capture. Autrement dit :

> Une unité de capture est l'association d'un transducteur et d'une fonction de calcul qui permet de quantifier l'énergie reçue du transducteur sous forme numérique en fonction d'un étalonnage.

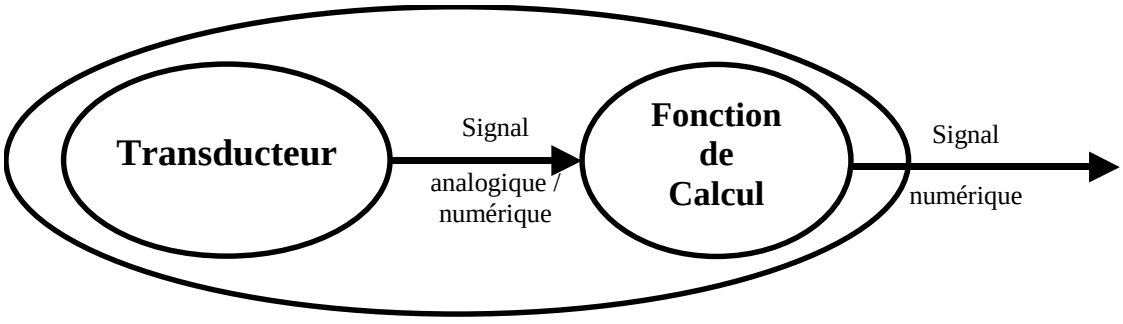

Figure 2. 7 : Structure d'une unité de capture.

3.2 Capteur : définition

Dans son acception courante, un capteur désigne un transducteur d'entrée : ce dispositif est capable de fournir des grandeurs physiques comme la lumière, le son, la température, la force, la pression, etc., ou chimiques, ainsi que leurs variations. [©Webencyclo des Éditions Atlas 2001] Pour chaque type de grandeur (mécanique,

électrique, magnétique, thermique, radiative ou chimique), le signal capté peut comporter plusieurs variables, par exemple, pour la lumière, la phase, l'amplitude, la longueur d'onde, la polarisation, etc.

Un système sensible au contexte, qui doit mesurer les phénomènes physiques de l'univers, doit comporter des capteurs mais ces capteurs doivent être encapsulés d'une unité de traitement.

Aussi, pour les besoins de cette étude, on acceptera la définition suivante d'un capteur :

> Un capteur est une unité de capture dont la fonction de calcul est un driver ou composant logiciel qui interface le transducteur d'entrée avec la plate-forme d'accueil.

Le choix d'un capteur et son intégration dans un système dépend de ses propriétés opérationnelles et de sa couverture fonctionnelle.

3.3 Propriétés opérationnelles d'un capteur

Une propriété opérationnelle caractérise la performance d'une entité sous forme, si possible, de métrique. Bérard dans [Bérard 99] présente les qualités requises d'un système de capture fondé sur la vision par ordinateur. Nous reprenons à notre compte ces travaux que nos complétons.

Il convient de distinguer les caractéristiques suivantes d'un capteur :

?? *La résolution* est la plus petite variation du phénomène physique à laquelle le capteur est sensible. Si la variation du phénomène est inférieure à la résolution du capteur, celle-ci ne sera pas relevée.

?? *La latence* est le temps écoulé entre la présentation d'un stimulus au capteur et le retour d'information qui résulte du traitement de ce stimulus.

?? *Le pas d'échantillonnage* est le temps minimal entre deux captures. En conséquence, toute variation du phénomène physique, quelle que soit son importance, survenant dans cet intervalle n'est pas relevée par le capteur.

?? *La stabilité* est la capacité du capteur à ne pas indiquer de variation si le phénomène physique ne varie pas.

?? *La portée* est la distance maximale à laquelle le phénomène physique est détectable par le capteur.

?? *Le champs d'action* est l'angle (centré sur le capteur) à l'intérieur duquel le phénomène physique peut être observé.

?? *La taille* désigne de manière simplifiée à la fois le volume, le poids et la forme du capteur. Cette caractéristique est déterminante lorsque le capteur doit être embarqué sur un dispositif mobile de petite taille.

?? *L'autonomie* est la durée de fonctionnement du capteur sans alimentation électrique. Tout capteur qui nécessite une alimentation électrique a une automomie nulle. Plus rarement, les capteurs pour lesquels le signal du phénomène physique suffit à leur fonctionnement, ont une autonomie infinie.

?? *L'orientabilité* est la capacité du capteur de changer de champs d'action. Elle peut être manuelle, motorisée ou automatique.

?? *Le calibrage* désigne le réglage des paramètres du capteur qui, à leur tour, déterminent les conditions du fonctionnement du capteur. Par exemple, pour un capteur de son, le paramètre « niveau de son » détermine le niveau sonore ambiant minimal auquel le capteur doit être sensible. Pour une caméra, le focus ou l'ouverture du diaphragme sont des paramètres de calibrage. Le calibrage peut être automatique. Dans ce cas, le délai de calibrage est une propriété pertinente du capteur. Le calibrage peut être manuel, c'est-à-dire nécessite une intervention extérieure, ou inexistant. Le calibrage peut aussi être semi-automatique : il n'est pas déclenché automatiquement, mais une fois l'ordre donné, le calibrage est automatique.

3.4 Couverture fonctionnelle d'un capteur

La couverture fonctionnelle d'une entité désigne l'ensemble des services qu'elle est capable de fournir. Concernant les capteurs, je propose de distinguer les fonctions suivantes (la liste n'est pas exhaustive) :

?? *La détection* est la capacité du capteur à repérer l'existence d'un type de phénomène physique. Par exemple, la présence d'une entité mobile dans la pièce.

?? *L'identification* est la capacité du capteur à reconnaître une instance d'une classe de phénomène physique. Par exemple, l'entité mobile en question est un chat.

?? *Le suivi* est la capacité du capteur à suivre à tout instant le phénomène physique détecté ou identifié.

Ayant introduit le composant de base, le capteur et ses propriétés, voyons comment, avec la notion de contexteur, on généralise le processus de capture du contexte par un système numérique.

4 Contexteur

Un contexteur est une abstraction logicielle qui fournit la valeur d'une variable du Contexte Système Capté.

Nous en fournissons une description, nous en précisons les propriétés, puis nous en proposons une taxonomie avant d'indiquer comment les composer. Cette section

s’achève par une analyse comparative du contexteur avec les propositions de l’état de l’art.

4.1 Description

Inspiré des « context-handling components » de Salber, un contexteur comprend trois classes d’éléments (voir figure 2.8) :

?? ses entrées,
?? ses sorties,
?? son corps fonctionnel.

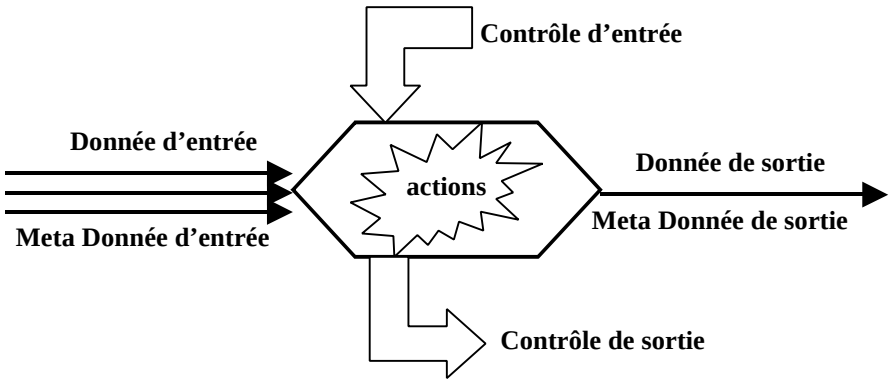

Figure 2. 8 : Modèle de contexteur.

?? *Les entrées* sont de 2 types :

le *contrôle d’entrée* permet de modifier les paramètres internes, donc le comportement, du contexteur. Le contrôle d’entrée sert notamment à arrêter le contexteur quand celui-ci a été reconnu déficient par d’autres contexteurs.

Les *données d’entrée* correspondent aux données fournies par d’autres contexteurs.

Chaque donnée d’entrée est assortie de *méta-données* qui expriment la qualité de cette donnée. La qualité peut inclure les propriétés opérationnelles énoncées pour les capteurs mais aussi un facteur de confiance.

?? Symétriquement, *les sorties* sont de 2 types :

le *contrôle de sortie* permet de modifier les paramètres internes d’un contexteur cible. L’ordre transmis par le contrôle de sortie est soumis à l’approbation des autres contexteurs utilisant le contexteur cible.

Les *données de sortie* sont des informations transmises soit à d’autres contexteurs soit à une entité logicielle externe.

Chaque donnée de sortie est assortie de méta-données qui en expriment la qualité.

?? *Le corps fonctionnel* désigne la fonction remplie par le contexteur. La nature de cette fonction détermine le type de contexteur.

4.2 Types de contexteurs

On distingue six catégories de contexteurs qui correspondent chacune à un besoin récurrent : le contexteur élémentaire, le contexteur mémoire, le contexteur à seuil, le contexteur de traduction, le contexteur de fusion et le contexteur d'extension. Nous les passons en revue en fournissant pour chacun, un exemple d'utilisation. Puis, nous les comparons au moyen d'un tableau récapitulatif.

4.2.1 Contexteur élémentaire

Un contexteur élémentaire :
?? encapsule un capteur (pris au sens où nous l'avons défini précédemment)
?? fournit en sortie les données du capteur
?? enrichit ces données au moyen de Méta-données
?? et introduit les Contrôles d'entrée et de sortie.

