N° d'Ordre : HDR 2007 017

Habilitation à Diriger des Recherches

présentée devant
l'INSA de Lyon
et
l'Université Claude Bernard Lyon I

De l'adaptation à la prise en compte du contexte
Une contribution aux systèmes d'information pervasifs

par

Frédérique LAFOREST

Soutenue publiquement le 10 décembre 2007 devant le jury :

Pr. Corine Cauvet, LSIS, Aix-Marseille	Examinateur
Pr. Bruno Defude, INT, Evry	Rapporteur
Pr. André Flory, LIRIS, Lyon	Président du jury
Pr. Maurice Laville, CNRHL, Lyon	Examinateur
Pr. Hervé Martin, LSR-LIG, Grenoble	Rapporteur
Pr. Aris Ouksel, UIC, Chicago	Examinateur
Pr. Florence Sèdes, IRIT, Toulouse	Rapporteur

Laboratoire d'Informatique en Images et Systèmes d'information
LIRIS UMR CNRS 5205

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

L'apparition des terminaux mobiles intelligents a fait évoluer considérablement le rapport de l'humain à l'outil informatique. Les assistants personnels ont longtemps cherché leur voie, mais maintenant que leurs capacités commencent à être comparables à celles de véritables ordinateurs en modèles réduits, on imagine bien la place prépondérante qu'ils pourront prendre dans quelques années. Ce phénomène est plus flagrant quand on regarde la téléphonie : aujourd'hui, le téléphone mobile est pressenti comme le futur outil à tout faire : téléphoner bien sûr, capturer des données multimédia comme cela se fait déjà, mais aussi accéder à la télévision ou à tout type de flux multimédia, utiliser des applications dédiées et adaptées au lieu actuel et à la situation courante, payer par le biais de cartes bancaires virtuelles, se faire soigner plus efficacement par l'accès à son dossier médical... Bref, le terminal mobile est considéré comme un nouveau composant de l'architecture des systèmes d'information. Les nouveaux outils tels les assistants personnels de navigation (TomTom® ou équivalents), les *-box (FreeBox®, LiveBox®) ou autres terminaux (Playstation®, decodeurs satellite) sont autant de nouveaux types de systèmes informatiques qui aideront à développer de nouvelles formes d'utilisation de l'informatique dans la vie de tous les jours.

Ces nouvelles conditions d'usage, existantes ou à venir, forment le socle des systèmes d'information pervasifs. L'objectif premier d'un système pervasif est de permettre de répondre au besoin de l'utilisateur où qu'il se trouve et à n'importe quel moment. M. Satyanarayanan définit un système pervasif comme étant "one saturated with computing and communication capability, yet so gracefully integrated with users that it becomes 'a technology that disappears'" [Sat01]. Les systèmes pervasifs peuvent être vus comme étant l'étape suivante après les systèmes mobiles et les systèmes distribués. Les maîtres mots des systèmes pervasifs sont : sensibilité au contexte, smartness, scalabilité, invisibilité et pro-action. La sensibilité au contexte concerne la perception de l'environnement pour interagir plus "naturellement" avec l'utilisateur. Cette perception passe par l'utilisation de capteurs de l'environnement physique, de matériels auto-descriptifs, de description des personnes, etc. Le contexte fournit un grand éventail d'informations qui permet au système pervasif d'agir de façon adaptée. Cette sensibilité au contexte a pour but l'adaptation du système pervasif. La contrainte de smartness est difficile à traduire en un seul mot ; elle signifie à la fois l'intelligence

et la réactivité, mais aussi les bonnes manières ou le fair-play. L'objectif ici est de bien utiliser les informations de contexte pour décider de l'action la plus adéquate à la situation ; cela nécessite de définir des mécanismes d'adaptation. La scalabilité, ou passage à l'échelle, est nécessaire dans les systèmes pervasifs, car on peut rencontrer des cas où le nombre de composants entrant en jeu est très important. On pense notamment aux réseaux de capteurs. L'invisibilité est une contrainte forte ; il ne faut pas encombrer l'utilisateur avec des considérations qui ne le concernent pas, il faut qu'il se concentre sur la tâche à réaliser. L'adaptation doit être transparente. La pro-action a pour objectif de préparer le traitement d'une demande utilisateur avant même que cette demande ne soit explicite. Cela requiert une historisation des situations et actions, une analyse fine des situations et actions antérieures et leur rapprochement avec la session actuelle.

Les systèmes pervasifs soulèvent de nouvelles problématiques informatiques, que l'on peut classer dans deux niveaux. Les questions relevant du réseau constituent le premier niveau. Il faut définir de nouveaux protocoles capables de gérer les caractéristiques spécifiques des terminaux mobiles et non permanents ou des réseaux de capteurs, il faut gérer de nouvelles notions comme la qualité de service, ou encore permettre l'interconnexion de systèmes hétérogènes en termes de support du signal ou de protocole. Le second niveau, qui nous intéresse, concerne les données. Il faut permettre l'accès aux données dans tout type d'environnement et avec un rendu efficace dans le contexte de l'utilisateur, assurer le traitement de données de façon transparente et distribuée entre les terminaux mobiles et les infrastructures fixes, faciliter la production de données sur des mobiles et leur synchronisation avec des systèmes centraux ou distribués en intégrant éventuellement des outils collaboratifs. Toutes ces problématiques ont reçu des solutions dans des environnements fixes et potentiellement distribués, mais les contraintes de mobilité et de faiblesse des terminaux mobiles demandent de reconsidérer les solutions existantes. De plus, il est impossible de proposer des solutions qui satisfassent tout type de système pervasif. Il est nécessaire de définir des mécanismes d'adaptation.

L'adaptation des systèmes informatiques aux besoins des utilisateurs a été depuis plus de dix ans un axe majeur de recherche dans le domaine des systèmes d'information. Déjà lors de ma thèse de doctorat, j'ai abordé le problème de l'adaptation à destination des utilisateurs. Il faut permettre à l'utilisateur de recevoir une vue adaptée des informations dont il a besoin, de saisir les informations sous une forme qui lui convient, de rechercher l'information avec des outils intuitifs et efficaces. Le confort de l'utilisateur est l'objectif de nos propositions. Mais l'adaptation traitée ne s'arrête pas à la vue utilisateur ; en effet, cette vue provient de tout un processus de traitement et d'acheminement de l'information. L'adaptation doit donc également concerner les services fournis et l'envoi de l'information. Nous avons traité l'ensemble de ces points dans différents projets de recherche.

L'adaptation à l'utilisateur traite principalement des informations à présenter. Le traitement de l'information connaissait, jusqu'à il y a une dizaine d'années, deux mondes distincts : celui des données structurées avec les bases de données relationnelles principalement, et celui des documents avec le traitement

automatique des langues et l'indexation. L'objectif des données structurées est orienté vers le traitement et l'analyse de données. Tableaux de bord, statistiques, requêtes avec réponses exactes sont les maîtres mots de ce domaine. L'objectif des recherches en systèmes documentaires est différent : retrouver l'information bien sûr, mais en gérant des réponses inexactes, dans le sens où la précision et la pertinence sont définitivement les deux critères à optimiser (alors que leur valeur est considérée optimale dans les bases de données structurées). SGML a été une norme très importante pour la structuration et l'indexation des documents. Mais la complexité des algorithmes de traitement de tels documents a été un frein à son expansion.
L'arrivée du Web et de XML a changé la donne, en redéfinissant la frontière entre les deux mondes, et surtout en la rendant plus floue. Ainsi, un document XML peut servir à stocker des données structurées et est alors utile pour le dialogue machine-machine. On peut par exemple citer des flux SOAP qui servent à la communication entre services ou des documents CC/PP pour la description de terminaux et profils utilisateurs. XML est ici considéré comme un bon moyen de structurer un flux ou une configuration de manière explicite et ad hoc.
Mais XML permet aussi d'organiser des informations multimédia à destination de l'utilisateur final. On pense en premier lieu à l'utilisation conjointe de XML et des feuilles de style à la CSS pour un rendu satisfaisant des préférences utilisateur dans les pages web. Mais XML peut également servir à l'indexation de documents par l'utilisateur ou à la structuration de documents par le rédacteur. Alors que les éditeurs de texte classiques travaillent sur la mise en forme d'un document (rendu graphique), des éditeurs de document basés sur XML permettent de définir des balises sémantiques adaptées au type de document en cours de rédaction. Ces balises sémantiques peuvent alors être manipulées par l'utilisateur ou par un système automatisé pour faire du traitement sur le contenu des documents. On a ainsi assisté à la naissance des langages de requêtes sur documents XML mêlant des critères sur le contenu et sur la structure.
Dans nos travaux, nous avons utilisé XML aussi bien pour le dialogue machine-machine que pour l'interaction utilisateur. Notre apport recherche dans l'utilisation de XML est orientée vers l'interaction utilisateur. Nous considérons que la structuration de l'information n'est pas une problématique utilisateur, mais une problématique informatique pour le traitement exact de l'information. Nous avons donc voulu décharger l'utilisateur des contraintes associées. Toujours dans des soucis d'adaptation et de confort utilisateur, XML permet une rédaction libre de l'information en alternative à une saisie de données structurées. Une partie de mes travaux de recherche ont donc cherché à combler le fossé entre la rédaction libre et le traitement exact de l'information.

Dans un premier temps, j'ai travaillé sur des systèmes d'information dits génériques dont le but est d'adapter la présentation et la saisie des informations à chaque unité (services, entreprises) d'un même domaine d'application ou même à chaque utilisateur, en se fondant sur des interfaces documentaires. Un système générique factorise les éléments communs à toutes les unités d'un même domaine d'application. On effectue ensuite une adaptation pour chaque unité à l'aide d'opérateurs de dérivation. Des informaticiens ont la charge de développer le système générique. Un nouveau type d'acteur est inséré dans le processus de développement du logiciel : l'utilisateur-concepteur. C'est un expert du domaine concerné qui connaît bien son unité ; il a la charge d'effectuer

l'adaptation du système générique à son unité propre. Ces premiers travaux de recherche ont voulu apporter une solution originale à l'adaptation à l'utilisateur.

Pour poursuivre sur l'adaptation à l'utilisateur, nous nous sommes intéressés aux documents comme paradigmes d'interaction avec l'utilisateur. Cette approche nous a permis de ne plus restreindre le document à un seul outil de consultation de l'information. Nous voulons lui faire revêtir toutes les facettes d'une interaction utilisateur. Nous proposons une *saisie* d'information sous forme de documents faiblement structurés. Associés à une extraction de termes pertinents, ces documents ont permis de s'abstraire des formulaires de saisie. L'*interrogation* d'informations sous la forme de requêtes par documents-exemples a fourni une méthode alternative de recherche d'informations. La *consultation* d'informations sous forme de documents est plus classique ; nous avons cependant proposé d'associer tableaux de données résultats et documents résultats en une même réponse. On peut trouver de nombreux travaux traitant de l'accès à l'information sous forme de documents, mais peu de travaux considèrent les documents rédigés comme des sources de données, ou même comme des requêtes pour la sélection d'informations.
Dans le projet QDE, nous avons étudié l'interrogation par l'exemple de documents semi-structurés, en se fondant sur des représentations arborescentes des documents et des requêtes. Cette forme de recherche d'information, si elle permet la sélection de documents, ne permet pas une manipulation aisée des informations ou données trouvées. C'est pourquoi nous avons poursuivi nos investigations dans le sens d'une extraction de données enfouies dans les documents, ce qui fait l'objet du projet DRUID. Les documents sources contiennent des portions de texte rédigées dans une forme plus ou moins standardisée. Nous utilisons la structure arborescente du document et les connaissances du domaine pour effectuer une extraction des données pertinentes. Ces données alimentent une base de données structurée et permettent ainsi d'utiliser des techniques d'interrogation à la SQL.
Ensuite, DRUID2 nous a permis de définir une extension à DRUID pour le partage d'informations. A partir des grammaires des documents, nous définissons un modèle de données pour chaque source. Pour interroger de façon uniforme toutes les sources, nous créons ensuite semi-automatiquement un schéma médiateur, et nous notons les correspondances entre chaque source et le schéma médiateur. Les requêtes sur le schéma médiateur sont traduites grâce aux correspondances en des requêtes sur chaque source. Une consolidation de toutes les réponses obtenues fournit la réponse complète à la requête ; cette réponse associe données structurées et documents sources.
Les projets QDE, DRUID et DRUID2 font l'objet de la première partie de mon mémoire, intitulée "Consultation et saisie adaptées de documents".

Qui dit partage d'information dit également diversité des utilisateurs et des terminaux. Le système d'information doit donc pouvoir s'adapter à la situation courante, tant dans sa forme que dans son contenu. Nous avons commencé par traiter le niveau des interfaces utilisateurs. La diversité des utilisateurs et des terminaux a plusieurs implications : premièrement, les données et les fonctionnalités proposées doivent être adaptées au métier de l'utilisateur ; deuxièmement, les préférences d'un utilisateur diffèrent de celles de son collègue et la mise en forme des interfaces graphiques doit être personnalisée ; troisièmement, les

utilisateurs ont des terminaux personnels divers et les interfaces graphiques proposées doivent être conformes aux possibilités du terminal disponible, et plus généralement aux capacités de l'environnement physique disponible. La seule adaptation des interfaces utilisateurs ne permet pas de résoudre toutes les problématiques liées au contexte de l'utilisateur. Il est important de prendre en compte également les données échangées, et même les services du système d'information qui ont fourni ces données.
Dans le cadre du projet SEFAGI, nous avons abordé l'adaptation d'interfaces graphiques à des terminaux multiples, ainsi que la personnalisation des interfaces. A l'aide d'un assistant, les utilisateurs finaux décrivent eux-mêmes le contenu des fenêtres qu'ils désirent. Cette description est abstraite, c'est-à-dire indépendante de toute plate-forme. Notre générateur construit ensuite entièrement automatiquement le code des fenêtres correspondantes, et ce pour trois types de plates-formes : PC standards, assistants personnels et téléphones mobiles. Outre les descriptions des fenêtres générées par l'assistant, le générateur utilise des documents de description des plates-formes cibles.
Enfin, le contexte est devenu de plus en plus prépondérant, les applications sont devenues de plus en plus réparties, et nous avons donc abordé l'adaptation dans le cadre plus général de la sensibilité au contexte dans les systèmes pervasifs. Le projet SECAS propose une méthode et une plate-forme pour l'adaptation d'applications legacy. SECAS construit une couche d'auto-adaptation au-dessus de l'application existante. L'adaptation concerne à la fois les interfaces graphiques et le contenu, mais également les services eux-mêmes. SECAS comporte deux autres couches : une couche de description de l'application à adapter et une couche de gestion du contexte.
Les projets SEFAGI et SECAS constituent la seconde partie de mon mémoire, intitulée "Adaptation au contexte".

Le lecteur remarquera aisément que nos propositions sont toujours étayées d'exemples relevant du domaine médical. C'est en effet un domaine d'application que je privilégie tout particulièrement, qui me permet pemièrement de valider par l'exemple et la pratique les propositions théoriques que nous élaborons. Mais ce domaine est également tellement riche en besoins qu'il est l'instigateur de thématiques de recherche théorique. Par exemple, c'est l'informatisation du dossier médical qui a lancé nos travaux sur la saisie et la consultation de documents ; ce sont les problématiques de terrain de l'hospitalisation à domicile qui ont lancé nos recherches en adaptation au contexte. La section 4.1 étaye ces propos.

Les deux prochaines parties de ce mémoire présentent les différents travaux que nous avons menés, premièrement pour la saisie et la consultation adaptées de documents, et ensuite pour l'adaptation au contexte. Chaque partie fournit ensuite un rapide positionnement par rapport à l'état de l'art. Elle présente également les points forts et indique les travaux qu'il reste à mener. Elle se termine par une vue synthétique des thèmes et sous-thèmes abordés, des encadrements effectués, des financements ayant soutenu ces travaux, et finalement des publications. Pour conclure, nous expliquons l'intérêt que nous avons trouvé à appliquer nos propositions aux systèmes d'informations médicaux, et nous traçons des perspectives situées dans le contexte actuel. Finalement, après un chapitre dédié à mon curriculum vitae, une sélection d'articles balaie l'ensemble des projets de recherche menés.