Les contexteurs élémentaires permettent de normaliser les capteurs et ce faisant, de les intégrer de manière cohérente dans notre modèle. La figure 2.9 illustre la structure d'un contexteur élémentaire.

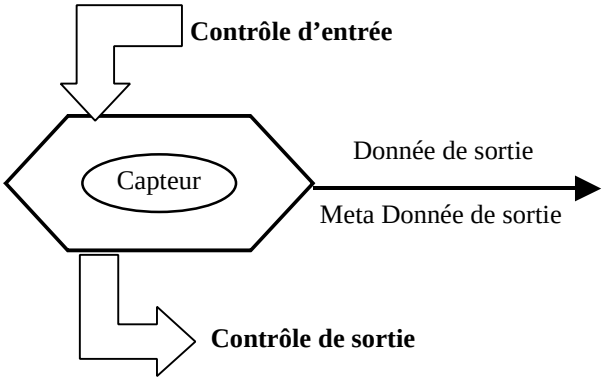

Figure 2. 9 : Modèle de contexteur élémentaire.

Exemple d'utilisation : Soit un contexteur élémentaire encapsulant une caméra vidéo. Dans ces conditions, les données de sortie correspondent au flux d'images, le contrôle d'entrée permet de régler le nombre d'images par seconde du flux de sortie, l'orientation de la caméra et le facteur de zoom. Les méta-données de sortie peuvent inclure la résolution, le nombre de bits par pixel, et de manière générale, tout facteur utile à la description de la qualité de l'image. Le contrôle de sortie est utilisé pour recevoir un retour sur l'accomplissement des ordres reçus sur le contrôle d'entrée.

4.2.2 Contexteur à mémoire

Comme son nom le suggère, le contexteur à mémoire mémorise un historique des valeurs des données et des méta-données reçues en entrée. En données de sortie, il fournit tout ou une partie de cet historique. Ce contexteur se justifie au regard de notre

notion de contexte qui est un cumul de situations : pour réaliser des cumuls, il faut avoir été capable de mémoriser les situations. La figure 2.10 montre le modèle de contexteur à mémoire.

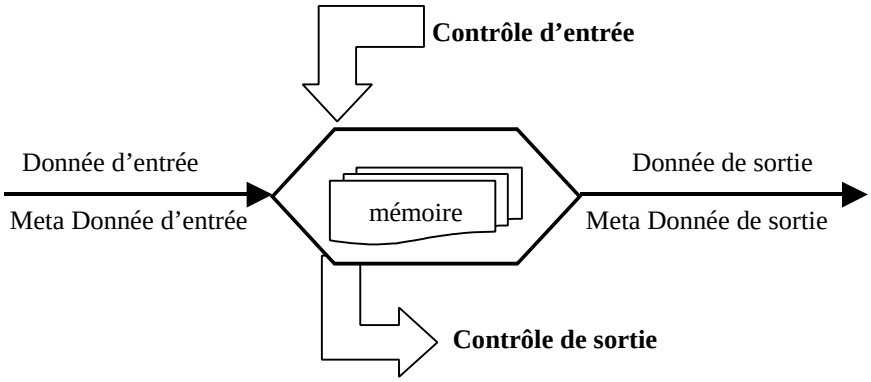

Figure 2. 10 : Modèle de contexteur à mémoire.

Exemple d'utilisation : Un contexteur à mémoire accepte en données d'entrée un flux de température en provenance d'un autre contexteur. Son contrôle d'entrée permet à d'autres contexteurs de régler le nombre de valeurs de température mémorisées en son sein. Le contrôle de sortie peut demander au contexteur précédant de changer sa fréquence d'émission des températures (suite à saturation de sa mémoire). Les données de sortie correspondent, par exemple, aux 30 dernières températures. A leur tour, ces valeurs peuvent être exploitées par un autre contexteur qui calcule une moyenne ou bien l'évolution de la température.

4.2.3 Contexteur à seuil

Un contexteur à seuil fournit en sortie une valeur booléenne qui traduit le résultat d'un test comparatif entre les valeurs des données d'entrée et la valeur d'un seuil. Voir figure 2.11.

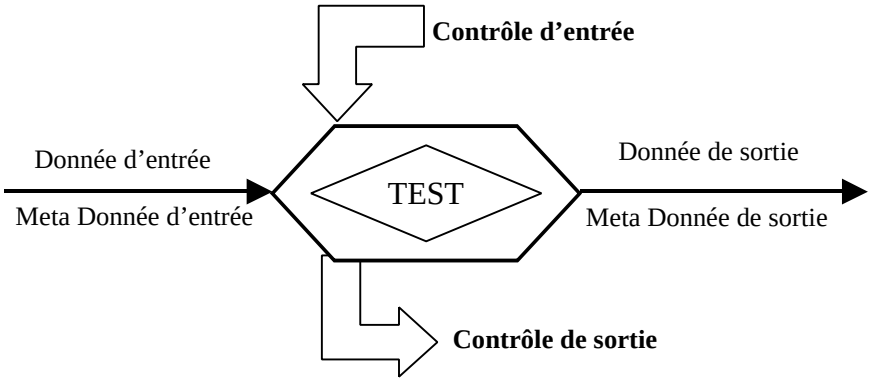

Figure 2. 11 : Modèle de contexteur à seuil.

Exemple d'utilisation : Un contexteur à seuil accepte en données d'entrée un flux de températures en provenance d'un autre contexteur. Les données de sortie répondent à la question « fait-il chaud ?». Le contrôle d'entrée permet de régler la valeur du seuil de température à partir de laquelle « il fait chaud ». Le contrôle de sortie permet de

demander au contexteur précédant de changer sa fréquence d'émission des températures.

4.2.4 Contexteur de traduction

Un contexteur de traduction retourne en sortie la valeur de la donnée d'entrée exprimée dans une autre unité. Le sens et le type de la donnée d'entrée restent inchangés. Voire figure 2.12.

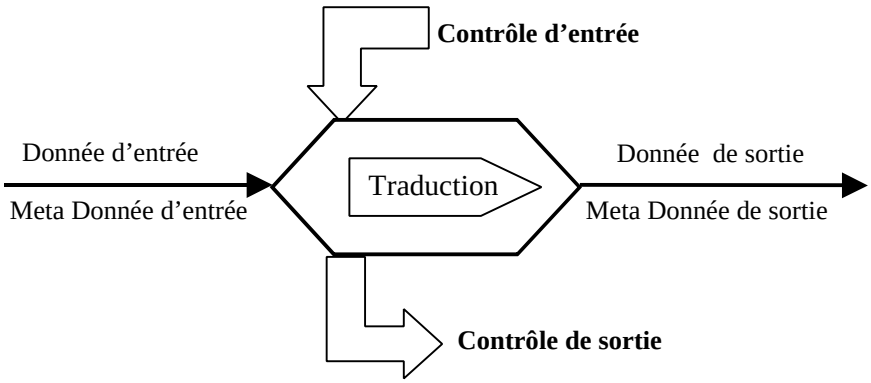

Figure 2. 12 : Modèle de contexteur de traduction.

Exemple d'utilisation : un contexteur de traduction prend en données d'entrée un flux de températures en degrés Fahrenheit en provenance d'un autre contexteur et fournit en données de sortie un flux de températures en degrés Kelvin. Le contrôle d'entrée permet de changer l'unité de traduction (Celsius, Kelvin, etc.). Comme dans l'exemple précédent, le contrôle de sortie permet de demander au contexteur précédant de changer sa fréquence d'émission des températures.

4.2.5 Contexteur de fusion

Un contexteur de fusion accepte les données de plusieurs contexteurs caractérisant la même variable du Contexte Système Capté. L'objectif d'un tel contexteur est l'amélioration des méta-données en sortie. Ce type de capteur encapsule les algorithmes de fusion tels que ceux développés en robotique (multi-sensor fusion). La figure 2.13 montre le modèle de ce type de contexteur.

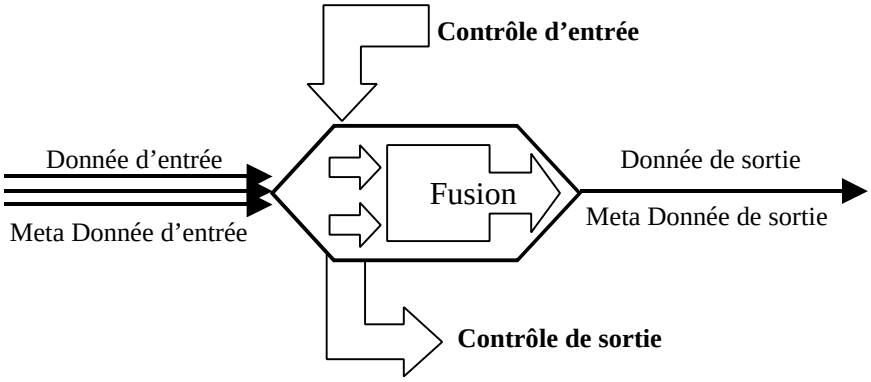

Figure 2. 13 : Modèle de contexteur de fusion.

Exemple d'utilisation : Un contexteur de fusion accepte en données d'entrées plusieurs flux fournissant chacun le nombre de personnes présentes dans une salle (l'un utilise la reconnaissance vidéo, l'autre une reconnaissance audio, etc.). Les données de sortie répondent à la question « nombre de personnes dans la pièce » avec une amélioration du facteur de qualité (facteur de confiance) de la donnée émise. Le contrôle de sortie permet d'arrêter l'une des source d'entrée si les données qu'elle émet sont de trop mauvaise qualité. Le contrôle d'entrée peut servir à modifier les coefficients pondérateurs sur les différents flux d'entrée.

4.2.6 Contexteur d'extension

Un contexteur d'extension déduit, à partir des données d'entrée, de nouvelles informations caractérisant le Contexte Système Capté. Voir figure 2.14.

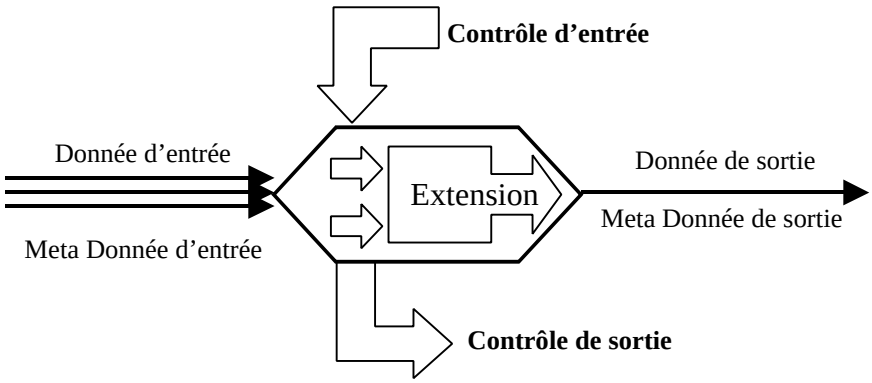

Figure 2. 14 : Modèle de contexteur d'extension.

Exemple d'utilisation : Un contexteur d'extension prend en données d'entrée un flux indiquant le poids sur une chaise, un autre la température ambiante et un troisième la température à proximité de la chaise. Les données de sortie indiquent s'il y a une personne sur la chaise. Le contrôle de sortie peut demander aux contexteurs précédents de changer leur fréquence d'émission.

4.2.7 Tableau récapitulatif

Le tableau de la figure 2.15 explicite les différences entre les classes de contexteurs. Celles-ci se distinguent par les facteurs suivants :

?? L'existence d'une liaison directe avec un capteur. Seul le contexteur élémentaire est en contact directe avec un capteur.

?? Le nombre de flux de données d'entrée. Par exemple, un contexteur de fusion comporte au moins deux flux d'entrée.

?? Le nombre de flux de données de sortie.

?? Le type des données d'entrée et de sortie. Par exemple, un contexteur de mémoire, qui reçoit des données de type X, retourne un ensemble de données du même type. Le contexteur de seuil retourne toujours un booléen quel que soit le type des données d'entrée. A l'exception du contexteur de seuil, un contexteur d'extension retourne des données de type différent des entrées. Il offre le moyen d'élever le niveau d'abstraction des données d'entrée.

Type de contexteur	Liaison directe avec un capteur	Nombre d'entrée	Nombre de sortie	Type des données d'entrée	Type des données de sortie
Élémentaire	Oui	0	1	-	X
Mémoire	Non	1	1	X	{X}
Seuil	Non	1	1	X	Booléen
Traduction	Non	1	1	X	X
Fusion	Non	2 ou +	1	X,X	X
Extension	Non	1 ou +	1	X,Y	Z

Figure 2. 15 : Récapitulatif des principales différences entre les types de contexteurs.

Si chaque catégorie de capteur a ses spécificités, tous partagent les propriétés suivantes.

4.3 Propriétés de contexteur

?? Un contexteur est réflexif : il est capable de s'auto-décrire et de fournir les informations suivantes :
- Son nom,
- La catégorie à laquelle il appartient. C'est a dire à la catégorie à laquelle appartient la variable qu'explicite le contexteur. On peut voir cette catégorie comme un groupe caractérisant la variable et permettant de classer les contexteurs dans une base de donnée.
- Son Interface d'entrée (Donnée In, Meta-Donnée In, Contrôle-In),
- Son Interface de sortie (Donnée Out, Meta-Donnée Out, Contrôle-Out).

?? Un contexteur est capable de modifier dynamiquement son fonctionnement en fonction des ordres qu'il reçoit sur son port « Control-In ».

?? Un contexteur est capable de s'arrêter s'il n'est plus utilisé. Cette propriété implique que si son état interne a une importance pour un futur redémarrage, il doit être capable de le sauvegarder.

?? Un contexteur est rémanent : il est capable de se charger dynamiquement dans le système et de restaurer son état interne si besoin est.

?? Un contexteur peut exister en plusieurs instances simultanées. Par exemple le contexteur de seuil « il fait chaud » est instancié pour plusieurs pièces différentes doté chacun de valeurs de seuil différentes. Pour augmenter la fiabilité d'un système critique, un contexteur pourra être « cloné » plusieurs fois.

Jusqu'ici, nous avons étudié le contexteur en tant qu'unité de représentation de variable de contexte système capté. Voyons comment ils se composent. Nous offrons deux mécanismes : assemblage hiérarchique et encapsulation.

4.4 Hiérarchisation

Les sorties d'un contexteur sont reliées aux entrées d'un autre. Et les sorties d'un contexteur peuvent être utilisées par plusieurs contexteurs. On obtient ainsi un graphe orienté hiérarchique. La figure 2.17 en montre un exemple.

Cette méthode de structuration permet d'étudier les contexteurs d'après leur niveau dans la hiérarchie à un instant donné (le niveau hiérarchique d'un contexteur pouvant évoluer durant son existence). Le niveau hiérarchique d'un contexteur se définit de la manière suivante :

?? Tous les contexteurs élémentaires possèdent un niveau hiérarchique nul (NH = 0).
?? Pour les autres contexteurs, leur niveau hiérarchique est égal au niveau hiérarchique du contexteur en entrée de niveau le plus haut + 1.

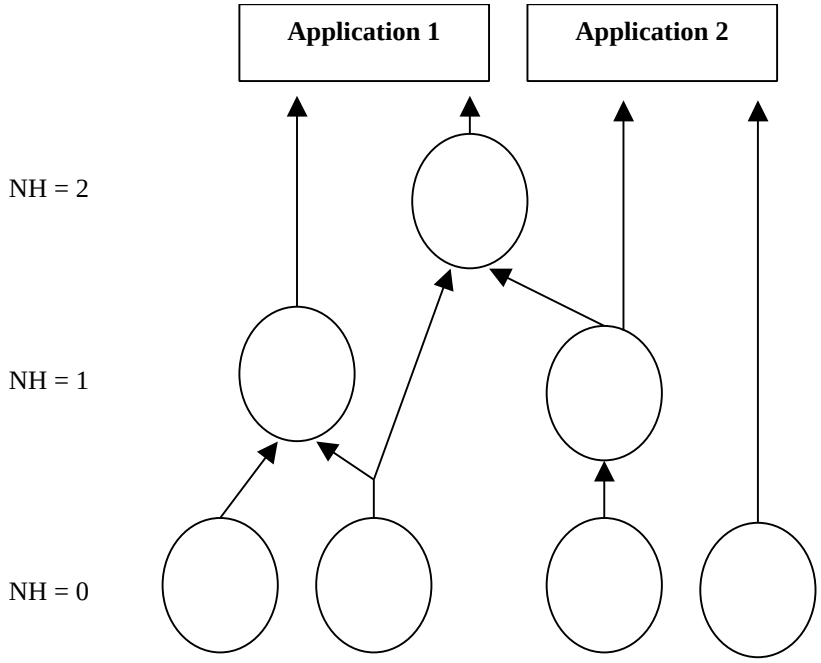

Figure 2. 16 : Exemple de hiérarchisation d'un ensemble de contexteurs.

Le niveau hiérarchique d'un contexteur sert à définir sa dépendance. Plus il est faible, moins le contexteur est dépendant d'autres contexteurs. Au contraire, plus il est élevé, plus le contexteur est dépendant de la présence d'autres contexteurs. La valeur du niveau hiérarchique indique la taille (en nombre de contexteurs) de la plus grande chaîne de contexteurs précédant le contexteur en question.

On appelle *chaîne de dépendance* d'un contexteur fournissant un service à une application, l'ensemble composé du contexteur et des contexteurs dont il a besoin pour fonctionner et ceci récursivement jusqu'aux contexteurs élémentaires.

On peut classer un contexteur suivant la place qu'il occupe dans sa chaîne de dépendance en trois rangs :

✍✍La capture
✍✍L'interprétation
✍✍L'utilisation

?? *La capture.* Cette étape est la première dans la chaîne. Elle se fait essentiellement au niveau des contexteurs élémentaires. Elle demande des compétences en électronique pour la création et / ou l'utilisation de capteur physique.

?? *L'interprétation.* Cette étape est traitée par les maillons intermédiaires de la chaîne de dépendance. Elle est donc essentiellement une étape de création de contexteur. Bien qu'elle ne demande pas de connaissance en électronique, elle peut nécessiter des aptitudes diverses (vision, ...).

?? *L'utilisation.* Cette étape arrive en fin de chaîne de dépendance. Elle est concentrée dans le maillon de niveau hiérarchique le plus élevé. Elle sert à faire le pont entre le monde applicatif classique et le monde sensible au contexte. Les compétences demandées pour les développeurs d'applications sensibles au contexte sont donc pratiquement les mêmes que pour le développement d'applications classiques.

4.5 Encapsulation

Le principe d'encapsulation permet de réutiliser des contexteurs prédéfinis, de les assembler et d'en masquer la composition pour fournir un nouveau service. Le modèle impose que cette composition corresponde à la définition d'un contexteur (en termes de flots d'entrée et de sortie) et obéisse aux propriétés de contexteur présentées en 4.3.

La figure 2.17 montre un exemple de mise en œuvre d'un contexteur d'extension encapsulant un assemblage de contexteurs organisés hiérarchiquement. On note que l'encapsulaiton permet de réduire le niveau de hiérarchie de contexteurs et donc le degré de dépendance.