Chapitre 2

Consultation et saisie adaptées de documents

2.1 Introduction

L'informatique a toujours eu la propention à uniformiser les choses. Ainsi, les formulaires sont devenus un standard pour la saisie de données, et très peu d'applications dérogent à leur utilisation. Pourtant, il y a des cas où l'utilisation d'autres artefacts serait bien plus efficace. Par exemple, dans les domaines d'expertise peu ou pas encore complètement normalisés comme la médecine et plus particulièrement le dossier médical, le document est un outil très prisé [JCB98]. Ces documents ne doivent pas non plus se réduire à leur première conception [Bri06] : il faut pouvoir y adjoindre des annotations, l'organiser un tant soit peu, travailler sur les données qui y sont saisies...
Durant ma thèse, un premier travail sur les interfaces utilisateurs à base de documents a été effectué. Nous avons utilisé la notion de document virtuel, permettant la consultation d'informations sous forme de documents, construits à la demande en fonction des besoins de l'utilisateur et des informations présentes dans une base de données sous-jacente. Le dossier médical était alors vu comme un ensemble de documents hétéroclytes à structurer et organiser.
Les médecins et autres acteurs du système de santé doivent déposer dans le dossier médical des traces de leurs observations, analyses et de leur plan d'action. De nombreuses tentatives de dossiers médicaux électroniques ont vu le jour, avec plus ou moins de succès. Les capacités de distribution des informations (transmission automatique aux acteurs concernés, connexion avec des systèmes de prise de rendez-vous...), de traitement de l'information automatisée (analyse de la prescription médicamenteuse vérifiant contre-indications et effets indésirables...), ou de synthèse (courbes de température, tableaux d'administration de médicaments...) sont grandement appréciées. Les formulaires sont par contre très critiqués : pas pratiques, trop standardisés, ne permettant pas de prendre en compte les cas particuliers... Paradoxalement les critiques restent souvent vives quant au traitement des parties textuelles. En effet, soit les informations sont saisies dans des formulaires structurés trop rigides et sont alors exploitables par les outils précités, soit on fournit des outils de saisie de texte libre dont le contenu n'est pas exploitable par ces mêmes outils. Nous avons donc voulu étu-

dier comment réconcilier la saisie "libre" des informations textuelles avec les autres fonctionnalités d'un système informatisé.

Les documents saisis par les professionnels de santé ont des formes diverses et des structures changeantes. Cependant, on peut trouver des récurrences dans les structures des documents d'une spécialité et d'une profession. Ainsi, par exemple, les dossiers médicaux de néo-natologie rédigés par les médecins comportent toujours les mêmes sections (examen neurologique, électro-encéphalogramme, prescription, etc.). Le texte contenu dans chaque paragraphe est rédigé de façon libre, mais des formules "standards" peuvent être dégagées comme par exemple "Le <examen> de contrôle du <date> montre <symptome>". Ainsi, nous avons choisi de proposer un compromis: les documents saisis par les utilisateurs ne sont pas complètement libres; ils se composent de paragraphes. Chaque paragraphe est repéré et typé (balisé puisque ce sont des documents XML), et contient du texte librement rédigé. Les formes récurrentes de rédaction permettent de définir des motifs pour l'extraction des informations. Les types de paragraphes disponibles sont définis par le type du document. L'utilisateur peut alors "piocher" dans la liste des types de paragraphes disponibles pour construire son document. Par exemple, un médecin voulant rédiger un compte-rendu d'examen clinique choisit de créer un document de type "CR examen clinique". Ce type de document inclut les types de paragraphes "symptome", "signe clinique", "prescription medicamenteuse", "prescription d'acte", "diagnostic"...
On pourra également noter que le document ne contient que les paragraphes utiles, à l'opposé des formulaires qui ne peuvent retirer des zones non remplies parce que non pertinentes dans le cas présent. On remarque aussi que la sémantique de l'information est très dépendante de la structure qui l'inclut. Ainsi, un nom de médicament dans un paragraphe de prescription n'a pas la même signification que le même nom dans un paragraphe d'allergies. Donc la recherche d'information doit prendre en compte le contexte des informations recherchées.
Les documents rédigés ne sont donc ni des document libres, ni des documents structurés au sens commun du terme. En effet, un document structuré comporte des balises qui repèrent chaque donnée unitaire. Or, dans les documents rédigés, un seul paragraphe prescription peut "enfouir" plusieurs données (dosage, nom médicament, durée, fréquence). Nous appelons ces documents des *documents faiblement structurés*, par opposition aux *documents centrés données* habituellement utilisés.
En première instance, nous avons d'abord pensé à adapter la recherche d'informations à ce type de documents. Nous avons proposé d'effectuer des requêtes par l'exemple: les réponses sont calculées en utilisant des techniques d'appariement d'arbres. Ceci fait l'objet de la section 2.2.
En suivant une autre direction, nous nous sommes attachés à définir un outil d'extraction des informations unitaires à l'intérieur de chaque paragraphe saisi. Le type du paragraphe dirige l'extraction de données: on sait quel type de données chercher; parfois même, on connait des formes types de rédaction. La consultation des informations est alors mixte: une recherche à la SQL dans les données extraites permet non seulement la visualisation standard des données, mais également un accès au document rédigé, et donc à des informations, détails ou nuances non exprimables dans la base de données. Un des objectifs du projet DRUID a donc été de définir une méthode d'extraction des données dans des

documents faiblement structurés. Cette extraction permet de construire automatiquement un document centré données. Ceci fait l'objet de la section 2.3. D'autre part, la médecine est par essence et de plus en plus répartie. La montée en puissance des réseaux de soin et de l'hospitalisation à domicile requiert la création et la consultation à distance de documents produits par des utilisateurs divers provenant de diverses institutions. Le partage d'information doit alors se faire sur des schémas multiples, et l'on doit s'intéresser à l'intégration de schémas pour le partage de données et de documents d'origines multiples. Il a également fallu construire les modèles de documents à partir des modèles de données. Ceci fait enfin l'objet de la section 2.4. Nous concluons ensuite ce chapitre par un positionnement par rapport à l'état de l'art, une synthèse des apports et des idées pour des travaux futurs. En tout dernier lieu, nous présentons une vue synthétique des actions que nous avons menées sur cette thématique, incluant notamment les participants, les publications et les financements associés.

2.2 QDE : Requêtes sur documents par l'exemple

2.2.1 Structure des documents

Dans ce travail intitulé QDE pour Querying Documents by Example, un document contient quatre "niveaux" de composants :

- des données structurées, constituées la plupart du temps de références à des tables de bases de données, comme l'identifiant d'un médicament par exemple. Ces données sont balisées par les identifiants nécessaires (clé de l'occurrence dans le cas d'un fichier indexé),
- des informations en texte libre, saisies directement par l'utilisateur final,
- des types de paragraphes, dont la liste est définie dans la grammaire du document (DTD) et qui permettent de valider le document,
- des balises sémantiques, disponibles indépendamment de la DTD au sens strict, que l'utilisateur peut placer à son gré autour d'expressions de texte libre. Par exemple, la balise <hypothèse> peut être mise par le médecin autour d'un nom de maladie, indiquant ainsi que ce diagnostic n'est pas encore confirmé. Les balises sémantiques sont posées librement, sans aucune contrainte dans la DTD du document.

Un document peut alors être vu comme un arbre à quatre niveaux : le niveau 1 est la racine de l'arbre, le niveau 2 est constitué des types de paragraphes, le niveau 3 comporte les éventuelles balises sémantiques, et le niveau 4 se réfère aux données et informations. La figure 2.1 donne un exemple de document.

2.2.2 Requêtes

Les requêtes posées au système gérant de tels documents sont rédigées sous forme de documents exemples. Ainsi, la requête suivante peut être posée : "Donner les dossiers dont le patient a indiqué une allergie au médicament A soulagée par le traitement B, et où des investigations supplémentaires ont montré une allergie à C." Le document et sa représentation sous forme d'arbre sont donnés dans la figure 2.2.

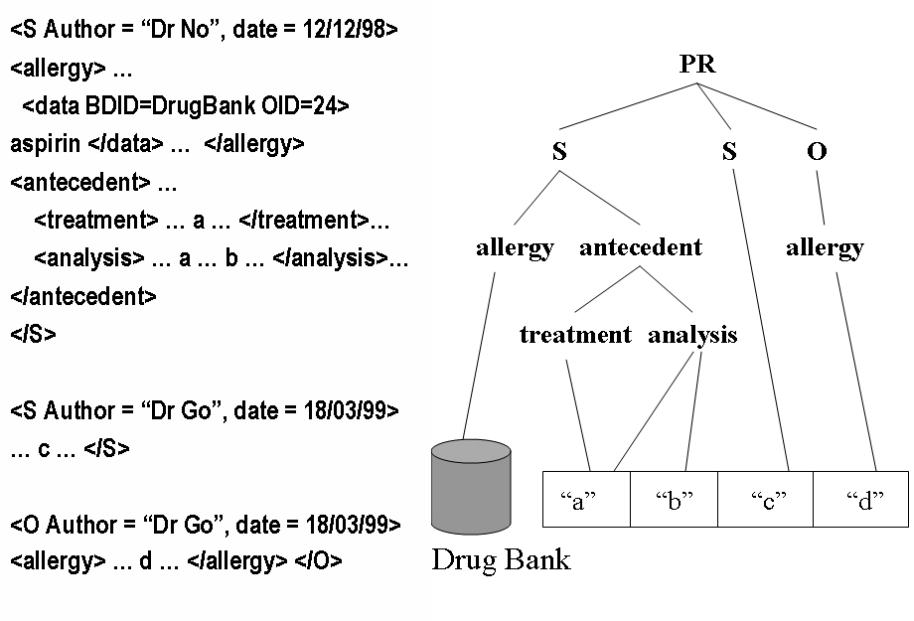


FIG. 2.1 – *QDE : Représentation arborescente d'un document*

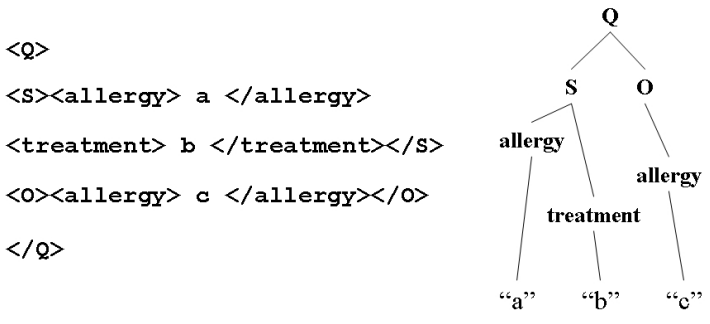


FIG. 2.2 – *QDE : Représentation arborescente d'une requête*

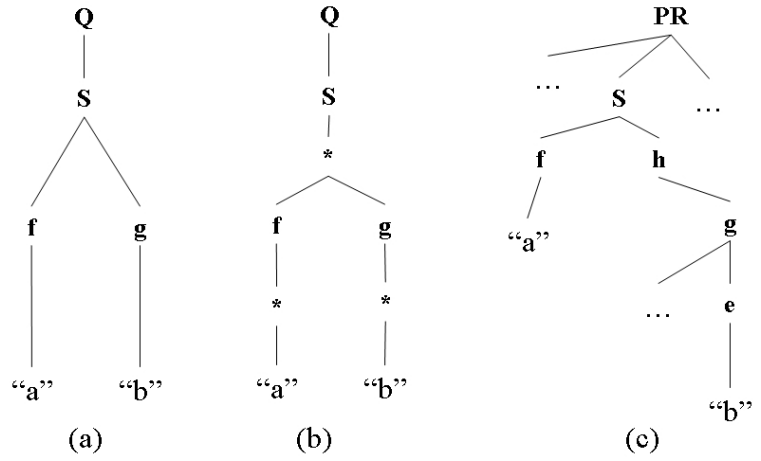


FIG. 2.3 – *QDE : Patron de requête*

Les réponses à une requête doivent contenir les types de paragraphes de la requête, les données de la requête, et les informations de la requête encadrées des mêmes balises sémantiques. Les documents pertinents peuvent également inclure d'autres paragraphes, données, informations ou balises sémantiques, et ce de façon intercalée avec les composants de la requête. Par exemple, dans la figure 2.3, le document dessiné en (c) doit être fourni en réponse à la requête (a). Pour prendre en compte cela, nous ajoutons à l'arbre d'une requête des noeuds virtuels représentant le fait qu'un élément recherché peut être dans une profondeur ou largeur variable. L'arbre dessiné en (b) est la requête (a) augmentée de ses noeuds virtuels. Chaque arbre est transformé en une chaîne de caractères par un parcours en profondeur d'abord. L'évaluation de la requête s'apparente alors à un appariement d'expressions régulières.

2.2.3 Conclusion "Requêtes par l'exemple"

Ce travail, effectué conjointement avec Anne Tchounikine, a fait l'objet de deux publications : [13] et [28]. Cette forme de recherche d'information, si elle permet la sélection de documents, ne permet pas une manipulation aisée des informations ou données trouvées. C'est pourquoi nous avons poursuivi nos investigations dans le sens d'une extraction des données enfouies dans les documents.

2.3 DRUID : Du document aux données

Dans la suite de notre travail, nous avons voulu nous attacher plus à la méthode de travail des utilisateurs finaux, et nous avons choisi de nous intéresser plus particulièrement à l'extraction de données "enfouies" dans des paragraphes de texte libre. C'est l'objet du projet Documents and Rules-based User Interface to Databases (DRUID). La figure 2.4 présente un exemple d'interface de saisie de documents. Le document est rédigé librement. Chaque paragraphe doit être

FIG. 2.4 – *DRUID : Prototype d'interface de saisie de documents dans DRUID*

balisé par l'utilisateur final. L'extraction de données dans les documents saisis est ensuite traitée comme une transformation d'un *document faiblement structuré* en un *document centré données*.

2.3.1 Principe de la transformation de document

Dans DRUID, le passage d'un document rédigé à des données consiste à *transformer le document faiblement structuré en un document centré données*. Cette transformation a lieu comme indiqué en figure 2.5. Nous avons utilisé la technologie XSLT pour implanter une telle transformation. A chaque type de document est associée une feuille de transformation, qui contient l'ensemble des règles concernant ce type de document. Les quatre étapes utilisées sont les suivantes :

- La *sélection* permet de prendre certains paragraphes dans les documents rédigés. Ceci se fait tout simplement par l'utilisation de règles XPATH.
- L'*extraction* permet d'identifier les données "enfouies" dans les paragraphes. Cette étape est de loin la plus complexe. En effet, elle requiert l'utilisation d'un processeur de la langue naturelle (NLP) et donc la définition de dictionnaires de termes et de règles d'extraction dans le format du NLP. Le format des règles est complexe, et plutôt que de demander leur rédaction par un utilisateur final, nous avons défini un langage de description de motifs et un compilateur de ces motifs utilisateur en des règles pour le NLP.
- Le *traitement* permet d'appliquer des opérations sur les données extraites (changement d'unité, opérations arithmétiques, manipulation de chaines...). Cette étape utilise des fonctions externes pour effectuer les traitements sur les données extraites.
- La *restructuration* permet de construire un document centré données conforme à la structure attendue. Nous avons pour cela étendu un processeur XSLT standard en y ajoutant l'élément *construct* qui permet soit de mettre dans le document centré données uniquement les données extraites, soit

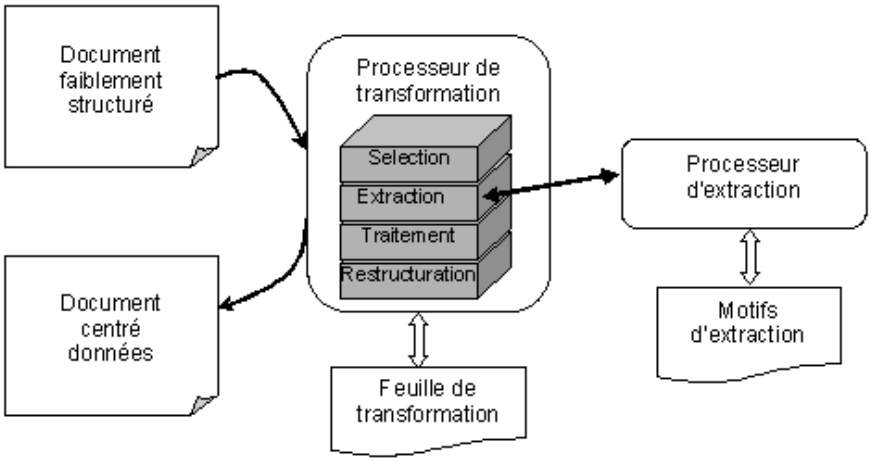


FIG. 2.5 – *DRUID : Architecture de transformation de DRUID*

de conserver l'intégralité du texte du paragraphe en y ajoutant des balises repérant les données extraites.