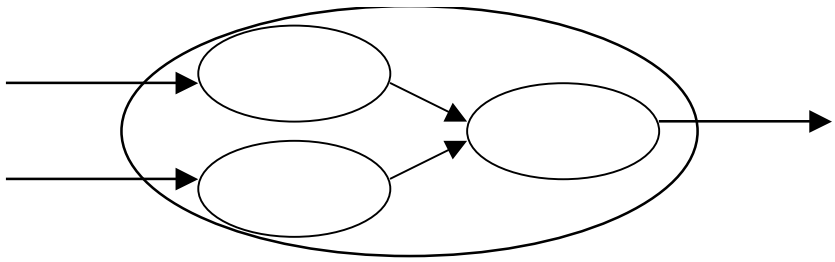

Figure 2. 17 : Exemple de contexteur d'extension obtenu par encapsulation.

4.6 Contexteur et état de l'art

Le contexteur rappelle le principe des « context-widgets » du Context-toolkit de Dey et du « context handling component » de Salber. Il les améliore en plusieurs points :

Par rapport au modèle de Dey, le contexteur fournit des informations qui font sens (inutile donc d'introduire la notion d'interpréteur). En outre un contexteur, tout comme

un « context handling component », dispose de contrôles, et décore ses retours d'information de méta-données qui expriment la qualité de sa production. Comme Schmidt, Dey introduit trois niveaux d'abstraction. Le mécanisme de hiérarchisation que je propose est général et n'impose pas de limite. Nos trois niveaux de dépendance rappellent cependant ces distinctions mais plutôt pour évoquer les compétences requises auprès des développeurs.

Par rapport au modèle de Salber,

?? Le contexteur associe sur un même flot les données et les méta-données qui sont, par nature et fonction, fortement couplées.
?? Le modèle est affiné en classes de contexteurs que n'offre pas Salber.
?? Le modèle introduit deux mécanismes complémentaires de composition de capteurs : la hiérarchisation et l'encapsulation. Cette dernière offre une vision récursive du modèle (un assemblage de contexteurs constitue un contexteur) que l'on ne retrouve pas chez Salber.
?? Enfin, nous exprimons des requis et des propriétés de contexteurs qui permettent de diriger les solutions de mise en œuvre présentées au chapitre suivant.

Chapitre III : Contexteur : modèle implémentationnel

Au chapitre précédent, nous avons vu le modèle conceptuel d'un contexteur. Dans ce chapitre, nous en voyons la dimension technique. Ici, le contexteur est vu comme un composant logiciel. En tant que tel, il a un cycle de vie. Ce cycle doit être géré par un composant dédié : dans notre cas, un serveur de contexteurs. La structure de ce chapitre suit cette logique : après la présentation du cycle de vie d'un contexteur, on étudie les possibilités de mise en œuvre en matière de serveur de contexteurs. A cet égard, on analyse l'apport de Jini avant de présenter la solution adoptée. Ce chapitre se termine avec la description de maquettes d'application sensibles au contexte démontrant la faisabilité de l'approche : les tasses disponibles à la cafétéria, les machines en fonctionnement sur un réseau local, le niveau d'activité de l'utilisateur sur sa station. Chaque exemple montre une configuration distincte de contexteurs.

1 Cycle de vie d'un contexteur

Un contexteur étant un composant logiciel autonome, il répond à un cycle qui rythme son existence. On distingue 5 états dans la vie d'un contexteur et ceci quel que soit son type. La Figure 3.1 en donne une représentation sous forme d'automate d'états finis.

?? *L'état 1* correspond à la phase où le contexteur se trouve sous forme d'un fichier binaire exécutable stocké sur un disque. Le chargement en mémoire de ce binaire (exécution du programme du contexteur) provoque le passage à l'état 2.

?? *L'état 2* correspond à la phase d'exécution avant la connexion au serveur de contexteur. Le contexteur est ici isolé. Cet état est transitoire : le contexteur va chercher à entrer en contact avec le serveur. Une fois la connexion établie, il passe dans l'état 3.

?? *L'état 3* est la phase d'identification du contexteur auprès du serveur. Une fois le contexteur identifié, il rentre dans l'état 4.

?? *L'état 4* correspond à la phase l'exécution « utile » du contexteur : le contexteur remplit sa fonction. Lorsqu'aucune application ou aucun contexteur ne l'utilise, il est suspendu. Il passe alors dans l'état 5.

?? *L'état 5* correspond à la phase de repos du contexteur. Il est de nouveau sous une forme de binaire exécutable éventuellement stocké en mémoire secondaire, mais à la différence de l'état 1, le contexteur est ici connu et identifié par le serveur. Si un composant le sollicite à nouveau, il est à nouveau activé et rentre dans l'état 4. Si le serveur décide de l'effacer de sa base, le contexteur revient dans l'état 1.

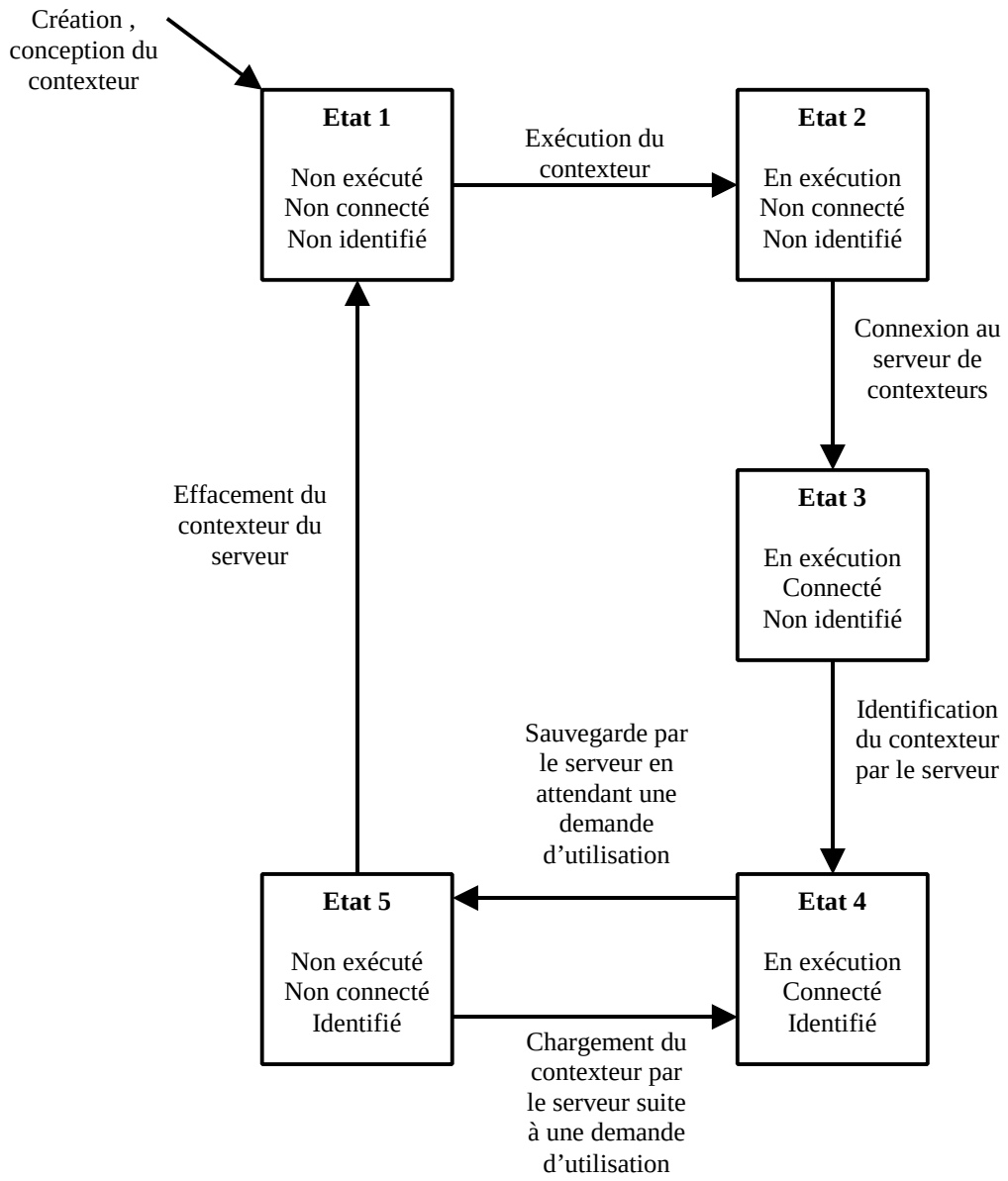

Figure 3. 1 : Automate d'états finis représentant le cycle de vie d'un contexteur.

Avant de présenter la solution adoptée pour le serveur de contexteurs, il convient d'analyser les offres en la matière.

2 Etude d'un serveur existant

Les technologies de partage de service tendent à s'imposer dans la communauté architecture logicielle et notamment dans le monde Java. Ayant choisi, pour des raisons de facilité et de temps, de développer le serveur de contexteurs en Java, je me devais d'étudier les possibilités de la technologie JINI [Avalon].

JINI s'appuie sur un concept simple : le partage de services entre les membres d'une communauté spontanée de dispositifs, sans aucune préparation préalable, installation supplémentaire ou intervention humaine. Les dispositifs se connectent, se reconnaissent

et travaillent ensemble, de n'importe où, et n'importe quand. Ils déclarent les services qu'ils sont capables de rendre et inversement expriment leurs besoins selon un protocole.

Les appareils et leurs services se déclarent comme un service. Dès qu'un élément rejoint la communauté, il manifeste sa présence en se déclarant par un processus d'enregistrement. Ce dernier enregistre alors le nouveau service.

Pour utiliser un service, une personne ou un programme consulte la liste du serveur de services mis à jour dynamiquement. L'objet du service désiré est alors copié vers le membre de la communauté qui en a fait la demande et d'où il sera utilisé.

Le serveur ayant rempli son rôle d'intermédiaire laisse en liaison directe le fournisseur et l'utilisateur du service. Le lieu d'exécution du service est sans importance, la compatibilité étant assurée par chaque service qui procure tout ce dont le client a besoin.

Chaque membre de la communauté peut retirer ses services. Le serveur remettra sa liste à jour automatiquement en y supprimant l'ex-membre ainsi que ses services.

Java est l'élément clé permettant à JINI de fonctionner. Les membres de la communauté sont mis en relation par le protocole RMI (Remote Method Invocation) de Java. JINI ne se contente pas uniquement de définir des protocoles de communication. Il inclut aussi des mécanismes de transaction qui permettent de construire une architecture distribuée en réseau dynamique et hétérogène sans installation ou déclaration préalable. Une communauté spontanée peut être créée entre stations de travail classiques, mais aussi entre appareils domestiques ou téléphones portables sans aucun ordinateur défini comme tel.

JINI constitue donc un bon support à la création d'une communauté de contexteurs, cet outil permettant de garantir le dynamisme et la robustesse nécessaire. Cependant JINI a ses limites. Il nécessite la présence de Java avec RMI ce qui limite l'implémentation des contexteurs au langage Java et aux plates-formes supportant ce langage. Un des autres points négatifs de JINI est le fait qu'il sert juste d'intermédiaire entre le service et l'application désirant ce service. La gestion des ports de contrôle nécessite une connaissance globale de « qui communique avec qui » car il sera mal venu de modifier un paramètre d'un contexteur pour en satisfaire un autre si cela en dérange plusieurs autres. Or JINI ne peut répondre à cette attente puisqu'il sert juste d'entremetteur entre un service et une application.

Les limites de JINI justifient le développement d'un serveur de contexteurs. Ce serveur est grandement inspiré du mode de fonctionnement de JINI tout en essayant de palier ses limites.

3 Le serveur de contexteurs

Le but du serveur de contexteur est de prendre en charge la gestion des contexteurs et de permettre leur mise à disposition le plus simplement possible.

3.1 Description

Le serveur de contexteurs comprend les cinq composants fonctionnels suivants : l'enregistreur, la base, les gestionnaires de déploiement, d'abonnement et d'activité. La figure 3.2 en montre les relations.

?? *L'enregistreur de contexteurs* a pour fonction d'enregistrer les nouveaux contexteurs dans une base de contexteurs après avoir vérifié que ces derniers répondent bien à la norme demandée. Il gère aussi l'effacement des contexteurs de la base quand ceux ci en font la demande ou quand leur délai d'existence est dépassé.

?? *La base de contexteurs* enregistre l'identité et les caractéristiques des contexteurs disponibles. Un contexteur peut y être enregistré pour une durée déterminée ou jusqu'à son désabonnement. On notera qu'un serveur de contexteurs peut contenir des contexteurs dès sa mise en place.

?? *Le gestionnaire de déploiement* teste si un contexteur est déjà déployé et le déploie si ce n'est pas le cas. Il gère également le déploiement des contexteurs à instances multiples, de même l'archivage des contexteurs qui ne sont plus sollicités par les applications ou par d'autres contexteurs. Il doit donc être capable de décider si l'état interne du contexteur doit être maintenu ou non.

?? *Le gestionnaire d'abonnement* permet aux applications de s'abonner aux données fournies par certains contexteurs. Il vérifie si le contexteur demandé est disponible dans la base de contexteurs. C'est aussi lui qui sollicite le système de déploiement en lui indiquant pour chaque contexteur requis par l'application s'il en faut une nouvelle instance ou non. Il doit aussi avertir les applications de la disparition de contexteur auquel elles s'étaient abonnées. Enfin, il gère le désabonnement des applications au service des contexteurs. Et par conséquence il doit prévenir le gestionnaire de déploiement pour que celui-ci vérifie si les contexteurs référencés par les applications doivent être maintenus actifs ou non.

?? *Le gestionnaire d'activité* prend en charge la gestion des connexions du contrôle d'entrée et de sortie de chaque contexteur. Il gère donc un système d'élection lors de demande de modification.

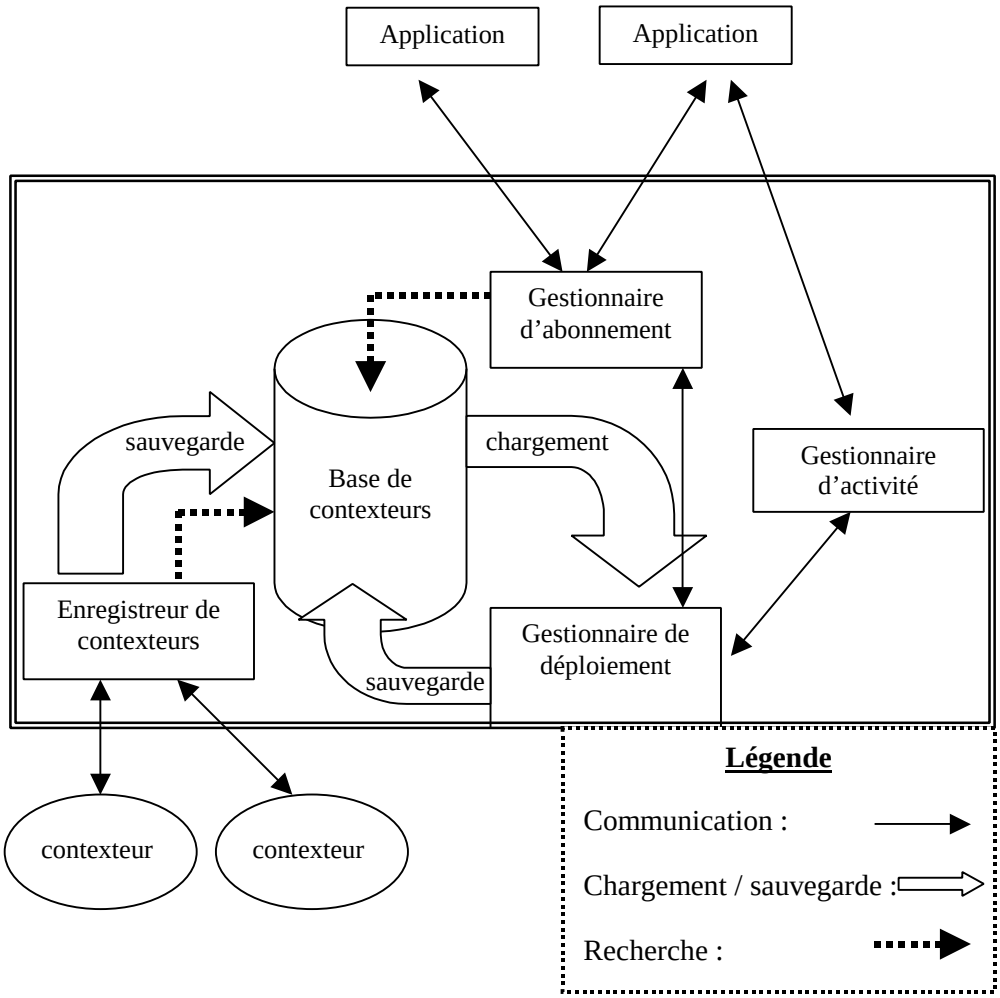

Figure 3. 2 : Structure fonctionnelle d'un serveur de contexteurs.

Dans les deux sections qui suivent, on montre, à l'aide d'exemples, le flux d'information entre les composants fonctionnels du serveur. Les exemples supposent que le serveur de contexteurs est déployé sur un réseaux de machines. Les détails de l'implémentation du serveur sont laissés aux choix des développeurs du serveur. Celui-ci peut être centralisé sur une machine ou réparti sur un ensemble de machines. Ce détail de mise en œuvre est sans incidence sur les exemples qui suivent.

Une fois que le serveur opérationnel, il attend soit l'arrivée de nouvelles applications, soit de nouveaux contexteurs. Il doit aussi gérer le départ d'anciens contexteurs qui peuvent devenir inutiles à cause de l'absence de contexteurs les précédant dans leur chaîne de dépendance (ou pour les contexteurs élémentaires, l'absence du hardware auquel ils sont associés).

Nous détaillons maintenant la connexion de contexteurs et celle de l'application, puis nous analysons le traitement particulier des flots de contrôle.

3.2 Connexion de contexteurs

Nous prendrons deux exemples complémentaires : le cas du contexteur élémentaire et le cas d'un contexteur composé.

Un nouveau contexteur élémentaire est déployé sur le réseau. Il encapsule une caméra vidéo qui, placée dans un bureau, fournit un flux d'images vidéo. Grâce à son « port » de contrôle d'entrée, on peut lui demander de zoomer, de faire pivoter la caméra. Le contexteur, quant à lui, peut demander une augmentation de la lumière de la salle à l'aide de son port de contrôle de sortie.

Le contexteur élémentaire se connecte à l'enregistreur de contexteurs auprès duquel il s'identifie. Durant l'identification, l'enregistreur vérifie que le contexteur répond bien aux normes et aux spécifications mises en place dans le serveur (il récupère l'ensemble des informations caractérisant le contexteur : nom, données de sortie, action des contrôles, fonction, capteur …).

Une fois l'identification terminée, le contexteur répondant aux critères d'identification, est archivé dans la base de contexteurs dans l'attente d'une utilisation future. A ce moment, le serveur peut mettre en place un système de compte à rebours indiquant le temps restant avant l'effacement du contexteur, ou la vérification de son fonctionnement.