La transformation complète du document donne lieu à des documents centrés données, qui peuvent ensuite être utilisés pour alimenter une base de données XML ou relationnelle. Il existe de nombreux travaux dans le domaine de l'alimentation de bases de données relationnelles à partir de documents XML centrés données, et nous avons repris l'un de ces travaux (XML-DBMS [BBB00]).

2.3.2 Extraction des informations

L'étape d'extraction des informations mérite notre attention. Nous avons choisi de réutiliser le moteur de traitement de la langue naturelle (NLP) Intex [Sil00] de l'université de Franche-Comté. Notre processeur d'extraction est une interface entre le processeur de transformations et le NLP. Il reçoit un paragraphe rédigé en langue naturelle et doit retourner des termes extraits du paragraphe grâce à l'application d'un motif d'extraction. Avant de faire effectuer l'extraction par le NLP, le processeur d'extraction lance des étapes préliminaires : il demande au NLP de faire (i) une validation du texte (que du texte et pas de caractères de contrôle), (ii) un pré-traitement pour identifier les phrases, délimiter les mots composés non ambigus, et marquer les mots contractés ou élidés, (iii) une analyse lexicale pour appliquer les dictionnaires de la langue et du domaine, et (iv) une désambiguïsation des mots correspondant à plusieurs entrées lexicales dans les dictionnaires. Après ces étapes préliminaires, le processeur d'extraction fournit au NLP le paragraphe à analyser et le motif d'extraction correspondant pour effectuer l'extraction proprement dite.
Les motifs d'extraction sont rédigés par un expert du domaine. Pour aider à la gestion de ces motifs, nous avons défini un langage de spécification à quatre niveaux que nous détaillons ci-après. Les motifs d'extraction sont très dépendants du domaine d'application, et nécessitent un travail préalable non négligeable. Ils intègrent des références à des dictionnaires de la langue française ainsi qu'à des dictionnaires de termes spécifiques au domaine. Ces motifs ne sont pas directe-

```
01 expression expDosage1= dosage [<NB> "/" <NB>] <unité> ;
02 expression expDosage2= dosage [<NB> "et" <NB> "/" <NB>] <unité> ;
03 expression expDosage3= dosage [<NB>] <unite> ;
04 expression expuDosage1= <NB> "/" <NB> unitéDosage[<unité>];
05 expression expuDosage2= <NB> unitéDosage[<unité>] ;

06 slot dosage= expDosage1 | expDosage2 | expDosage3 ;
07 slot unitéDosage = expuDosage1| expuDosage2;
08 slot médicament = médicament[<medical>];
09 slot durée = 'pendant' durée [<NB>] <période> ;
10 slot unitéDurée = 'pendant' <NB> unitéDurée[<période>] ;
   ...
11 pattern prescription = dosage, médicament, durée?, unitéDurée?, fréquence? ;
```

FIG. 2.6 – *DRUID : Exemple de motifs rédigés par un utilisateur*

ment compréhensibles par le processeur de langue naturelle, mais doivent être transformées en transducteurs à états finis (FST). Il est difficile pour un humain de rédiger des FSTs pour des cas complexes, et le nombre de FST nécessaires est important. C'est pourquoi nous avons préféré définir un langage de spécification de plus haut niveau. Ce langage de spécification permet à l'utilisateur d'écrire des motifs d'extraction. Nous avons développé un compilateur permettant de traduire les motifs en FST.

2.3.3 Motifs utilisateur et règles

Le langage de haut niveau que nous avons défini permet à l'utilisateur d'écrire simplement des motifs d'extraction. En regard des classiques expressions régulières, ce langage ajoute les fonctionnalités suivantes :

- définir des méta-mots comme <maladie> pour faire référence à des dictionnaires de termes,
- faire référence à un motif existant dans la définition d'un nouveau motif,
- rédiger en plusieurs couches les motifs, et donc fournir un résultat plus lisible à l'utilisateur qui rédige ou réutilise des motifs.

Pour la rédaction des motifs d'extraction, nous avons défini quatre couches :

- La couche 1 est composée d'un ensemble fini de termes. Un terme est soit une séquence de lettres délimitée par des séparateurs, soit un méta-mot décrivant un nombre <NB>, un mot <MOT>, un mot vide <E> ou une référence à un dictionnaire <dico>. Un terme peut être nommé, c'est-à-dire que le terme identifié doit être retourné balisé par le nom donné : medic [<medicamentDic>] indique que le terme du dictionnaire medicamentDic doit être balisé par le mot medic.
- La couche 2 est composée d'un ensemble fini d'expressions. Une expression est une concaténation de termes séparés par des espaces ou des tabulations. On dit qu'une expression est valide dans un paragraphe si on y trouve une séquence de termes correspondant à cette expression.
- La couche 3 est constituée d'un ensemble fini de segments. Un segment est une liste d'expressions alternatives, exprimant les différentes formes de rédaction d'un tel segment. On dit qu'un segment est valide dans un paragraphe si une de ses expressions est valide dans ce paragraphe.
- La couche 4 est constituée d'un ensemble fini de motifs. Un motif est un ensemble fini non ordonné de segments. Certains segments sont obligatoires, d'autres sont optionnels. Des contraintes sur les segments peuvent

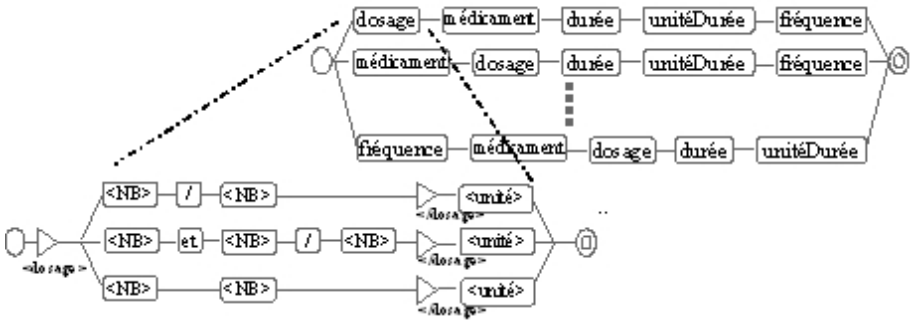


FIG. 2.7 – *DRUID : FST construit par notre compilateur de motifs*

être indiquées, pour réduire le nombre factoriel de combinaisons possibles (cardinalité, ordre entre n segments...). On dit qu'une règle est valide pour un paragraphe si l'ensemble de ses segments obligatoires est valide dans ce paragraphe.

Par exemple, pour définir un motif d'extraction de prescription (figure 2.6), on définit un motif (pattern, ligne 11) indiquant qu'un paragraphe de prescription contient les segments (slots, lignes 6 à 10) dosage, médicament, et optionnellement les segments fréquence, durée et unité de durée, une ou plusieurs expressions (expression, lignes 1 à 5) indiquant la définition de chaque segment avec ses différentes formes d'écriture possibles. Le segment dosage indique qu'il y a trois expressions possibles pour repérer le dosage tel que '$\frac{1}{2}$ comprimé', '1 cachet', '1 $\frac{1}{2}$ gélule' etc. Les termes comprimé, cachet et gélule sont recensés dans un dictionnaire et repérés par le meta-mot <unité>. De même, le meta-mot <période> regroupe les termes tels que jour, semaine, mois, etc.

Le NLP que nous avons choisi pour l'implantation de notre système utilise les transducteurs à états finis (FST) comme format d'entrée des motifs d'extraction. Ces FST sont abrupts à lire pour des cas non triviaux, et c'est pourquoi nous avons préféré fournir à l'utilisateur concepteur un langage de haut niveau pour la rédaction des motifs. La traduction de motifs en FST nécessite l'utilisation d'un compilateur. Nous avons développé un tel compilateur. Il prend en entrée des motifs dans le langage de spécification et retourne en sortie les mêmes motifs traduits en FST. Ce compilateur suit les étapes classiques d'analyse lexicale, syntaxique et sémantique. Il lit les scripts contenant les spécifications des motifs et compile chaque motif d'extraction en un FST en utilisant une procédure récursive bottom up et ceci en incluant à chaque couche les FSTs de couche inférieure : un FST est écrit pour chaque terme, puis pour chaque expression (les FSTs des termes inclus sont mis en série), puis pour chaque segment (les FSTs des expressions correspondantes sont mis en parallèle), et enfin pour chaque motif (on met en parallèle tous les ordres possibles de segments). La figure 2.7 présente le FST correspondant à la traduction du segment dosage rédigé dans notre langage de spécification à quatre couches.

2.3.4 Conclusion "Du document aux données"

Ce travail a fait l'objet de plusieurs stages de DEA, master recherche ou projets de fin d'études ainsi que de la thèse de doctorat de Youakim Badr [Bad03], et a donné lieu à douze publications : un article dans une revue internationale [4], deux articles dans des revues nationales [16, 17], deux chapitres de livres [29, 30], quatre articles dans des conférences internationales à comité de programme [9, 10, 11, 12], deux articles dans des conférences nationales ayant donné lieu à des chapitres de livres [22, 23] et un article dans un workshop VLDB [27]. Nous avons également fait une présentation de ces idées dans le GT Documents multimédia du GDR I3 en juin 2001 [46].

Deux prototypes ont vu le jour dans ce projet :

- le compilateur de motifs qui permet de transformer des motifs en FST,
- le processeur de transformation qui permet, en association avec le logiciel Intex, de construire des documents centrés données à partir de documents faiblement structurés.

Ces prototypes ont été écrits dans le langage java.

Les projets TIPHAD (ACI 99-01), COQUAS (RNTS 01-04) et DocPatient (HTSC Haute Picardie 02-05) nous ont permis de collaborer avec des médecins et d'avoir ainsi accès à des corpus de documents, pour tester le travail effectué. Nous remercions ici les médecins qui ont accepté de nous fournir ces corpus et qui ont pris le temps d'anonymiser les documents fournis. Nous avons eu accès à deux corpus :

- des résumés de séjour en cardiologie et pneumologie (une quarantaine),
- des dossiers de néo-natologie (une dizaine).

Ces documents étaient sous forme de documents Word ou de documents papier. Nous avons en premier lieu défini les DTD des documents centrés données et faiblement structurés correspondant au domaine. Les documents fournis ont ensuite été créés sous la forme de documents faiblement structurés. Nous avons également élaboré les dictionnaires du domaine, avant de rédiger les motifs d'extraction. A l'aide de notre compilateur ERel, nous avons pu obtenir les FST correspondants. Nous avons ensuite introduit les documents dans DRUID pour obtenir des documents centrés données.

Il est difficile de fournir un pourcentage de réussite pour l'extraction. En effet, le succès d'un tel processus dépend de beaucoup de facteurs. Il est par exemple difficile de construire des dictionnaires complets, et cela influence bien sûr la qualité de l'extraction. D'autre part, la régularité effective des styles rédactionnels influence la complexité et le nombre de motifs à écrire : plus on requiert de complétude dans l'extraction, plus le nombre de motifs à rédiger est important. Le degré de détail du modèle de données est encore un autre facteur : plus le modèle de données entre dans les détails, plus le nombre d'informations à extraire est grand et plus la tâche est difficile en regard de la variabilité des rédactions.

2.4 DRUID2 : Partage de données et documents

Le dossier médical est un exemple concret de répartition de données ; le parcours de soin d'un patient engendre une répartition des informations le concernant : chaque lieu où le patient a été pris en charge constitue une source de

données, et la vue complète de l'histoire médicale du patient ne peut être faite que par l'intégration des données provenant de ces différentes sources. De plus, la règle d'or dans le domaine médical est que chaque information est conservée à sa source, et on ne peut donc pas imaginer avoir un entrepôt de données centralisé qui contiendrait une version intégrée des informations. Les actions actuelles mettent en place des serveurs régionaux qui constituent pour chaque patient un index des éléments répartis dans chaque source (voir par exemple la plate-forme de la région Rhône-Alpes (http://www.sante-ra.fr/) ou de la région Lombardie (http://www.crs.lombardia.it/)). Le dossier médical constitue d'ailleurs un domaine d'application privilégié des recherches en grilles de données [SPB05]. De plus, si de nombreux standards ont été mis en place internationalement (par exemple HL7 ou CEN TC-251), les données restent peu structurées, faiblement structurées ou structurées de manière ad hoc. Notre travail concernant l'extraction de données dans des documents a eu pour objectif de construire des informations structurées à partir de sources faiblement structurées. Ici l'objectif a été de définir un moyen d'accéder de façon unifiée à un ensemble de sources réparties. Nous n'avons pas travaillé sur l'indexation des sources ni sur la découverte de sources de données, mais nous nous sommes intéressés à la construction d'une vue unifiée de données de formats divers, mais traitant d'un même domaine d'application. Nous traitons de l'intégration de schémas de données structurées (tables et attributs). Le partage entre utilisateurs s'étend par contre aux documents puisque les documents sont traités comme des cas particuliers d'attributs (url stockée dans la base de données). Fondée sur une ontologie du domaine, notre proposition fonctionne en deux temps. Tout d'abord nous décrivons le schéma des données de chaque source dans le "schéma" de l'ontologie du domaine et un schéma médiateur est construit. Ensuite des requêtes peuvent être posées sur le schéma médiateur. Des wrappers traduisent les requêtes sur le schéma médiateur en requêtes sur les sources, et mettent en forme les réponses dans la structure du schéma médiateur.

2.4.1 Une vue unifiée fondée sur XML

Notre proposition se fonde sur une ontologie dite *ontologie de structure* qui répertorie et organise les concepts du domaine et leurs caractéristiques. Ces concepts sont issus des vocabulaires utilisés dans les modèles de bases de données du domaine (tables et attributs). La figure 2.8 donne une représentation UML du schéma de notre ontologie de structure. Une ontologie de structure est stockée sous la forme d'un document XML-Schéma. Certains concepts de l'ontologie de structure peuvent être associés à des valeurs possibles. Par exemple, le concept Diagnostic peut être associé à la classification internationale des maladies (CIM-10). L'ensemble des valeurs possibles de tous les concepts de l'ontologie de structure forme une *ontologie de valeurs*.

L'intégration d'une nouvelle source de données requiert une étape préalable de description de son schéma dans les termes de l'ontologie de structure. Ceci donne lieu à un *document de description de la structure* de la nouvelle source de données. Chaque table et attribut de la source de données doit être associé à son équivalent dans l'ontologie de structure. Nous avons défini un algorithme de construction du document de description qui soit le plus automatique possible. Ainsi, pour chaque table de la source, la recherche d'un concept équivalent se

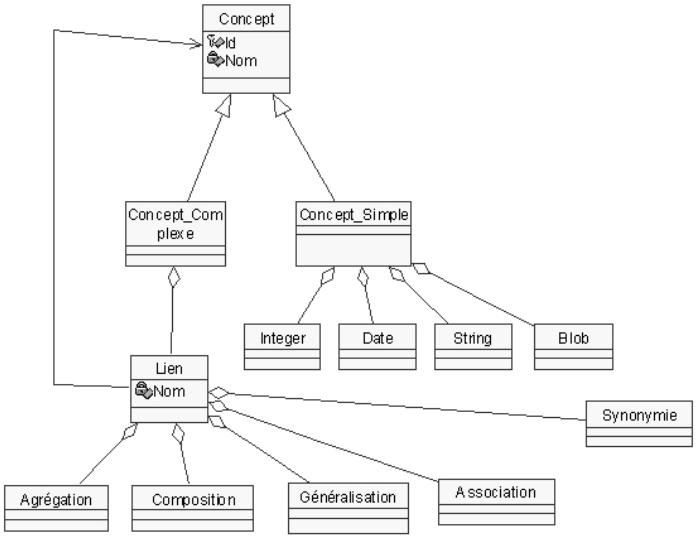


FIG. 2.8 – *DRUID2 : Diagramme de classes de l'ontologie de structure*

fonde sur l'ensemble des synonymes disponibles, sur l'équivalence entre les liens existants dans l'ontologie de structure et ceux décrits dans le schéma de la source, sur l'ontologie de valeurs... Nous avons ainsi défini un ensemble de seize règles que le processus de construction de la description doit suivre. Nous avons également conçu et développé un algorithme suivant ces règles. Les documents de description de la structure des sources sont conformes à la DTD donnée en figure 2.9. Nous avons également défini les étapes à suivre pour la prise en compte de modifications de structure dans les sources.

Le processus suivant concerne la construction du schéma médiateur et des informations de correspondance entre sources et schéma médiateur. Ce processus prend en entrée les documents de description de toutes les sources. Nous avons défini un ensemble de 15 règles permettant d'orienter l'algorithme de génération du schéma médiateur et des correspondances. Le schéma médiateur et les correspondances sont également stockés sous forme de documents XML.

2.4.2 Médiation de vues XML

L'utilisateur pose ses requêtes sur le schéma médiateur. Le médiateur complète la requête posée en y insérant des informations provenant de la structure de l'ontologie (notamment en exploitant les relations de synonymie et de généralisation). Chaque requête est ensuite traduite dans le langage de chaque source par le wrapper correspondant. Ces wrappers prennent en entrée les requêtes et les correspondances établies entre la source et le schéma médiateur. Les wrappers transforment les réponses fournies par les sources pour qu'elles soient conformes au schéma médiateur. Le médiateur construit une réponse par concaténation des réponses des différentes sources. Aucune synthèse ou union n'est effectuée sur les sources : il est important de connaître la provenance de chaque information, et qu'elle soit "mise en contexte".

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT schema (Nom-schema, Relation+)>
<!ELEMENT Nom-schema (#PCDATA)>
<!ELEMENT Relation (Nom-concept, description, CleP?, attribut+, References?)>
<!ATTLIST Relation nom-relation ID #REQUIRED>
<!ELEMENT Nom-concept (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT attribut (Nom-concept, description)>
<!ATTLIST attribut Nom-attribut ID #REQUIRED>
<!ELEMENT References ((IDrelation, CleP)+)>
<!ATTLIST References IDrelation IDREF #REQUIRED>
<!ELEMENT CleP (IDAttribut*)>
<!ATTLIST CleP IDREFS #REQUIRED>
```

FIG. 2.9 – *DRUID2 : DTD pour la description des sources de données*

2.4.3 Du modèle de la base de données aux modèles des documents

Les documents rédigés par les utilisateurs finaux permettent de décrire leurs activités de façon semi-libre, et également d'alimenter une base de données. Souvent, la base de données existe déjà dans l'organisation, et une interface utilisateur "classique" est fournie. Dans ce cas, notre système DRUID peut être vu comme une interface utilisateur alternative. Les données enfouies dans les documents saisis doivent permettre l'alimentation de la base de données existante. Nous avons donc élaboré un outil interactif qui permet de construire les modèles de documents en fonction du schéma de la base de données et de quelques choix de l'utilisateur.

En premier lieu, l'outil présente l'ensemble des tables disponibles dans le modèle, et l'utilisateur choisit celle qui constitue le thème central du type de document (e.g. Patient pour la fiche administrative patient). Notre outil construit alors une DTD représentant la structure de la base de données. Les étapes principales de cet algorithme sont les suivantes :

- la table choisie devient la balise principale du document,
- ses attributs clés deviennent des attributs de la balise correspondante,
- les tables référencées par ses attributs clés étrangères deviennent des balises filles de la balise principale et incluent un attribut par clé. L'utilisateur peut ici choisir de déployer l'arborescence plus avant, mais la plupart du temps, les documents s'arretent à ce "niveau de profondeur". (e.g. dans le type de document fiche administrative patient, la liste des services hospitaliers dans lesquels il a déjà séjourné est affichée, mais pas les détails de chaque service),
- les autres attributs deviennent des balises filles.

La DTD du document faiblement structuré est une simplification de cette DTD (on ne conserve que les balises de premier niveau), augmentée de paragraphes non prévus dans la base de données mais définies par l'utilisateur de cet

outil, et dont les informations ne seront pas introduites dans la base (puisque non prévues!). L'utilisateur peut également retirer des paragraphes ou balises qui ne l'intéressent pas, tant que ce retrait ne met pas en jeu l'intégrité de la base de données.

D'autre part, les droits de lecture des documents rédigés et des données incluses dépendent du profil de l'utilisateur. Nous avons donc défini une méthode de filtrage des documents ne fournissant un accès qu'aux informations utiles. Pour cela, nous avons proposé de filtrer la DTD du document à visualiser en fonction des droits de l'utilisateur sur la base de données. Lors d'une demande de consultation, le document est restreint pour être conforme à la DTD de consultation construite pour ce profil.

2.4.4 Conclusion "Partage de données et documents"

Ce travail a fait l'objet du travail de master recherche et de la thèse de doctorat de Mohamed Boumédiene [Bou05], ainsi que de trois Travaux d'Etudes et de Recherche (TER) de maîtrise informatique. Il a donné lieu à deux publications : [44, 26]. Nous avons également développé un prototype en langage Java composé de plusieurs parties :

- un descripteur de bases de données,
- un générateur de schémas médiateurs,
- un médiateur,
- un générateur de modèles de documents à partir de schémas de bases de données.

La thèse de Mohamed Boumédiene a été financée par une bourse de la Région Rhône-Alpes, dans le cadre du projet SICOM (thématique prioritaire région Rhône-Alpes 00-03).

2.5 Conclusion sur la consultation et la saisie adaptées d'informations

2.5.1 Positionnement par rapport à l'état de l'art

De nombreux travaux ont été effectués sur l'interrogation de documents centrés données, notamment sur le principe de requêtes mixtes, c'est-à-dire intégrant à la fois la structure et les données elles-mêmes (XQuery [CFR+01], UnQL [BFS00]...). Mais ces langages portent sur des documents centrés données. La problématique est différente lorsqu'on s'intéresse à des documents non structurés ou faiblement structurés.

L'indexation de données enfouies dans du texte libre ou des textes en langue naturelle ([AEG00]) nécessite l'extraction des informations. De nombreuses propositions d'outils ont été faites dans la communauté linguistique pour l'utilisation d'automates à états finis pour les processeurs de langue naturelle ([Gro89]) et pour l'extraction d'information, tel FS5 [Kar01]. Dans la communauté française, INTEX [Sil00] et Syntex [DBO05] semblent être les outils les plus utilisés.

Un autre domaine de recherche proche de notre problématique concerne l'interrogation et la gestion de données dans des documents XML. Cela consiste la plupart du temps en la mise en correspondance de données irrégulières avec un schéma traditionnel de base de données. Ce type de mapping est proposé par de nombreux systèmes, tels Stored [DFS99], UnQL [BFS00], Lorel [AQM+97], struQL [FFK+98]. De nombreuses études ont été menées également pour la construction de wrappers de pages HTML [CMM01] ou de documents XML [Mus99]. Cependant, la plupart des wrappers ne parviennent pas à effectuer une extraction efficace des données dans le cas où la structure n'est pas complètement régulière. A l'opposé des travaux classiques de wrappers de documents XML, notre processeur d'extraction permet d'une part la recherche d'informations dans des paragraphes narratifs ou de style télégraphique et d'autre part a recours à des dictionnaires linguistiques et techniques, ce qui augmente le taux d'extraction des données.
Par ailleurs, nous n'utilisons pas un format de base de données à la XML, mais utilisons une base de données relationnelle et pouvons donc profiter de la maturité de SQL pour manipuler les données avec des requêtes complexes (étude statistique, analytique, mining, etc) et pour sélectionner les documents correspondants.

Peu de systèmes ont été conçus pour la gestion de documents faiblement structurés. NoDoSE [Ade98] fournit un système semi-automatique pour l'extraction de données dans des instances d'un type de document. Il supporte complètement les documents en texte libre, les documents HTML et en partie les documents XML faiblement structurés. Les données extraites sont stockées dans un schéma relationnel traditionnel. Sur la base d'ontologies, NoDoSE [ECJ+99] formule des règles permettant l'extraction de données pertinentes et applique un système de reconnaissance pour construire des valeurs de tuples d'attributs qui sont insérés dans un schéma de base de données. Cette approche est limitée à des ontologies relativement petites et à des ensembles de mots-clés petits.
Notre système partage avec NoDoSE la même vision basée sur l'extraction des données à partir de documents pour alimenter une base de données. Bien que notre travail soit basé sur des dictionnaires pour extraire les informations, le travail d'Embly est plus ambitieux : il utilise des ontologies pour modéliser une application et ensuite pour générer automatiquement les motifs d'extraction. Ces motifs ne couvrent pas bien la richesse du langage humain ; l'intervention d'un humain est indispensable pour lever les ambiguïtés.

DRUID2 traite d'intégration de schémas. La littérature est très riche dans ce domaine et ce depuis de nombreuses années. Deux étapes majeures ont été suivies : les systèmes fédérés étudiés jusque dans les années 90 ([FRV96], [IFF+99], [RS97]) ont laissé la place aux systèmes de médiation [Wie96]. La médiation suit deux principales approches. L'approche GAV (Global as View) définit le schéma médiateur en fonction des schémas des sources. On peut citer par exemple MIX [BGL+99] ou DISCO [BDGR97]. L'approche LAV (Local as View) définit les schémas des sources en fonction du schéma médiateur prédéfini. Les systèmes PICSEL [Rou02], SIMS [AKS99] ou Observer [MIKS00] suivent cette approche. La plupart des approches LAV, comme la nôtre, initient leur travail de construction du schéma médiateur en utilisant une ou plusieurs ontologies. Cependant, la plupart d'entre elles ne permettent pas de remettre en cause l'ontologie si

elle venait à être insuffisante. Notre proposition le permet, et pour cela, nous dirons que notre travail se situe à mi-chemin entre les approches LAV et GAV, parce que même si les sources sont décrites dans un schéma global, il est possible d'induire des modifications dans le schéma global en fonction de besoins d'une source particulièrement riche.

2.5.2 Points forts et travaux futurs

Dans le cadre du projet DRUID, nous avons montré comment il est possible de relier les mondes bases de données et bases documentaires pour tirer parti des avantages de chacun des mondes. Les documents saisis sont transformés en documents centrés données utilisés pour l'alimentation d'une base de données. Cette base de données peut être utilisée en tant que telle, ou en tant qu'index structuré pour la recherche de documents. La recherche d'un document ne se fait alors plus par mots-clés, mais par rédaction d'une requête SQL complète. La réponse à la requête inclut les documents répondant aux critères de la requête. Nous avons également défini une technique d'intégration de données, permettant une recherche multi-site. Cette intégration de données se fonde sur une ontologie du domaine et sur un appariement des modèles des sources avec l'ontologie. Le schéma médiateur est ensuite contruit par un filtrage de l'ontologie sur les éléments ayant des correspondances dans les sources.

Les résultats de nos travaux pour la consultation et la saisie adaptées de documents peuvent être synthétisés comme suit:
Côté "utilisateur"

- une rédaction de documents faiblement structurés par les utilisateurs finaux,
- une transformation de documents faiblement structurés en documents centrés données pour l'alimentation d'une base de données,
- une recherche d'information sur les données et fournissant une réponse mixte donnant accès aux données et aux documents sources,
- un accès uniforme à plusieurs bases de données.

Côté "administrateur"

- un outil de construction de modèles de documents en fonction de la base de données,
- un langage de rédaction de règles adapté à l'humain,
- un compilateur de règles en transducteurs à états finis,
- un outil semi-automatique d'intégration de schémas de bases de données.

Nous avons justifié ces travaux en nous fondant sur les besoins du dossier médical informatisé. Cependant, ces résultats ne sont pas restreints à ce domaine, et nous avons proposé des solutions génériques, indépendantes du domaine d'application. Ainsi, tout domaine où les utilisateurs rédigent des documents ayant une certaine organisation mais pas une structure finement établie est un candidat potentiel à l'utilisation de nos propositions. Par exemple, NoDoSE s'est intéressé à la gestion de petites annonces de vente ou d'échange de biens et doit fournir un moteur de recherche dans des informations pas toujours bien structurées. Mais ce sont des documents très courts, et il n'est pas sûr que le balisage de paragraphes soit pertinent dans ce domaine.

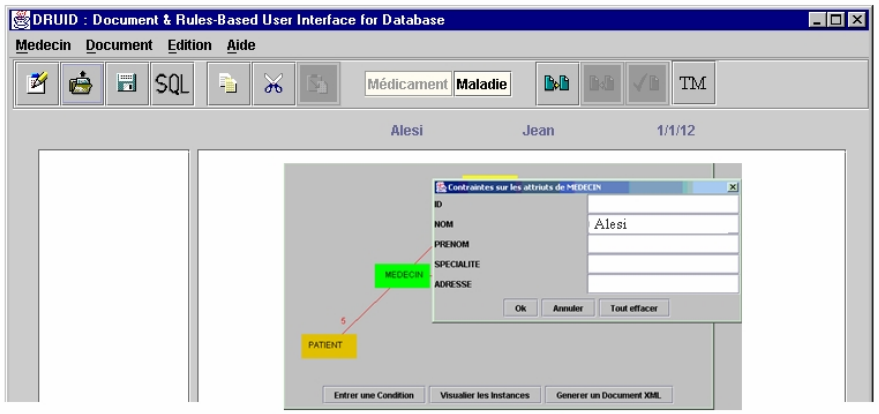


FIG. 2.10 – *Navigation dans une carte de concepts*

Dans un tout autre domaine, les systèmes d'enquêtes pourraient avoir un intérêt à utiliser notre système DRUID dans le cas de questionnaires ouverts. Dans ces questionnaires, la réponse aux questions ne consiste pas en un choix parmi un ensemble fini de réponses possibles, mais en une réponse librement rédigée. L'analyse de ces réponses pourrait être facilitée par une identification de termes-clés potentiellement attendus dans les réponses.

Notre système DRUID est plutôt dédié aux domaines d'expertise où l'utilisateur se sert du document comme un outil lui permettant de garder trace de ses observations, conclusions et plans d'action. L'analyse de l'activité de ces experts requiert une extraction d'informations pour une indexation systématique des cas rencontrés. Pour aller plus loin, on pourrait utiliser DRUID dans l'objectif de construire des modèles de savoir-faire, en l'associant à un outil de classification ou de data mining. Les données extraites peuvent être utilisées pour caractériser les situations et les savoir-faire associés, les documents peuvent être vus comme des fiches exemples illustrant le savoir-faire correspondant.

DRUID pourrait être également utilisé pour construire des synthèses de documents. Les données extraites d'un document peuvent être vues comme un résumé de ce document. Dans le cadre du dossier médical européen, on peut par exemple imaginer que le document centré données construit par DRUID soit fourni à un outil d'adaptation qui pourra fournir des équivalents pour les éléments de banques de données (équivalences de médicaments) ou des traductions pour les termes des classifications (maladies, examens de biologie ou actes par exemple).

L'accès aux informations peut se faire de deux manières : soit par la rédaction de requêtes (et c'est le but premier de DRUID), soit par navigation. Nous avons commencé à étudier la possibilité d'utiliser les données extraites et leur schéma pour élaborer un système de navigation dans les données et les documents. Le principe des topic maps [ISO02] est de fournir une carte de concepts. La thèse de Mourad Ouziri [Ouz04] effectuée au LIRIS a étudié la mise en correspon-

dance d'une carte de concepts avec un ou plusieurs schémas relationnels. Il a également permis d'utiliser la carte de concepts comme un réseau de navigation par l'utilisateur. L'utilisation conjointe de DRUID et des travaux de Mourad Ouziri permettent donc d'effectuer une recherche de documents par navigation dans une carte de concepts. A la sélection d'un concept, le système présente à l'utilisateur l'ensemble des autres concepts liés à celui sélectionné. L'utilisateur peut exprimer des contraintes sur les instances d'un concept en fournissant une requête à la SQL sur le concept (par exemple ne conserver que les instances concernant tel patient, ou telle maladie) et poursuivre sa navigation dans les instances liées à sa sélection. La figure 2.10 montre un exemple d'interface utilisateur de navigation. L'article [2] détaille ce travail.

2.5.3 Vue synthétique

Le tableau 2.1 fournit une vue synthétique du travail effectué pour la consultation et la saisie adaptées de documents. Les informations suivantes sont répertoriées : participants et sujet de leur travail, financements associés, publications.

Thème, thésard rattaché, publications	**Sous-thème**	**Participant au sous-thème**	**Contrats**
QDE [13, 28]		MCF Anne Tchounikine 1998-99	
DRUID thèse Badr 2000-03 [4, 16, 17, 29, 30, 9, 22, 23, 27, 46]	génèse de l'idée première architecture jeux d'essai XSLT pour l'extraction outil de saisie de documents traitement de la langue naturelle compilation de règles d'extraction	PFE/DEA Weill 1999 PFE/DEA Garcia 2000 DEA Varron 2001 stage ingé info Alussi 2001 stage ingé CNAM Bellet 2001 DEA Oprescu 2002 PFE Sastre 2002 et Salce 2003	TIPHAD 99-01 COQUAS 01-04 DocPatient 02-05
DRUID2 thèse Boumediene 2001-05 [44, 26]	du modèle de données à la DTD description d'un modèle de données médiation de schémas	DEA Boumédiene 2001 TER maitrise Boukhari, Benjamaa 2003 TER maitrise Kebaili 2004	SICOM 00-03
Navigation [2]		Pr Christine Verdier 2006-07	

TAB. 2.1 – *Consultation et saisie adaptées d'informations : vue synthétique*

Chapitre 3

Adaptation au contexte

3.1 Introduction

Les avancées en réseaux sans fil et mobiles d'une part et en terminaux mobiles d'autre part ont ouvert une perspective nouvelle en systèmes d'information. Les systèmes d'information pervasifs [Sat01, SM03] sont ainsi nés de l'ambition de fournir un accès au système d'information en tout lieu, à tout moment, et sous une forme et un contenu correspondant à la situation de l'utilisateur. L'informatique ambiante [Wei91] va encore plus loin en associant à tout objet de la vie quotidienne un outil de calcul et de communication, et rendant son intelligence transparente à l'utilisateur. Les systèmes pervasifs et ambiants se fondent en premier lieu sur une connaissance de la situation, appelée communément contexte. L'article [CCDG05] a d'ailleurs pour titre *"Context is key"*. De nombreuses définitions du contexte peuvent être trouvées dans la littérature. La plus fréquemment citée est celle de Dey dans [DSA01] :

> *"context : entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects."*

Le contexte est donc constitué de nombreux types d'informations. On utilise le plus souvent des informations sur la localisation, le réseau, le profil de l'utilisateur, les capacités du terminal utilisé, mais d'autres informations plus "spécifiques" sont également inclues dans cette définition, comme la température, le bruit, la présence de personnes, la tâche en cours, l'agenda... La liste est infinie et chaque application définit les informations qui lui sont utiles.

Le terme *"context-awareness"*, souvent traduit par "sensibilité au contexte", inclut en fait souvent deux thématiques :

- *sentir le contexte* : interagir avec des capteurs de contexte, interpréter les informations de bas niveau captées et ainsi construire une représentation de la situation dans laquelle on se trouve,
- *réagir aux changements de contexte* : en fonction de la situation perçue, améliorer le fonctionnement du système logiciel pour qu'il puisse traiter ou utiliser les ressources et informations à disposition de façon optimale.

La seconde thématique concerne l'adaptation logicielle et nos travaux se situent à ce niveau. Il s'agit de définir et mettre en oeuvre des actions d'adaptation permettant au système logiciel d'agir et interagir de la façon la plus optimale possible. Les changements de contexte sont les déclencheurs des actions d'adaptation. Le contexte est la principale source d'information pour l'adaptation de l'application.

Le domaine médical est en pleine mutation vis à vis des technologies de communication. Les cas où la notion de distance devient une caractéristique non négligeable se multiplient : que ce soit une distance entre le professionnel de santé et le patient (téléconsultation), ou entre le professionnel de santé et son lieu d'exercice habituel (visite ou hospitalisation à domicile) ou entre deux professionnel de santé partageant le suivi d'un patient (réseaux de soins, télé-expertise), la télémédecine est un champ très large et nécessite des supports de communication importants. Vue la multitude des cas, une solution unique ne peut pas être envisagée. Dans les projets TIPHAD, COQUAS et SICOM, nos collaborations avec des associations de suivi d'hospitalisation à domicile (HAD) nous ont conduits à réfléchir à des outils permettant au professionnel de santé d'accéder au système d'information de l'HAD où qu'il se trouve. La première idée, mise en oeuvre dans le projet SICOM et en collaboration avec France Telecom R&D, a été d'installer chez le patient un PC avec visioconférence, permettant à la fois au patient d'effectuer des visioconférences et des saisies d'informations dans le système d'information de l'HAD de façon autonome, mais aussi au professionnel de santé de se connecter au système d'information de l'HAD pendant sa visite pour consulter et mettre à jour le dossier du patient. Mais cette architecture requiert un équipement volumineux et l'installation d'une connexion ADSL chez le patient. Sa mise en place et sa maintenance sont coûteuses. Nous nous sommes donc plus orientés vers l'utilisation de terminaux légers et mobiles, tels les téléphones mobiles ou les assistants personnels (PDA).