Analysons maintenant le cas d'un contexteur non élémentaire. Soit un contexteur d'extension admettant comme entrée un flux d'images vidéo et donnant en sortie le présence ou l'absence de personnes dans une salle. Le contrôle de sortie peut envoyer des ordres pour faire bouger la camera. Le contrôle d'entré reçoit des données pouvant modifier des paramètres internes du contexteur.

Comme pour le contexteur précédent, le contexteur d'extension se connecte à l'enregistreur de contexteurs et s'identifie. Comme pour le contexteur élémentaire, l'enregistreur vérifie que le contexteur répond aux normes. En sus, l'enregistreur analyse les données d'entrée du contexteur et vérifie qu'il dispose des contexteurs capables de fournir ces données. Si c'est le cas, le contexteur est archivé dans la base dans l'attente d'une utilisation future. Si ce n'est pas le cas, le contexteur est marqué comme « inutilisable » car il ne possède pas les données d'entrée. Son états sera réinitialisé quand un contexteur, pouvant fournir les données d'entrée, sera disponible. Ce protocole permet une plus grande souplesse quant à l'ordonnancement de l'enregistrement des contexteurs.

Poursuivons l'exemple avec l'arrivée d'une application.

3.3 Connexion d'application

Le serveur est déployé et dispose d'une base de contexteurs y compris les deux contexteurs introduits ci-dessus. Une application de surveillance automatique doit détecter la présence de personnes dans une pièce. Elle se connecte au gestionnaire d'abonnement du serveur et demande si elle peut obtenir cette information. Le gestionnaire émet une requête auprès de la base de contexteurs qui, en l'occurrence,

répond favorablement. Le gestionnaire d'abonnement en informe l'application et demande au gestionnaire de déploiement de mettre en place le contexteur.

Le gestionnaire de déploiement charge le contexteur depuis la base, vérifie ses dépendances et les charge si elles ne sont pas déjà en activité. Si un contexteur se trouve dans la base, toutes ses dépendances s'y trouvent ou si ce n'est temporairement pas le cas, il est marqué « indisponible ». Une fois la « chaîne » de contexteurs déployée, le gestionnaire inscrit leurs dépendances dans le gestionnaire d'activité (on détaillera cette partie au paragraphe suivant) et informe le gestionnaire d'abonnement que tout est prêt à fonctionner.

Le gestionnaire d'abonnement est alors en mesure d'abonner l'application aux données qu'elle a demandées. Elle recevra directement les données du contexteur. Après plusieurs heures de fonctionnement l'application n'a plus besoin du service auquel elle s'est abonnée. Elle en informe le gestionnaire d'abonnement.

Le gestionnaire d'abonnement commence par désabonner l'application du service puis contacte le gestionnaire de déploiement pour qu'il arrête le contexteur. Le gestionnaire de déploiement vérifie que le contexteur n'est pas utilisé. Si une autre application, ou un autre contexteur l'utilise, il ne fait rien. Sinon il arrête le contexteur et applique le même algorithme à ses dépendances.

L'arrêt de contexteur revêt deux formes : soit le contexteur est retiré de la mémoire, soit le contexteur est retiré avec sauvegarde de son état dans la base. On notera que la deuxième n'intervient que si l'état interne du contexteur a un sens. Si c'est le cas, un nouvelle exemplaire de contexteur est créé au moment de la sauvegarde.

Pour finir le tour d'horizon des différents composants du serveur de contexteurs, étudions le gestionnaire d'activité.

3.4 Traitement des flots de contrôle

Reprenons l'exemple au moment où les contexteurs sont déployés par le gestionnaire de déploiement. Le gestionnaire d'activité vient de recevoir la liste de dépendances des contexteurs nouvellement déployés. Il ajoute cette liste au graphe de dépendances qu'il maintient. Ce graphe lui permet, lors de l'arrêt d'un contexteur, d'informer le gestionnaire de déploiement, de ses dépendances. Il lui permet aussi de savoir si ce contexteur est ou n'est pas utilisé par d'autres contexteurs ou d'autres applications. Lors de l'arrêt des contexteurs, il les retire de son graphe de dépendances.

Une seconde fonction du gestionnaire d'activité est de prendre en charge les requêtes de modification émises sur les ports de contrôle. Lorsqu'un contexteur émet une demande de modification sur un paramètre d'un autre contexteur (exemple bouger la caméra), ou si une application demande une modification sur le service qu'elle reçoit, le gestionnaire d'activité regarde dans son graphe tous les contexteurs ou applications qui dépendent de lui. Si personne d'autre que l'émetteur de la requête ne dépend du contexteur, la requête est transmise sinon deux cas peuvent se présenter en fonction de l'implémentation du gestionnaire d'activité :

?? Soit il informe le contexteur, émetteur de la requête, que celle-ci ne peut aboutir (solution simplement implémentable).
?? Soit il procède à une élection auprès de tous les clients du contexteur. Si ceux-ci acceptent la modification, celle-ci aura lieu, sinon il informe le contexteur, émetteur de la requête, que celle-ci ne peut aboutir. Le principe d'élection est laissé au choix des développeurs du serveur, mais un système d'élection par veto semble un choix raisonnable.

4 Mise en œuvre

Pour valider les modèles développés dans cette étude, j'ai réalisé en Java un serveur de contexteurs. Le choix du langage Java se justifie ainsi : langage de haut niveau disponible sur un grand nombre de plates-formes, qui inclut la notion d'objets utile à la définition d'une classe abstraite de contexteurs. Java supporte les sockets TCP choisies comme connecteur uniforme entre les composants contexteurs, serveur de contexteurs et applications. Nous allons successivement voir la réalisation du serveur de contexteur et des exemples d'application.

4.1 Le serveur de contexteurs

L'implémentation du serveur de contexteurs s'est limitée à certains composants du serveur dont on a présenté précédemment la décomposition fonctionnelle. Cette limitation s'est imposée à nous dans le temps imparti à l'étude. De plus, le développement complet du serveur demanderait des compétences approfondies dans certains domaines techniques que je ne maîtrise pas complètement.

En l'état, le serveur comprend :

?? **L'enregistreur de contexteurs** implémenté dans sa totalité.

?? **La base de contexteurs** qui gère l'enregistrement de l'identité et des caractéristiques de tous les contexteurs disponibles dans le système. Un contexteur ne peut cependant pas y être enregistré pour une durée déterminée mais seulement jusqu'à son désabonnement.

?? **Le gestionnaire de déploiement** implémenté dans sa quasi totalité. Seule la gestion des multiples instances n'a pas été prise en compte.

?? **Le gestionnaire d'abonnement** a été implémenté dans sa totalité à l'exception de la gestion des instances multiples.

?? **Le gestionnaire d'activité**, en cours de développement, n'a pas été inséré dans la version actuelle du serveur de contexteurs.

Le serveur n'a pas d'IHM permettant à un utilisateur d'en contrôler le fonctionnement. Cependant, une fois en fonction, le serveur n'a pas vraiment besoin d'administration puisqu'il gère automatiquement l'ajout et le retrait des contexteurs. J'ai donc seulement représenté l'état de la base de contexteurs en permettant la visualisation de l'identité et des caractéristiques des contexteurs qui y sont enregistrés. La figure 3.3 montre l'IHM de la base de contexteurs.

Ce chapitre se termine avec la description de maquettes d'application sensibles au contexte dont j'ai eu l'idée.

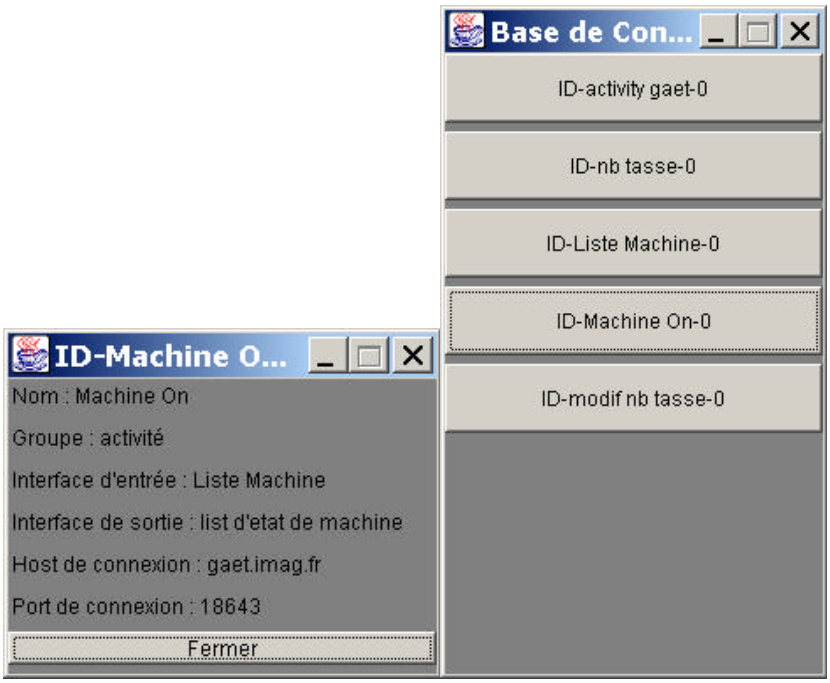

Figure 3. 3 : A droite, l'IHM du contexteur « Machine On ». A gauche, l'IHM de la Base de contexteurs.

4.2 Rappel automatique des tasses dans la cafétéria

La cafétéria du laboratoire CLIPS met à la disposition de ses membres des tasses pour que ceux-ci puissent boire le café durant leurs poses. Cependant, certains membres profitent de leurs poses pour continuer leurs travaux et emportent les tasses dans leur bureau et, distraits, oublient de les ramener à la cafétéria à la pose suivante. Au bot d'un certain temps, la cafétéria n'a plus de tasse à offrir ! Alors circulent les mails de rappel à l'ordre !