Une autre caractéristique des systèmes de HAD est qu'il existe un serveur "centralisé" stockant et gérant les informations sur les patients de HAD. Ce système est hébergé à l'hôpital ou dans l'association de suivi des HAD. Le système est souvent déjà existant, et donc notre problématique s'est concentrée vers la construction d'interfaces utilisateurs adaptables accédant à des services distants et existants. Nos débuts dans cette problématique ont commencé en 2002 avec le projet SEFAGI. Il s'intéresse principalement à l'adaptation des interfaces utilisateurs. L'objectif est de construire, à partir d'une description générique d'une fenêtre composée de couples (fonctionnalité, forme de présentation), le code complet d'une fenêtre interactive, et ce pour différents types de terminaux. Outre l'adaptation de l'interface utilisateur, nous avons dû intégrer des mécanismes simples d'adaptation des données. Le projet SEFAGI fait l'objet de la section 3.2.

En second lieu, nous avons voulu augmenter les capacités d'adaptation de nos travaux, pour intégrer l'adaptation des services et des données. En effet, l'adaptation des interfaces utilisateurs n'est pas suffisante : par exemple, un trop gros volume de données n'est pas non plus acceptable par des terminaux légers. Cet exemple introduit la nécessité de s'intéresser également aux données, et aux services fournisseurs de données. Nous avons donc défini une architecture d'adaptation complète, considérant l'adaptation selon trois axes : les services, les données et les interfaces utilisateur. C'est l'objet du projet SECAS, initié en

2004, et présenté en section 3.3.

La dernière section de ce chapitre (section 3.4) fournit un positionnement par rapport à l'état de l'art et indique des objectifs qui permettront de poursuivre les travaux. Nous présentons finalement une vue synthétique de nos travaux sur l'adaptation au contexte, en y positionnant les participants, les financements et les publications.

3.2 SEFAGI : Interfaces multi-plates-formes

Dans mes travaux de thèse, nous avons travaillé sur l'utilisation de formes documentaires et hypertextes pour la gestion des informations du dossier médical aux utilisateurs finaux. Nous avons également proposé un outil interactif de construction de ces interfaces. Le but est d'"*aider l'utilisateur à se construire une interface qui lui est propre, en le guidant dans ses choix, et en lui assurant une certaine cohérence. Pour assurer la cohérence entre les écrans et le modèle sous-jacent, la construction des écrans se fait à partir d'une représentation de ce modèle. Plus précisément, le concepteur d'une interface (informaticien ou utilisateur) sélectionne un élément du modèle de données, et lui associe un mode de représentation et une position dans l'écran*" ([56], page 111). Les interfaces construites par le prototype étaient en HTML. Dans le projet SEFAGI, nous avons voulu aller plus loin, et ce sur trois points :

- En premier lieu, dans les réseaux d'hospitalisation à domicile, les profils d'acteurs sont très diversifiés (assistante sociale, infirmière, médecin de proximité, médecin spécialiste, kinésithérapeute, pharmacien...), et chacun doit pouvoir utiliser une interface graphique correspondant à ses droits, besoins et préférences.
- Deuxièmement, la multiplicité des intervenants implique une multiplicité des types de terminaux utilisés. La description d'une interface graphique doit donc s'abstraire des plates-formes cibles et servir à construire des codes sources d'interfaces pour plusieurs types de plates-formes.
- Enfin, les interfaces HTML sont réduites en fonctionnalités ; et nous avons donc choisi de travailler sur les interfaces WIMP plus classiques (le choix ne serait peut-etre plus le même aujourd'hui, mais en 2002...).

En ce qui concerne le premier point, nous en avons déduit qu'il n'est pas possible que la construction de toutes les interfaces nécessaires soit confiée à un informaticien. En 1990, Morejon militait déjà pour une "*modification du rôle de chacun dans l'étude d'un système d'information, en passant d'un pilotage 'externe' par des spécialistes à un pilotage 'interne' dans lequel les acteurs de l'entreprise pourront élaborer eux-mêmes les représentations conceptuelles des objets du monde réel qu'ils ont l'habitude de manipuler ainsi que les interactions entre ces objets*" [MON90]. Les utilisateurs finaux doivent pouvoir décrire une nouvelle interface ou adapter une interface existante à leur propres préférences. Un *assistant de description d'interfaces* par les utilisateurs finaux est donc nécessaire.

Pour le second point, nous avons défini un *langage de description d'interfaces abstraites*, qui associe à chaque élément choisi dans l'assistant un mode de représentation abstrait indépendant de toute plate-forme cible. L'assistant de description d'interfaces utilise ce langage pour la description des interfaces abstraites. Un générateur d'interfaces prend en entrée cette description abstraite

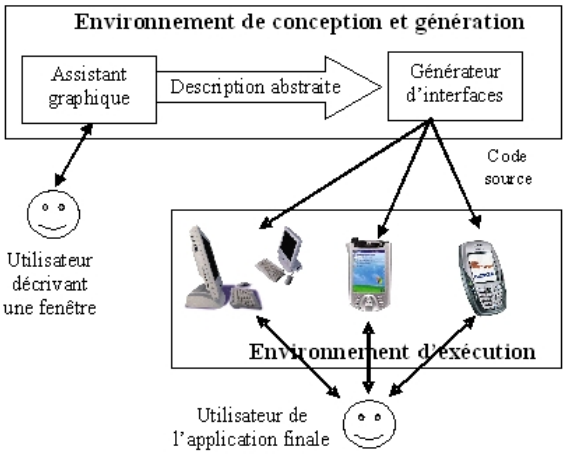


FIG. 3.1 – *SEFAGI : Principe général*

et construit une interface concrète (le code de la fenêtre) pour chaque type de plate-forme cible. Pour chacune, le générateur a des connaissances sur son API graphique et ses capacités.

Pour le troisième point, les plates-formes cible ont été classées en trois catégories : les terminaux standards (acceptant l'API J2SE), les terminaux mobiles à faible capacité (conformes au profil J2ME CDC) et les terminaux mobiles à très faible capacité (conformes au profil J2ME CLDC) [sun07].

La figure 3.1 montre le principe général de SEFAGI : un utilisateur décrit son interface à l'aide de l'assistant. L'assistant construit une description abstraite de fenêtre qu'il fournit en entrée du générateur. Le générateur peut alors produire trois codes sources différents, selon les trois types de terminaux retenus. Ce code source peut ensuite être téléchargé sur chaque terminal qui en aurait besoin. Chacun de ces terminaux doit également contenir un environnement d'exécution qui permet le lancement des fenêtres générées et qui gère les interactions entre elles et avec les services applicatifs.

3.2.1 Description abstraite de fenêtres

L'objectif de faire faire une description abstraite de fenêtre à tout utilisateur final nous a contraints à proposer un langage de description de fenêtres très simple. Une fenêtre est constituée d'un ensemble de panneaux, chaque panneau permettant l'interaction avec un service (une fonction) de l'application. Chaque panneau est décrit par son type et le service invoqué. Nous avons défini et implanté six types de panneaux:

1. le type tableau fournit une présentation tabulaire d'informations structurées, chaque "case" du tableau peut contenir un composant élémentaire (libellé, zone de texte, liste de choix, groupe de boutons radio ou groupe de cases à cocher...),
2. le type texte permet l'édition de textes multi-ligne,
3. le type commande permet l'exécution d'une ou plusieurs commandes.

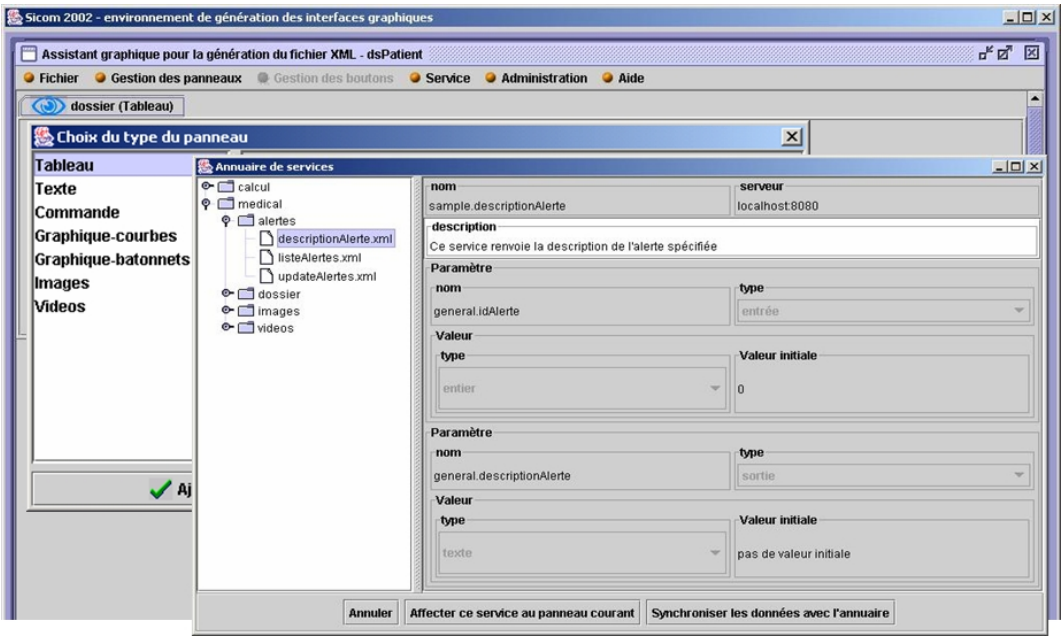


FIG. 3.2 – *SEFAGI : Assistant graphique pour la description de fenêtres*

4. le type graphique permet un affichage de courbes ou de graphiques batonnets, les informations en entrée sont des vecteurs, la première colonne représente l'axe des x, chaque colonne suivante fournit une valeur pour une courbe dans l'axe des y,
5. le type image permet la navigation dans un ensemble d'images, un panneau image est décomposé en trois parties : liste des images, affichage de l'image, affichage du texte associé à l'image,
6. le type video est identique au type image, mais traite des séquences animées.

Cette liste n'est pas définitive. Par exemple il faudrait ajouter un type de panneau permettant de naviguer dans des structures arborescentes.

D'un autre côté, nous fournissons à l'utilisateur un annuaire de services qui répertorie les services disponibles et fournit pour chacun une description textuelle ainsi qu'une spécification de ses entrées, de ses sorties et de son protocole de communication.

Une description abstraite de fenêtre est donc principalement composée d'une liste de couples (type de panneau, service). Pour chaque composant élémentaire, l'utilisateur peut s'il le désire effectuer quelques spécifications : il peut interdire la modification de la valeur présentée, contraindre les valeurs acceptables, cacher l'élément... Des valeurs par défaut sont associées à chacune de ces spécifications.

La figure 3.2 présente une copie d'écran de notre assistant. L'ajout d'un panneau dans une fenêtre se fait par deux sélections. La sélection du service se fait dans une présentation arborescente des services de l'application. La sélection du type de panneau se fait dans une présentation de la liste des types disponibles.

La figure 3.3 fournit une représentation arborescente de la DTD des descriptions abstraites de fenêtres construites par notre assistant. Aucune mention

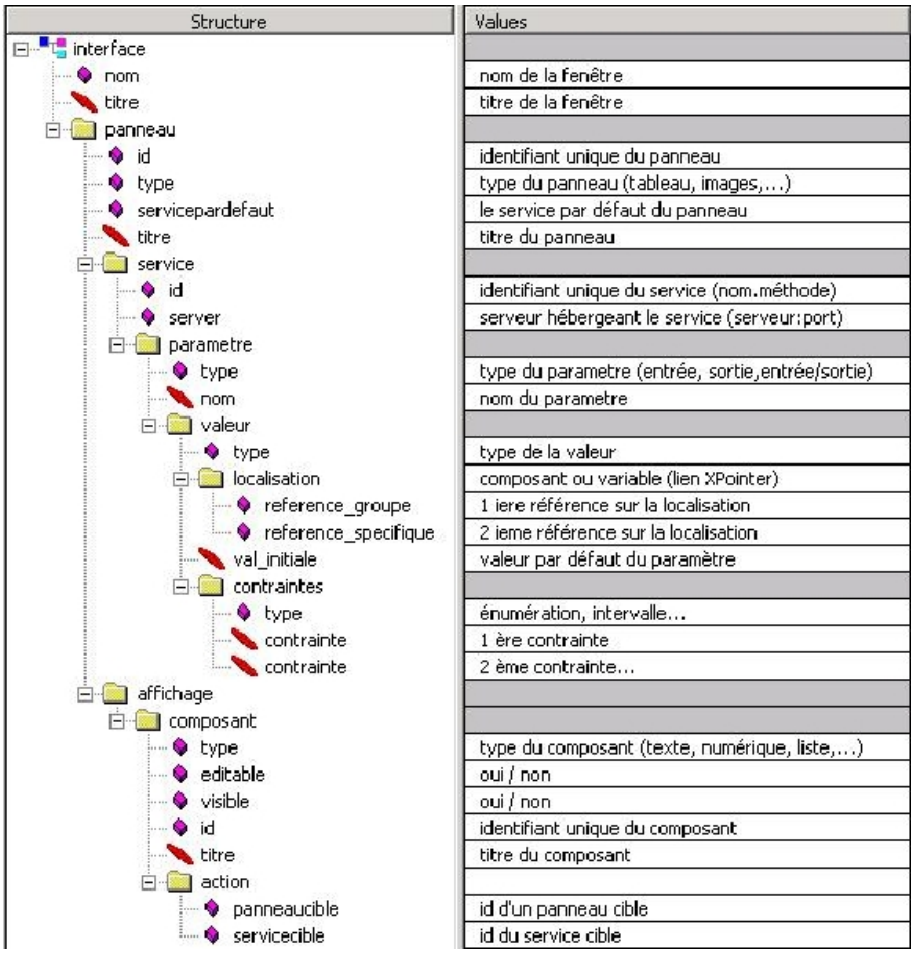


FIG. 3.3 – *SEFAGI : Description abstraite de fenêtres*

à la plate-forme cible n'est faite dans les descriptions abstraites de fenêtres. Elles sont donc très courtes par rapport à celles que l'on peut trouver dans la littérature (voir section 3.4.1). Mais ces descriptions n'ont pas le même objectif. La littérature propose de faire des descriptions fines d'interfaces (intégrant par exemple le positionnement et la configuration de composants élémentaires), souvent par des informaticiens, et rarement de façon totalement dissociée des plates-formes cibles. Elles intègrent souvent des descriptions des services et des protocoles de communication avec ces services. Dans notre proposition, le positionnement des éléments dépend de la plate-forme cible, et est donc défini dans la description de la plate-forme (voir plus loin).

Par ailleurs, la navigation dans une application requiert la mémorisation temporaire d'informations sur les objets en cours de consultation (par exemple l'identifiant du patient dans le cas d'un dossier médical). Ces informations doivent transiter d'un panneau à un autre ou d'une fenêtre à une autre et sont stockées dans une structure d'échange dédiée. Cette structure d'échange prend la forme d'un arbre à deux niveaux : les feuilles sont des couples (items, valeur), les noeuds regroupent des feuilles appartenant à la même entité (par exemple entité patient, entité médecin, entité appareil médical...). Cette structure est automatiquement mise à jour suite aux actions de l'utilisateur final sur l'interface en cours de fonctionnement.

3.2.2 Transformations au cours de la génération

La génération adaptative d'interfaces a pour but de construire le code de l'interface graphique à partir de la description abstraite et des spécificités de chaque plate-forme. Les différences dans le code généré pour les différentes plates-formes concernent trois points inter-reliés : l'affichage, la disposition et la navigation.

La **génération de l'affichage** consiste à remplacer chaque composant de la description abstraite dans son équivalent dans la bibliothèque graphique associée à la plate-forme cible. Pour chaque type de plate-forme cible, il faut construire une sur-couche à l'API standard, contenant l'implantation des différents types de panneaux et composants élémentaires correspondant aux descriptions abstraites.

La **génération des dispositions** consiste à ajuster le positionnement (et la taille) des composants dans le panneau. La disposition est par exemple horizontale sur un grand écran, et verticale sur un petit écran. D'autre part, un panneau abstrait peut être amené à être décomposé en plusieurs affichages. Pour un panneau image ou video, la décomposition fournira trois écrans : la liste des données fournies par le service, la visualisation de la donnée sélectionnée, l'affichage du texte associé à la donnée.

La **génération de la navigation** doit être faite en cohérence avec les générations précédentes et conformément aux possibilités du terminal cible. Ainsi, les classiques barres de menus ou d'onglets seront utilisées sur les grands écrans, alors que des menus contextuels seront privilégiés sur les petits écrans de terminaux mobiles. Une navigation intra-panneau doit également être construite lorsque le panneau a donné lieu à plusieurs écrans.

Dans un premier temps, le travail a consisté à construire un générateur pour chaque type de plate-forme cible. Ces générateurs sont opérationnels. Les principes de transformation sont intégrés dans le code de chaque générateur, ce qui n'est pas souhaitable pour l'évolutivité et l'extensibilité de l'approche. Pour y remédier, un travail de stage de master a eu pour objectif d'étudier les différences entre les différents générateurs, d'en extraire les spécificités de chaque plate-forme pour fournir:

- un générateur générique unique remplaçant les trois générateurs,
- un format de document pour la description de chaque plate-forme.

Ainsi, la génération d'une interface graphique requiert en entrée du générateur générique deux documents : la description abstraite de fenêtre et la description de la plate-forme cible.

La description d'une plate-forme se fait selon trois axes. La structure logique représente la hiérarchie des différentes entités graphiques et leurs propriétés. La structure comportementale définit l'implantation du comportement des différentes entités définies dans la structure logique. Elle permet de décrire l'instanciation, l'initialisation des différents composants graphiques (fenêtres, panneaux et composants élémentaires) et enfin les différentes méthodes utiles pour ces composants. La structure physique définit la disposition des différents composants graphiques. Cette disposition suit des règles d'adaptation dépendantes de la plate-forme cible.

Par manque de moyens humains, le générateur générique n'a pas été implanté. Ce travail est donc resté théorique mais a montré qu'il est possible d'extraire du générateur toute connaissance sur la plate-forme.

La génération adaptative des interfaces graphiques effectue des transformations au cours de la génération du code des interfaces graphiques. C'est donc une adaptation statique. Cependant, ce type d'adaptation n'est pas suffisant, et des transformations sont également nécessaires au cours de l'exécution de l'application.

3.2.3 Transformations au cours de l'exécution

Au cours de l'exécution d'une application utilisant des interfaces générées par SEFAGI, certaines adaptations sont exécutées dynamiquement. Ce sont des transformations de cohérence, de contenu et de transmission.

Les **transformations de cohérence** ont pour but d'adapter les envois de données des services vers le terminal en fonction des écrans générés et des capacités du réseau disponible. Ainsi, lorsqu'un panneau est décomposé en plusieurs écrans, le terminal ne reçoit pas l'intégralité des données fournies par le service, mais seulement celles correspondant à l'écran en cours. De plus, une gestion de points de reprise a été mise en place pour prendre en compte les possibles interruptions de connectivité. Ainsi, nous proposons trois types d'actions d'adaptation : (i) le groupage de données qui est par exemple utile pour consolider les entrées à un service à appeler, à partir de données saisies depuis le terminal ; (ii) la dissociation qui consiste à décomposer un bloc de données en plusieurs "paquets" ; (iii) la reprise qui utilise un mécanisme de sessions pour gérer les

points de reprise éventuellement nécessaires.

Les **transformations de contenu** ont pour objectif de transformer les données pour qu'elles soient manipulables par le terminal cible. Ces transformations consistent en des modifications classiques de taille, format ou structure ou bien en des substitutions de données par d'autres.

Les **transformations de transmission** sont à la marge. Elles ont pour objectif de transformer les données de l'application en données transmissibles sur le réseau. Il s'agit d'une sérialisation des données dans un flux XML à l'envoi, et d'une réification du flux XML à la réception.

3.2.4 Exemple de fenêtre générée avec SEFAGI

Dans ce paragraphe, nous présentons un exemple de description et de génération d'une interface graphique avec SEFAGI. La figure 3.4 présente la description abstraite d'une fenêtre. Elle décrit les caractéristiques générales de la fenêtre (nom, titre) puis donne la composition de chacun des panneaux qui la constituent. Nous avons retiré le détail de la description des panneaux 2 et 3 pour ne conserver que la description du premier panneau. Ce panneau est de type tableau. La première partie de sa description fournit les composants d'affichage requis en précisant leur titre, leur type et s'ils sont visibles et/ou éditables. La seconde partie de sa description concerne les services associés, et la liaison des entrées et sorties des services avec les composants du panneau ou la structure interne de stockage de valeurs de référence.

Cette description abstraite de fenêtre est fournie au générateur, qui produit une interface graphique pour chaque plate-forme cible désignée. La figure 3.5 présente la fenêtre générée pour une plate-forme J2SE, tandis que la figure 3.6 fournit les écrans générés pour les terminaux J2ME CDC. Les écrans générés pour les plates-formes J2ME CLDC sont très proches de ces derniers.

Dans la première figure, on remarquera tout d'abord dans la fenêtre principale qu'une liste de boutons permet de choisir la fenêtre à utiliser. Pour ce type de plate-forme, tous les panneaux sont présentés dans une même fenêtre graphique; ils sont disposés les uns en dessous des autres. Les deux premiers panneaux sont des tableaux, le troisième permet la visualisation d'images ; il inclut une liste d'images, une zone de visualisation de l'image sélectionnée et une zone présentant le texte associé.
La seconde figure présente la même fenêtre pour un terminal CDC. Un premier écran permet d'obtenir la liste des fenêtres disponibles et chaque panneau est décomposé en un ou plusieurs écrans. Pour la fenêtre considérée, le premier panneau tableau est constitué d'un unique écran, car le service ne fournit qu'une ligne de réponse ; les composants sont disposés verticalement. Le second panneau tableau est par contre décomposé en deux écrans : le premier fournit la liste des valeurs de la première colonne du tableau, et la sélection d'une valeur permet d'accéder au second écran fournissant verticalement les informations de la "ligne" correspondante. Le troisième panneau, de type image, est décomposé en trois écrans: le premier écran fournit la liste des images, le second fournit l'image sélectionnée et le troisième présente le texte associé à l'image. Le second écran ne sera fourni au terminal que si celui-ci a les capacités requises pour

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<fenetre nom="DossierPatient">  <titre>test fenetre dossier</titre>
<panneau id="1" type="Tableau" servicepardefaut="1">  <titre>Dossier</titre>
 <affichage>
 <composant id="1" editable="non" visible="oui" type="numérique">  <titre>dossier.idDossier</titre>  </composant>
 <composant id="2" editable="non" visible="oui" type="texte">  <titre>dossier.typeDialyse</titre>  </composant>
 <composant id="3" editable="non" visible="oui" type="texte">  <titre>dossier.modeDialyse</titre>  </composant>
 <composant id="4" editable="non" visible="oui" type="date">  <titre>dossier.dateCreation</titre>  </composant>
 <composant id="5" editable="non" visible="oui" type="date">  <titre>dossier.dateModification</titre>  </composant>
 <composant id="6" editable="non" visible="oui" type="texte">  <titre>dossier.nom</titre>  </composant>
 <composant id="7" editable="non" visible="oui" type="texte">  <titre>dossier.prenom</titre>  </composant>
 </affichage>
 <service nom="sample.listeDossierDialyse" serveur="localhost:8080" id="1">
  <description>Ce service retourne la liste des dossiers de dialyse associés au dossier spécifié</description>
  <parametre type="entrée"> <nom>dossier.idDossier</nom>
    <valeur type="entier"> <val_initiale>0</val_initiale>
      <contraintes type="intervalle">  <contrainte>0</contrainte>  <contrainte>999999</contrainte>  </contraintes>
      <localisation reference_groupe="dossier" reference_specifique="idDossier">variable</localisation>
    </valeur>
  </parametre>
  <parametre type="sortie">  <nom>dossier.idDossier</nom>
     <valeur type="entier"><localisation reference_groupe="1" reference_specifique="1">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie"> <nom>dossier.typeDialyse</nom>
    <valeur type="texte"> <localisation reference_groupe="1" reference_specifique="2">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie"> <nom>dossier.modeDialyse</nom>
    <valeur type="texte"> <localisation reference_groupe="1" reference_specifique="3">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie">
    <nom>dossier.dateCreation</nom>
     <valeur type="date"><localisation reference_groupe="1" reference_specifique="4">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie"> <nom>dossier.dateModification</nom>
    <valeur type="date"><localisation reference_groupe="1" reference_specifique="5">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie"> <nom>dossier.nom</nom>
    <valeur type="texte"><localisation reference_groupe="1" reference_specifique="6">composant</localisation></valeur>
  </parametre>
  <parametre type="sortie"> <nom>dossier.prenom</nom>
    <valeur type="texte"> <localisation reference_groupe="1" reference_specifique="7">composant</localisation> </valeur>
  </parametre>
 </service>
</panneau>
 <panneau id="2" type="Tableau" servicepardefaut="1">  <titre>Températures</titre>
...
</panneau>
 <panneau id="3" type="Images" servicepardefaut="1"> <titre>Images</titre>
...
</panneau>
</fenetre>
```

FIG. 3.4 – *SEFAGI: Exemple de description abstraite de fenêtres*

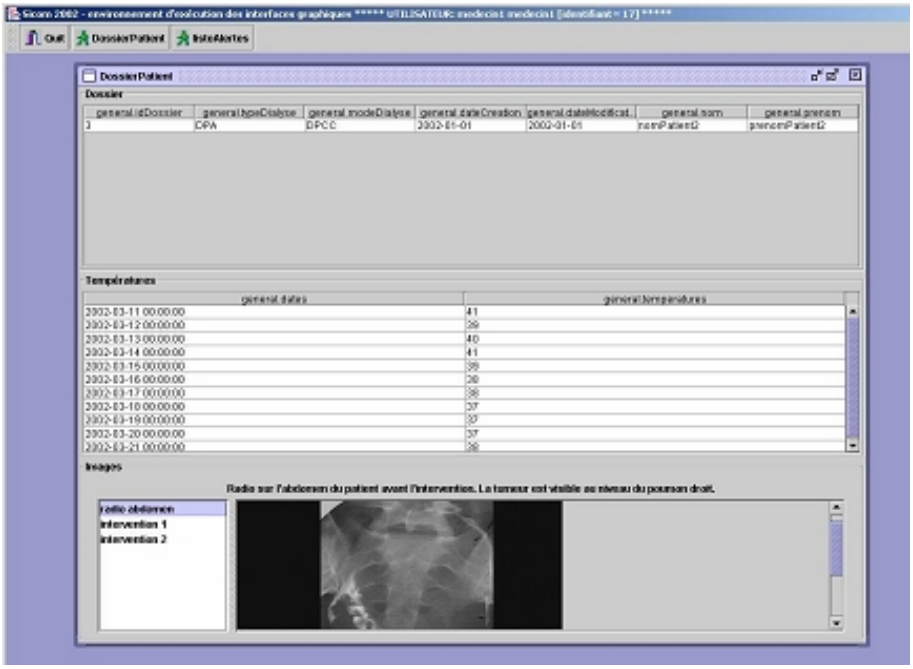


FIG. 3.5 – *SEFAGI : Exemple de fenêtre générée pour PC*

afficher une image (API disponible). La décomposition de la fenêtre en écrans nécessite également la création de liens de navigation entre ces écrans. Ces liens sont disponibles dans les menus associés aux écrans. Une adaptation de données a également été effectuée pour l'image : son format et ses dimensions ont été modifiés.

3.2.5 Conclusion SEFAGI

Ce travail a fait l'objet de stages de master 2 recherche et mastère, de stages d'élèves ingénieurs, de projets de fin d'études d'ingénieurs de spécialités informatique et télécommunications. Une étudiante en DESS ingénierie documentaire a également rédigé un mémoire sous ma direction sur "l'étude des systèmes de télécommunication utilisés dans le cadre de la mobilité géographique des médecins" en 2002. SEFAGI a donné lieu à six publications : un article de revue nationale [15], deux articles dans des conférences internationales [8, 43], un article dans une conférence nationale [19], et un article dans un workshop [24]. Plusieurs prototypes (en langage Java) ont vu le jour dans ce projet:

- un générateur d'interfaces graphiques pour PC J2SE,
- un générateur d'interfaces graphiques pour mobiles J2ME CLDC,
- un générateur d'interfaces graphiques pour mobiles J2ME CDC,
- un assistant pour la description abstraite d'interfaces graphiques.