L'idée est de réaliser une application qui suit le niveau de disponibilité des tasses. La figure 3.4 en montre les composants.

Le contexteur « nombre de tasses » encapsule un capteur vidéo et calcule le nombre de tasses présentes dans l'image. Pour faciliter la détection de la présence de tasses et permettre une implémentation rapide et fiable du contexteur, nous avons utilisé un fond uniforme qui tranche avec la couleur des tasses.

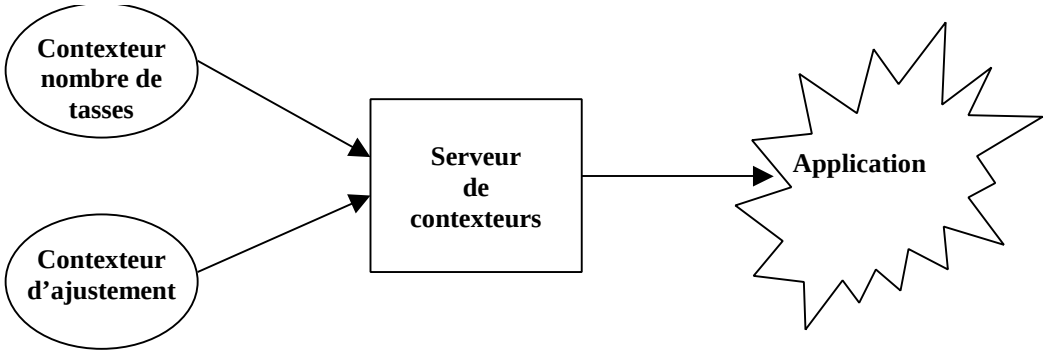

Figure 3. 4 : : Représentation des composants nécessaires au fonctionnement de l'application « tasses de la cafétéria ».

L'application s'abonne auprès du serveur pour obtenir l'information sur le nombre de tasses. Ne désirant pas déranger les membres du laboratoire, et pour faciliter la représentation de l'information obtenue par l'application nous avons choisi de représenter la disponibilité des tasses par un code couleur à l'intérieur d'une petite fenêtre. Il s'agit d'un dégradé allant du vert au rouge. Le vert indique une quantité de tasses suffisante, le rouge un manque de tasses.

Pour enrichir cette application j'ai créé un contexteur qui fournit une valeur de contrôle sur le nombre de tasses disponibles. Ce contexteur fournit à l'application une valeur (entière) comprise entre +3 et –3 permettant à celle-ci de modifier son information concernant la disponibilité des tasses. Cette valeur est calculée par le contexteur en fonction de la date courante. Il paraît évident que suivant l'heure de la journée le nombre de tasses nécessaires à la cafétéria n'est pas le même. Les personnes consommant plus facilement du café entre 8h et 10h qu'entre 18h et 20h.

4.3 Machines en fonctionnement

Contrairement à l'exemple précédent, celui-ci a été conçu pour présenter simplement l'abonnement et la communication entre contexteurs. L'application affiche en permanence la liste des machines d'un certain groupe (en l'occurrence, il s'agit de l'ensemble des machines de l'équipe IIHM) et indique si elles sont en marche ou non. Pour ce faire, elle s'abonne auprès du serveur de contexteurs et lui demande si l'information est disponible.

Pour répondre aux besoins de cette application j'ai ajouté deux nouveaux contexteurs dans la base de contexteurs. Le premier renvoie la liste des machines de l'équipe IIHM. Le deuxième indique, pour chacune des machines de la liste qu'on lui a fourni en entrée, si elles sont en marche ou non. La figure 3.5 montre la configuration correspondante.

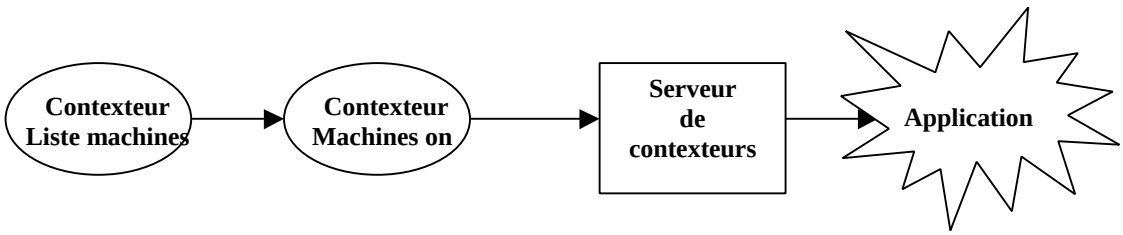

Figure 3. 5 : Représentation des composants nécessaire au fonctionnement de l'application « Machines en fonctionnement ».

L'abonnement de l'application au contexteur « machines On », par l'intermédiaire du serveur, provoque l'abonnement du contexteur « machines On » au contexteur « liste machines ». On notera que les abonnements se font en cascade et dans le sens opposé à celui de la communication des informations. Dans le cas où l'un des maillons de la chaîne de contexteurs est manquant, l'application est avertie que l'information à laquelle elle désire s'abonner n'est pas disponible.

4.4 Activité sur une station de travail

Cet exemple a pour but d'illustrer une fusion d'informations. Pour cela, nous avons conçu une application fournissant le niveau d'activité de l'utilisateur sur une station de travail. L'activité est mesurée à partir des actions clavier et/ou souris. L'objectif est de montrer que l'encapsulation de capteurs tels que le clavier er la souris peut servir à renseigner des variables de contexte. Un contexteur prend en charge l'activité du clavier, un autre l'activité de la souris tandis qu'un troisième gère la fusion des informations en provenance des deux premiers.

L'application s'abonne auprès du serveur pour obtenir l'information concernant l'activité sur la station de travail. Elle n'a absolument pas connaissance du fait que cette information provient de deux contexteurs distincts et qu'elle est fusionnée par un troisième. Comme pour les tasses, j'ai choisi de représenter l'activité sur la station de travail par un dégradé de couleurs allant du vert quand il y a de l'activité au rouge quand il n'y a pas d'activité.

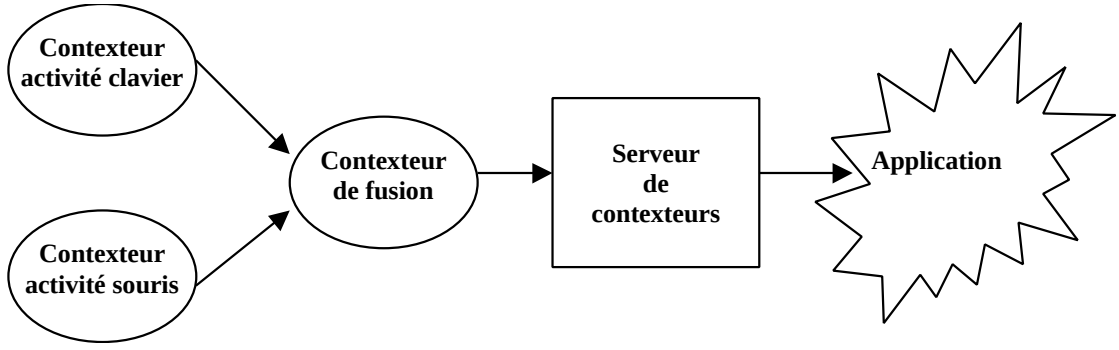

Figure 3. 6 : Représentation des acteurs nécessaire au fonctionnement de l'application « niveau d'activité ».

Le contexteur « activité clavier » capture les évènements « Key Down » du clavier qui correspondent à la frappe d'une touche. Il transmet alors au contexteur de fusion le nombre d'événements détectés par seconde.
Le contexteur « activité souris » fait de même avec les événements « Mouse Down » correspondant à la frappe sur un bouton de la souris ainsi qu'avec les événements « Mouse Move » correspondant au déplacement de la souris.

Le contexteur de fusion somme le nombre d'événements reçus toutes les cinq secondes et en fait la moyenne. Il transmet ensuite le résultat à l'application.

Conclusion

En conclusion, nous rappelons les éléments clefs de la contribution qui conduisent à leur tour à de nouvelles pistes de recherche.

1 Synthèse de la contribution

Alors que les concepts d'ordinateur évanescent et d'informatique ubiquitaire (ubiquitous computing) ne sont pas des idées nouvelles, ce n'est que récemment que les progrès de la technologie permettent d'en envisager la mise en œuvre. Il en résulte un foisonnement de recherches autour des *systèmes interactifs sensibles au contexte*. Avec l'hypothèse que ces systèmes présentent des avantages pour les activités humaines, cette étude a contribué à clarifier la situation en trois points complémentaires : définition, modèle, mise en œuvre :

1. Définition de la notion de contexte. Un contexte n'existe pas en tant que tel mais il convient de parler du *contexte d'une tâche T réalisée par un utilisateur U à un instant t*. Ceci précisé, un contexte est *un cumul de situations entre les instants t_0 et t intervenant dans la réalisation de T par U*. La *situation* à l'instant t est l'ensemble des variables périphériques à la tâche T mais susceptibles de l'influencer. Les *variables périphériques* se répartissent en attributs de l'environnement physique, de la plate-forme logicielle et interactionnelle, de l'environnement social et de l'utilisateur U considéré. Cette définition du contexte vaut :

- ?? selon plusieurs points de vue qui appellent la distinction entre le contexte brut, le contexte système, le contexte utilisateur et le contexte net (intersection des contextes système et utilisateur) ;
- ?? et selon les deux étapes essentielles du processus de développement d'un système (l'étape de conception/analyse et l'étape d'exécution,) qui amènent à distinguer les contextes système captable et capté, les contextes utilisateur utilisable et utilisé et les contextes net exploitable et exploité.

2. Proposition, avec le *contexteur*, d'un modèle conceptuel visant la mise en œuvre de systèmes sensibles au contexte. Un contexteur est une abstraction logicielle qui fournit la valeur d'une variable relevant du Contexte Système Capté. Inspiré des travaux de Salber, le contexteur présente comme originalité la notion de méta-donnée que je lie étroitement aux données d'entrée et de sortie. Comme chez Salber, le contexteur distingue les données de contrôle des données proprement dites. Au-delà des travaux de Salber, je propose :

- ?? Une *typologie* des contexteurs (contexteurs élémentaire, à seuil, à mémoire, de fusion, de traduction, d'extension),
- ?? Des *propriétés* de contexteur (réflexivité, rémanence, etc.),
- ?? La *composition* de contexteurs et la notion de chaîne de dépendance qui permet de raisonner sur la modifiabilité de la composition, enfin
- ?? L'*encapsulation* qui permet de considérer une composition de contexteurs comme un contexteur.

3. Validation du modèle conceptuel par la mise en œuvre d'un serveur de contexteurs et d'exemples simples d'applications sensibles au contexte ;

- ?? Le contrôle de la présence des tasses à café de cafétéria,
- ?? Le niveau d'activité de l'utilisateur sur sa station,
- ?? La présence de stations de travail sur un réseau local.

En résumé, cette étude a permis de poser un canevas général pour l'analyse et la mise en œuvre du problème des systèmes sensibles au contexte. Elle appelle de nombreuses questions.

2 Perspectives

Ce travail est imparfait.

Au niveau des concepts, il faut approfondir les notions d'environnement physique, social, logiciel introduites dans notre chapitre de définition du contexte. En particulier, des éléments complexes comme la localisation, l'épaisseur du temps, la granularité de l'espace posent de nombreuses questions.

Au niveau du « run time », il convient de :

- ?? Comparer l'approche adoptée avec les modèles fondés sur les agents cognitifs,
- ?? Etudier la représentation de l'incertitude,
- ?? Traiter le problème des ruptures dues aux pannes.

Au niveau de la conception logicielle, il faut étudier :

- ?? Les outils de spécification et alimenter notamment le processus de développement envisagé pour la plasticité des IHM.

Fournir des outils de conception et de mise en œuvre est une chose. Il faut aussi se préoccuper de leur bon emploi au regard des attentes utilisateur. Cet aspect motive à lui seul des travaux de recherche doctorale.

Références bibliographiques

[© 2001 Hachette Multimédia / Hachette Livre]
http://www.club-internet.fr/encyclopedie/

[©Webencyclo des Éditions Atlas 2001]
http://www.webencyclo.com/

[Abowd 98]
Gregory D. Abowd, Anind K. Dey, Robert J. Orr, and Jason Brotherton.
Context-awareness in Wearable and Ubiquitous Computing
Virtual Reality, Vol. 3 (1998), pp. 200-211.

[Abowd 00]
Towards a Better Understanding of Context and Context-Awareness
Workshop on The What, Who, Where, When, Why and How of Context-Awareness
Extended Abstract of the 2000 Conference on Human Factors in Computing Systems
(CHI'2000) G Szwillus & T Turner (eds), ACM Press, pp 371

[Avalon]
http://www.avalon.fr/faccueil.htm

[Bardram 97**]**
J. E. Bardram
Plans as Situated Action: An Activity Theory Approach to Workflow Systems
Proceedings of ECSCW'97 Conference, Lancaster UK, September 1997.

[Bérard 99]
François Bérard
Vision par ordinateur pour l'interaction fortement couplée.
Thèse de l'université Joseph Fourier.

[Beyer 98]
H. Beyer, K. Holtzblatt. *Contextual Design*.
Morgan Kaufman Publ. San Francisco, 1998.

[Brown 96]
P.J. Brown,
The Stick-e Document: a framework for creating Context-aware applications.
Electronic Publishing '96 (1996) 259-272

[Brown 97]
P.J. Brown, J.D. Bovey, X. Chen.
Context-Aware applications : From the Laboratory to the Marketplace.
IEEE Personal Communications, 4(5) (1997) 58-64

[Calvary 01a]
G. Calvary, J. Coutaz, D. Thevenin.
A Unifying Reference Framework for the Development of Plastic User Interfaces,
A paraître dans *EHCI'2001*.

[Calvary 01b]
G. Calvary, J. Coutaz, D. Thevenin.
Supporting Context Changes for Plastic User Interfaces.
a Process and a Mechanism, à paraître dans *HCI-IHM 2001*.

[Card 83]
Card, S.Moran, T.Newell,A.
The Psychology of Human-Computer Interaction,
Lawrence Erlbaum Associates,1983.

[Cheverst 01]
K. Cheverst, N. Davies, K. Mitchell, C. Efstratiou.
Using Context as a Crystal Ball : Rewards and Pitfalls.
Personal and Ubiquitous Computing, 5(1), Springer Verlag Publ., 2001, pp.8-11.

[Dey 99a]
An Architecture To Support Context-Aware Applications.
Anind K. Dey, Daniel Salber, Masayasu Futakawa and Gregory D. Abowd.
GVU Technical Report GIT-GVU-99-23. June 1999.

[Encyclopédie Larousse]

[Fano 01]
A. Fano.
What are a Location's File and Edit Menus ?
Personal and Ubiquitous Computing, 5(1), Springer verlag, 2001, pp. 12-15

[Gram 96]
C. Gram, G. Cockton
Design Principles for Interactive Software
Chapiter 3 : internal Properties: The Software developer's Perspective. pp .53-89

[Halverson 94]
Halverson, C.A.
Distributed Cognition as a theoretical framework for HCI: Don't throw the Baby out with the bathwater - the importance of the cursor in Air Traffic Control.
Tech Report No. 94-03, Dept. of Cognitive Science, University of California, San Diego. (1994)

[Nardi 96]
B. Nardi,
Context and Conciousness. Activity Theory and Human Computer Interaction,
MIT Press, Cambridge, Massachusetts, 1996.

[Pascoe 98]
J. Pascoe
Adding Generic Contextual Capabilities to Werable Computers.
2nd International Symposium on Wearable Computers (1998) 92-99

[Ryan 97]
N. Ryan, J. Pascoe, D. Morse,
Enhanced Reality Fieldwork : the Context-Aware Archeological Assistant.
V. Gaffney, M. van Leusen, S.
Exxon Computer Applications in Archeology (1997)

[Salber 99]
D. Salber, A.K. Dey, G. D. Abowd$
The Context Toolkit: Aiding the Develepment of Context-Enabled Applications
Proceedings CHI'99

[Salber01]
P. Gray, D. Salber
Modelling and Usins Sensed Context Information in the design of Interactive applications
EHCI 2001 (May 2001) 92-111

[Schilit 94]
B.N. Schilit, M. Theimer
Disseminating Active Map Information to Mobile Hosts, IEEE Network,
(1994) 22 – 32

[Schilit 94a]
B.N. Schilit, N.I. Adams and R. Want.
Context-Aware Computing Applications.
Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), IEEE Press, pp 85-90

[Schmidt 99]
Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven, Walter Van de Velde.
Advanced Interaction in Context.
1rst International Symposium on Handheld and Ubiquitous Computing (HUC99), Karlsruhe, Germany, 1999 & Lecture notes in computer science; Vol 1707, ISBN 3-540-66550-1; Springer, 1999, pp 89-101

[Schmidt 00]
Albrecht Schmidt.
Implicit Human Computer Interaction Through Context.

Personal Technologies Volume 4(2&3), June 2000. pp191-199.

[Schuman 87]
L. A. Schuman, Plans and situated actions. *The problem of human-machine communication*. Cambridge: Cambridge University Press. (1987)

[Thevenin 99]
D. Thevenin, J. Coutaz.
Plasticity of User Interfaces: Framework and Research Agenda.
In *Proceedings of INTERACT'99*, 1999, pp. 110-117.

[Ward 97]
Ward, A. Jones, A. Hopper,
A New Location Technique for the Active Office.
IEEE personnal Communications (1997) 42-47

[Weiser 93]
M. Weiser,
Some Computer Science Problems in Ubiquitous Computing,
Communications of the ACM, July 1993. (reprinted as "Ubiquitous Computing". *Nikkei Electronics*; December 6, 1993; pp. 137-143.)

[Yan 00]
H.Yan, T. Selker
Context – aware office Assistant
Proceedings of the 2000 International Conference on Intelligent User Interfaces (IUI'2000) ,H Lieberman (eds), ACM Press

Résumé

Cette étude a trait aux systèmes interactifs sensibles au contexte, c'est-à-dire aux systèmes capables d'identifier les circonstances qui entourent l'action de l'utilisateur en vue d'offrir des services adaptés. En réponse au foisonnement de propositions sur le sujet, cette étude présente trois contributions complémentaires servant la conception amont, la conception logicielle et la mise en œuvre :

- Une définition de la notion de contexte qui doit se comprendre comme un cumul de situations, une situation comprenant un ensemble de variables périphériques à la tâche utilisateur considérée mais susceptibles de l'influencer. On propose de distinguer le contexte système, du contexte utilisateur et leur intersection : le contexte net.
- Un modèle conceptuel visant la mise en œuvre de systèmes sensibles au contexte fondé sur la notion de contexteur. Un contexteur est une abstraction logicielle qui fournit la valeur d'une variable relevant du contexte système. Par composition de contexteurs, il est possible de construire une application sensible au contexte.
- Une validation du modèle par la mise en œuvre d'un serveur de contexteurs et d'exemples simples d'applications sensibles au contexte.