Le projet SICOM a financé nos travaux pour SEFAGI, et nous avons validé les transformations à l'aide de l'étude sur le suivi de dialyses à domicile étudiées dans ce projet. Ces travaux, principalement menés par Tarak Chaari (projet de fin d'études, stage de master et co-encadrement des stagiaires des années

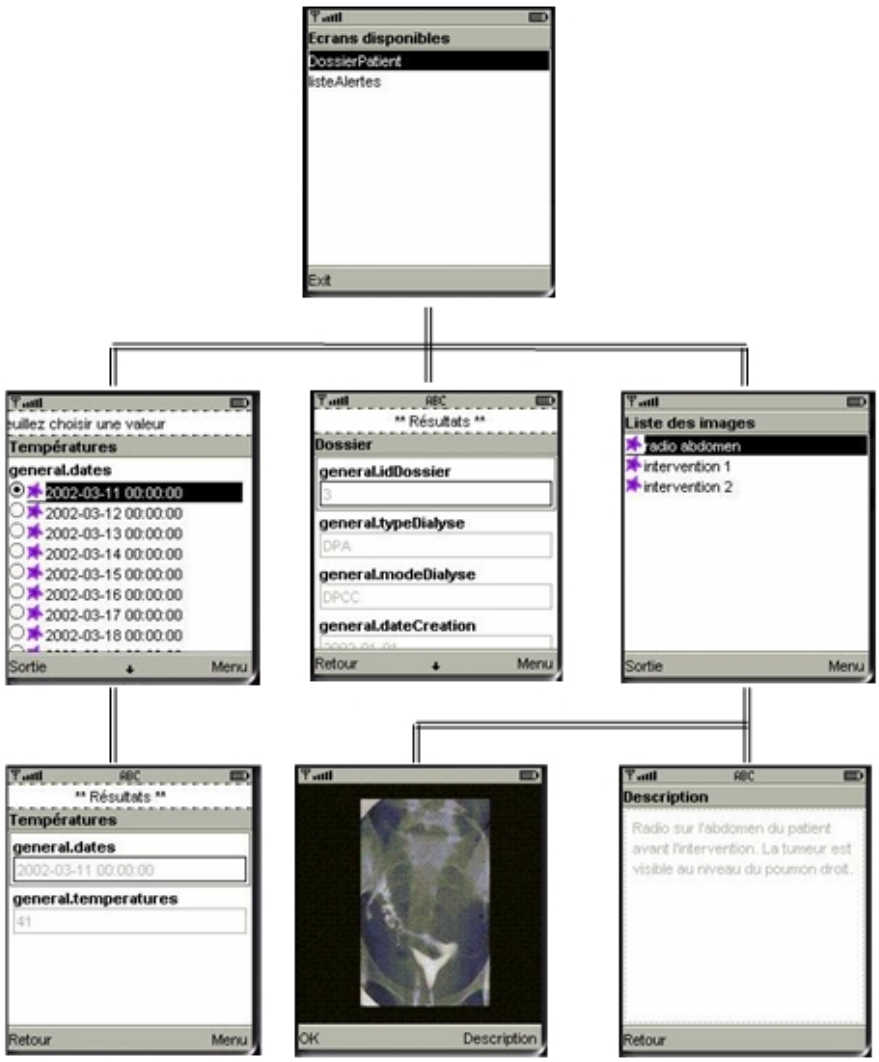


FIG. 3.6 – *SEFAGI : Exemple de fenêtre générée pour PDA*

suivantes), ont servi de point de départ pour le projet SECAS présenté en section 3.3.

3.3 SECAS : Adaptation au contexte selon trois facettes

Le projet SEFAGI s'est essentiellement centré sur l'adaptation des interfaces utilisateurs. Des transformations de contenu ont été introduites, mais de façon marginale et insuffisante. Les tests que nous avons pu faire sur SEFAGI nous ont montré que l'adaptation des interfaces graphiques n'est pas suffisante en soi pour l'accès à un ensemble de services pré-existants. Il est nécessaire d'effectuer au préalable une adaptation des données échangées. L'originalité de SECAS par rapport à beaucoup de travaux existants, est que nous poussons l'adaptation jusqu'aux services de l'application, pour fournir à l'utilisateur une vue de l'application conforme à son contexte actuel. Cette vue est dynamique ; elle évolue avec le contexte. Le projet SECAS a donc eu pour objectif de proposer une architecture et des techniques d'adaptation pour une adaptation complète selon trois volets : adaptation des services, des données et des interfaces utilisateurs.

D'autre part, dans SEFAGI, l'adaptation se concentre sur une seule composante du contexte: la plate-forme du terminal. Avec SECAS, nous proposons une architecture permettant la prise en compte de toutes les composantes du contexte, comme l'utilisateur et son environnement, le terminal utilisé, le réseau disponible, l'application... SEFAGI n'a été qu'un premier pas dans le monde de la gestion du contexte. SECAS y entre complètement.

Après une description de l'architecture générale de SECAS, nous introduisons les principes d'adaptation dans SECAS puis nous détaillons les techniques d'adaptation de chacune des trois facettes.

3.3.1 Architecture générale de SECAS

La figure 3.7 présente l'architecture générale de la plate-forme SECAS. Cette architecture est décomposée en trois couches. La couche d'intégration d'applications permet la prise en compte de nouvelles applications. La couche de gestion du contexte représente le contexte de l'utilisateur et fournit les informations nécessaires à l'adaptation. La couche d'adaptation est la couche principale de notre travail. Elle s'occupe d'orchestrer et effectuer l'adaptation des trois facettes de l'application : services, données et interfaces utilisateurs. Nous détaillons chaque couche ci-dessous.

Dans la **couche d'intégration d'applications**, une application est vue comme un ensemble de services inter-reliés par leurs entrées et sorties. Elle fournit un accès à ses services de façon standard. Elle fournit également un modèle de ses services et de leurs interactions. Nous l'appelons le modèle fonctionnel de l'application (MF). L'intégration d'une nouvelle application se fait par l'introduction dans SECAS du modèle fonctionnel correspondant.

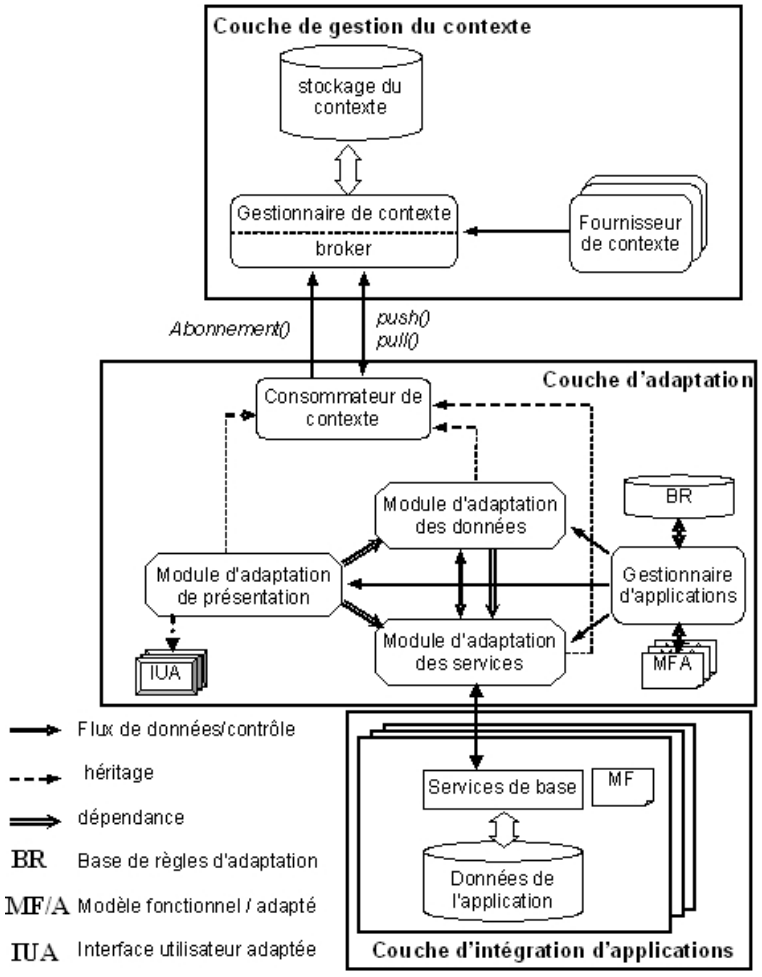


FIG. 3.7 – *SECAS : architecture générale*

Dans la **couche de gestion du contexte**, les fournisseurs de contexte s'occupent de capturer, interpréter et structurer le contexte pour qu'il soit dans une forme exploitable par la couche d'adaptation. Le contexte est ensuite géré par un gestionnaire de contexte, qui fournit les modifications à la couche d'adaptation (en mode push ou pull selon les besoins) et stocke les valeurs de manière persistante si nécessaire. Chaque session utilisateur a son contexte.

Dans la **couche d'adaptation**, le gestionnaire d'applications orchestre l'adaptation. Il dirige les trois modules responsables des facettes à adapter. Il utilise pour cela une base de règles d'adaptation (BR) qui indique quelles adaptations effectuer selon le contexte. Le module d'adaptation de services reçoit un modèle fonctionnel et le transforme selon les règles et la situation contextuelle en un modèle fonctionnel adapté (MFA). Le module d'adaptation des données reçoit le modèle fonctionnel adapté et y ajoute les outils de transformation des données correspondant à la situation contextuelle. Le module d'adaptation des interfaces reçoit le modèle fonctionnel adapté et la description du contexte et en déduit l'interface utilisateur pertinente. Les fenêtres générées sont déployées sur le terminal cible (IUA). Ce processus se déroule à chaud, et ne nécessite pas l'interruption du dialogue avec l'utilisateur. Chaque session utilisateur a son modèle fonctionnel adapté qui évolue indépendamment de celui des autres sessions.

3.3.2 Principe d'adaptation dans SECAS

Avant de décrire les étapes de l'adaptation dans SECAS, nous devons préciser deux notions importantes dans notre travail : le contexte et le modèle fonctionnel.

Contexte Nous partons du principe que les informations de contexte ne concernent pas l'application, mais ont une influence sur le "retour" à l'utilisateur. Ainsi, nous ne parlerons d'information contextuelle que lorsque de telles informations ont pour objectif d'adapter l'application pour qu'elle puisse être utilisée de manière optimale par l'utilisateur. Or, les définitions du contexte que l'on peut trouver dans la littérature ne font pas une distinction franche entre les informations de l'application et les informations du contexte. Certains utilisent d'ailleurs le terme contexte pour toute information qui serait issue de capteurs. Pour nous, un capteur peut selon les cas fournir des données applicatives (par exemple des capteurs de mouvement en sismologie ou des capteurs GPS dans une application de guidage), ou des données contextuelles (par exemple des capteurs GPS dans une application médicale ou des capteurs de niveau sonore dans une application bancaire).
Les informations contextuelles n'ont pas d'influence sur les services originaux de l'application. Par contre, ces informations contextuelles entraînent des adaptations. Une information contextuelle ne sera pas fournie en entrée des services de l'application; elle sera utilisée par la couche d'adaptation pour transformer la vue que l'utilisateur a de l'application. Par exemple, pour fournir un long texte à l'utilisateur, on préfèrera une interface audio dans un environnement calme, et une interface graphique dans un environnement bruité; mais le service de l'application fournit dans tous les cas un texte. La couche d'adaptation décidera d'en faire une synthèse vocale le cas échéant.

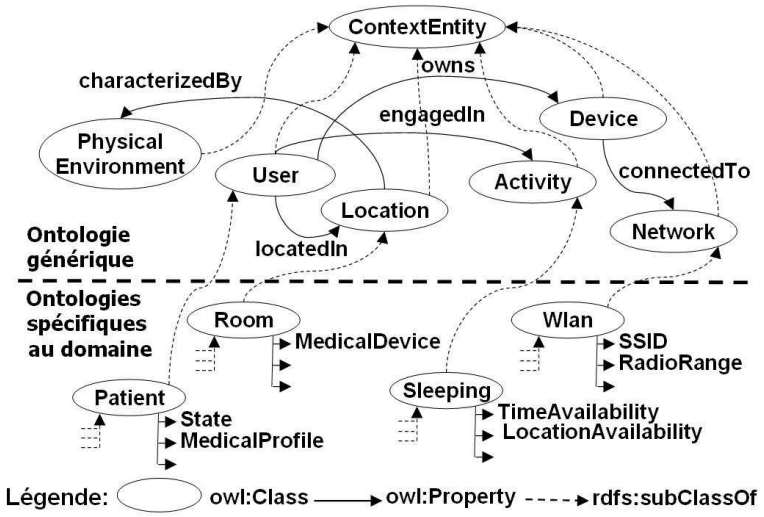


FIG. 3.8 – *SECAS : ontologie du contexte*

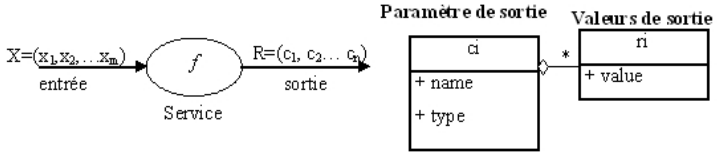


FIG. 3.9 – *SECAS : modélisation d'un service*

Un travail en collaboration avec Dejene Ejigu et Marian Scuturici du LIRIS nous a permis de définir une structuration du contexte sous la forme d'une ontologie à deux niveaux. Le premier niveau, que l'on peut qualifier de basique ou générique, définit les grandes thématiques habituellement rencontrées dans les travaux sur la sensibilité au contexte. Mais il est rare que ce niveau basique soit directement instancié. Le second niveau permet de définir des concepts spécifiques héritant des concepts de base. Ces concepts sont spécifiques à chaque application cible. Dans ce premier travail réalisé avec Dejene et Marian, l'application et les services font partie du contexte. Ceci va à l'encontre de notre vision des choses dans SECAS, et nous avons poursuivi la réflexion pour aboutir à un modèle de contexte comme présenté dans la figure 3.8.

Service, dépendance d'exécution et modèle fonctionnel Les services sont vus comme des boites noires, recevant en entrée un vecteur de paramètres valués et fournissant en sortie un autre vecteur de paramètres valués. Un vecteur est un tableau de listes de valeurs. La figure 3.9 illustre la modélisation d'un service.

Il existe des liens entre les services, basés sur les entrées et sorties. Les paramètres nécessaires à l'exécution d'un service f peuvent provenir de la sortie de plusieurs services f1, f2... On dit que deux services ont une dépendance d'exécution lorqu'une partie de la sortie d'un service se retrouve à l'entrée de l'autre

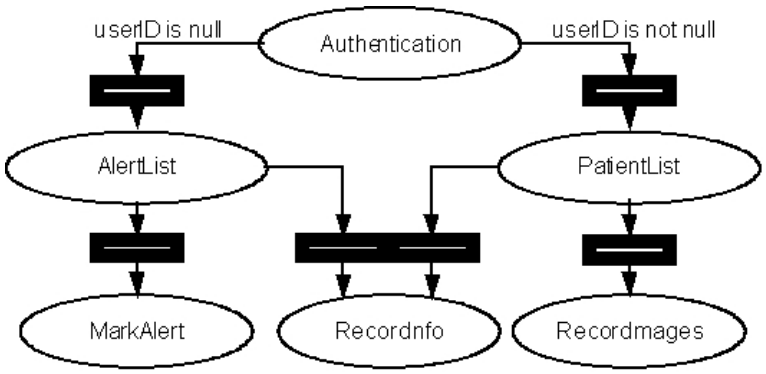


FIG. 3.10 – *SECAS : exemple de modèle fonctionnel*

service. Les dépendances entre services peuvent être en "et" si f dépend de f1 et f2 ou en "ou" si f dépend de f1 ou de f2.
Chaque transition est un triplet (d, gc, A) où d est le délai maximum pour passer la transition, gc est la condition générale de la transition et A est un ensemble fini d'associations entre services (a1,...,an). Chaque transition ai est un ensemble fini de paires (inputExpression, destinationParameter) liant un paramètre de sortie inputExpression d'un service à un paramètre d'entrée destinationParameter d'un autre service.
Pour représenter ces liens, nous avons opté pour un modèle des applications fondé sur un réseau de Petri, que nous appelons modèle fonctionnel. Dans ce modèle fonctionnel, les places représentent les services, et les transitions représentent les dépendances d'exécution entre les services. La figure 3.10 présente un exemple de modèle fonctionnel. Un service "racine" est activé au lancement de l'application (ici, et comme souvent, un service d'authentification). Un service du réseau de pétri est accessible à l'utilisateur dès lors que sa transition en entrée est vraie. Par exemple, le service MarkAlert ne peut pas être utilisé avant que le service AlertList ait donné une sortie valide et que l'expression booléenne de la transition soit vraie.

Le modèle fonctionnel fournit une représentation de l'application disponible. La couche d'adaptation utilise ce modèle et l'adapte au contexte pour en fournir une vue conforme à la situation contextuelle de l'utilisateur. Comme nous le verrons plus tard, l'adaptation du modèle fonctionnel consiste en la création d'adaptateurs qui remplacent les services originaux dans la vue utilisateur. Ils se chargent de faire les appels aux services originaux et de les adapter. Pour l'instant, les modèles fonctionnels sont rédigés manuellement par un administrateur. Nous avons dans un premier temps défini une extension du modèle PNML (Petri Net Markup Language) [BCvH+03] pour rédiger les modèles fonctionnels. C'est ce modèle qui est implanté dans notre prototype.
Cependant, nous avons poussé le travail plus loin en proposant une ontologie de description des applications, présenté en figure 3.11. Cette ontologie comprend un niveau générique indépendant de toute plate-forme et de toute application, et un niveau spécifique décrivant un graphe de composants interconnectés. La figure 3.12 donne le niveau spécifique correspondant au modèle fonctionnel sous

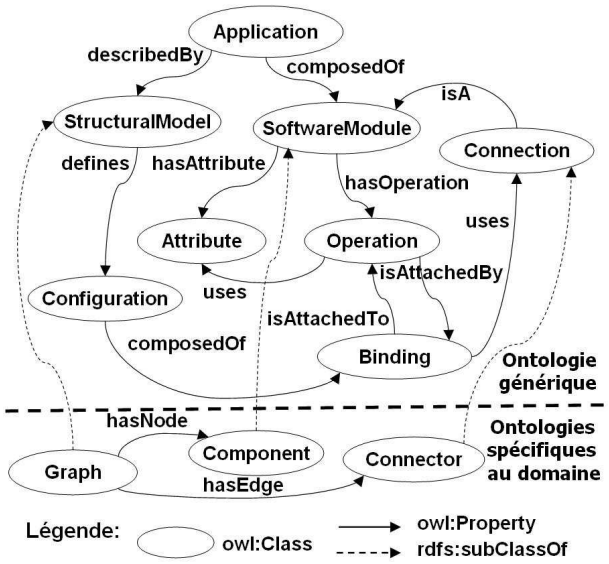


FIG. 3.11 – *SECAS : ontologie de description d'applications (niveau générique) et ontologie de description d'un modèle fonctionnel (niveau spécifique)*

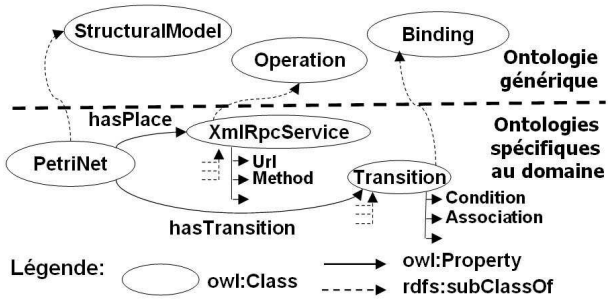


FIG. 3.12 – *SECAS : ontologie de description d'un modèle fonctionnel sous forme de réseau de Petri*

la forme d'un réseau de Petri comme utilisé dans SECAS.

Etapes d'adaptation La figure 3.13 donne une vue générale des étapes suivies par la couche d'adaptation. Le modèle fonctionnel initial de l'application est utilisé à chaque changement de contexte pour calculer le modèle le plus adapté à la situation. Ce modèle fonctionnel est en premier lieu pris en charge par le module d'adaptation des services de l'application. Le modèle fonctionnel sortant de ce module comporte des adaptateurs qui masquent les services initiaux et en adaptent le comportement. Ce modèle est fourni en entrée du module d'adaptation de contenu qui effectue les adaptations des données nécessaires. Le modèle fonctionnel sortant de ce module comporte des adaptateurs de contenu qui masquent les adaptateurs de comportement insérés précédemment. C'est ce modèle fonctionnel doublement adapté qui est visible par l'utilisateur final via une interface utilisateur. Ce modèle fonctionnel doublement adapté est fourni

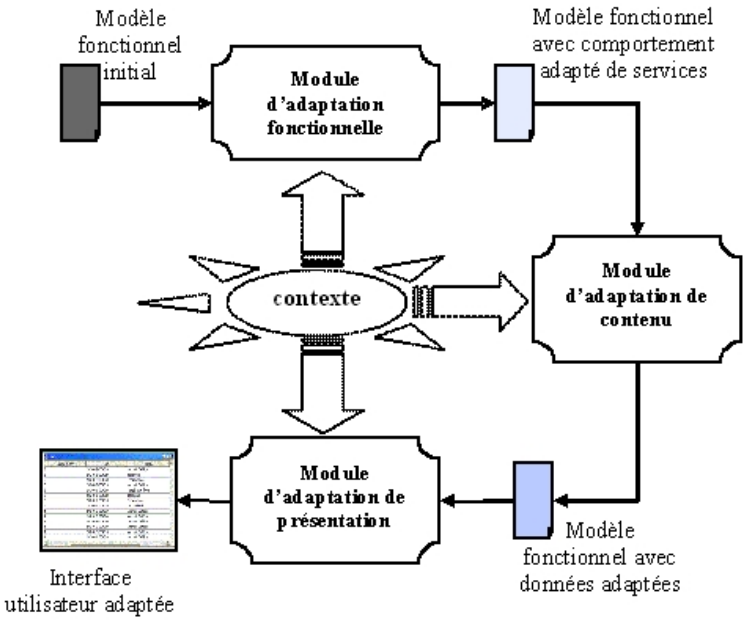


FIG. 3.13 – *SECAS : étapes d'adaptation*

au module d'adaptation des interfaces graphiques qui associe à chaque place du modèle une interface d'interaction en entrée et en sortie. Le contexte est utilisé par chaque module d'adaptation, chaque composant de ces modules étant abonné aux paramètres contextuels qui l'intéressent.

3.3.3 Adaptation dynamique de services

La première étape de l'adaptation du modèle fonctionnel concerne les services. Cette adaptation consiste en des transformations aussi bien au niveau des places que des transitions du modèle. L'adaptation des services nécessite l'établissement d'une politique d'adaptation. Cette politique se fonde sur deux éléments essentiels : les opérateurs d'adaptation définissant les transformations possibles sur le modèle fonctionnel, et les règles d'adaptation qui précisent les actions à effectuer selon les situations contextuelles rencontrées. Nous avons défini des opérateurs de transformation du modèle, ainsi qu'un langage de spécification de règles d'adaptation indiquant les opérateurs à appliquer en fonction de l'état du contexte.

Opérateurs d'adaptation Les opérateurs d'adaptation que nous avons définis sont de deux types : les opérateurs intra-service qui agissent sur les places du modèle et les opérateurs inter-services qui agissent sur les transitions et les

Opérateur	**Description**	**Paramètres** (en plus du/des service(s) concerné(s))
projection	sélectionne un sous-ensemble de paramètres de sortie d'un service	liste des paramètres à conserver
sélection	retire des instances du vecteur de sortie du service	critère de sélection des instances
produit	effectue le produit cartésien entre les sorties de deux services	
union	concatène les sorties de deux services ayant des vecteurs de sortie de même structure	
addVersion	ajoute une nouvelle version d'un service	
activateVersion	sélectionne une version d'un service parmi la liste des versions disponibles	
removeVersion	supprime une version d'un service	

TAB. 3.1 – *SECAS: opérateurs d'adaptation intra-service*

services rattachés aux transitions impliquées.

Un **opérateur d'adaptation intra-service** prend en entrée un sevice à adapter et produit en sortie un autre service. Dans le modèle fonctionnel, chaque service à adapter est substitué par un service adaptateur qui englobe l'appel au service à adapter et applique les opérateurs d'adaptation requis. Chaque adaptateur peut référencer plusieurs services, dans le cas où une fonctionnalité peut être remplie par plusieurs services concurrents (versions). L'adaptateur reçoit en entrée les paramètres d'entrée du service adapté et les valeurs de contexte nécessaires à l'adaptation. Le tableau 3.1 donne une vue synthétique de ces opérateurs. Certains de ces opérateurs sont inspirés des opérateurs de l'algèbre relationnelle. Mais il sont différents dans leur mise en oeuvre. En effet, ils agissent sur les entrées et sorties des services, et non sur des relations normalisées. Les vecteurs n'ont pas la notion de clé, ils ne sont pas soumis aux contraintes d'intégrité des bases de données.

Les **opérateurs d'adaptation inter-services** prennent en entrée un modèle fonctionnel et produisent un modèle fonctionnel adapté. Ces opérateurs sont appliqués par le gestionnaire d'adaptation de services, chef d'orchestre du module d'adaptation de services. Le tableau 3.2 donne une vue de synthèse de l'ensemble des opérateurs définis. Nous avons également défini des algorithmes permettant de vérifier la cohérence du modèle fonctionnel résultant de l'application de ces opérateurs.

Règles d'adaptation Les règles d'adaptation définissent les actions d'adaptation à effectuer sur le modèle fonctionnel de l'application en fonction des situations contextuelles rencontrées. Une règle est une paire (condition, action). La condition est une expression booléenne classique sur des valeurs du contexte et des propriétés de l'application. L'action décrit un ensemble d'opéra-

Opérateur	**Description**	**Paramètres** (en plus du/des service(s) concerné(s))
replaceService	remplace un service par un autre et met à jour les transitions correspondantes	ancien service et nouveau service
insertServiceAfter	ajoute un nouveau service à la sortie d'un service existant et construit les transitions correspondantes	service existant et nouveau service
insertAlternative	insère un service comme frère d'un service existant et construit les transitions adéquates	service existant et nouveau service
lockService	vérrouille l'accès au service	service à bloquer
unlockService	dévérrouille l'accès au service	service à débloquer

TAB. 3.2 – *SECAS : opérateurs d'adaptation inter-services*

teurs d'adaptation à appliquer sur le modèle fonctionnel au cas où la condition est vraie. Par exemple, $\neg context.terminal.acceptedDataTypes.acceptImages \wedge \exists f \in F | \exists i | f.R[i].type = "image" \Rightarrow lockService(f)$ donne une règle d'adaptation. La condition est la suivante : "le terminal ne supporte pas les images et il existe un service qui fournit un paramètre de sortie de type image". L'action est "bloquer l'accès au service".

Modèle de la politique d'adaptation La politique d'adaptation a été modélisée sous la forme d'une ontologie dans le travail de master de Mohamed Zouari (figure 3.14). Cette ontologie est construite sur deux niveaux, selon le même principe que pour l'ontologie du contexte. Le niveau générique est indépendant de toute plate-forme d'adaptation. Les spécificités de SECAS constituent un niveau spécifique possible, comme présenté en figure 3.15.

3.3.4 Adaptation dynamique de contenu

L'adaptation de contenu a pour objectif de transformer les données échangées entre les services et le terminal de l'utilisateur pour qu'elles correspondent à la situation contextuelle. Pour chaque place du modèle fonctionnel avec services adaptés, le module d'adaptation de contenu instancie un adaptateur de contenu. Cet adaptateur de contenu vérifie l'adéquation entre les caractéristiques des données et les caractéristiques actuelles du contexte. Le cas échéant, il fait appel à un outil de planification d'adaptation de contenu. L'outil que nous utilisons est le travail de thèse de Girma Berhe [BBP05].

3.3.5 Adaptation dynamique d'interfaces

A la différence de SEFAGI, SECAS ne peut pas se baser sur un assistant pour construire les interfaces utilisateur. En effet, les services disponibles changent dynamiquement suite aux adaptations qu'ils subissent ; les données échangées également. Il faut un processus autonome pour construire à la volée les interfaces nécessaires.

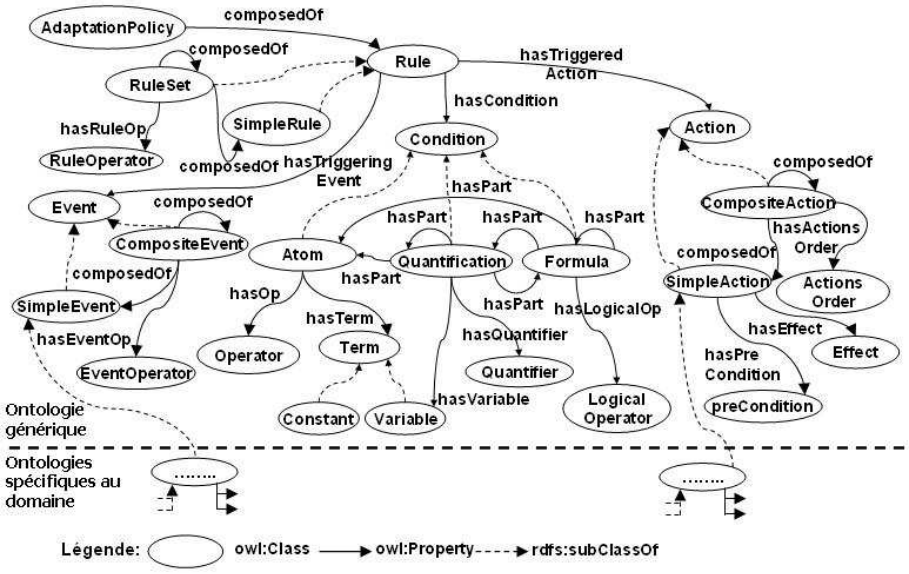


FIG. 3.14 – *SECAS : ontologie générique de politique d'adaptation*

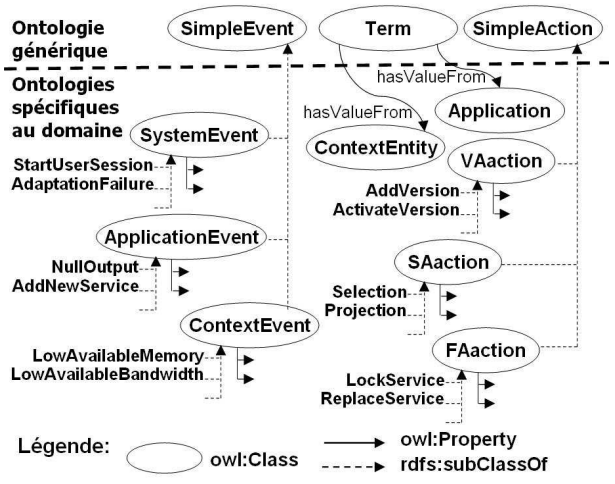


FIG. 3.15 – *SECAS : ontologie spécifique de politique d'adaptation pour SECAS*

De SEFAGI, nous avons retenu la structuration de l'interface graphique en fenêtres comportant des panneaux. Un panneau est constitué d'un ensemble de composants graphiques abstraits, pour lesquels chaque plate-forme doit fournir une implantation.

L'interaction avec un service se fait en deux temps : saisie des entrées et appel au service puis présentation des sorties. Un panneau d'entrée permet de compléter les entrées nécessaires à l'appel du service (affectation de valeurs au vecteur d'entrée du service). Un panneau d'entrée présente les valeurs déjà connues du vecteur d'entrée du service ; elles proviennent des transitions entrant vers le service. Le panneau d'entrée fournit également des composants graphiques permettant la saisie des valeurs manquantes dans le vecteur. Un panneau de sortie permet de présenter à l'utilisateur les résultats de l'activation du service, c'est-à-dire de présenter les valeurs de son vecteur de sortie.

A chaque type de données est associé un ensemble de composants abstraits d'affichage et d'édition. Le module d'adaptation des interfaces graphiques est ainsi capable de construire de lui-même les panneaux d'entrée et de sortie en fonction des types de données contenus dans les vecteurs correspondants et des composants graphiques disponibles. Les panneaux sont élaborés au niveau abstrait, c'est-à-dire indépendamment de la plate-forme cible. Cette description suit la DTD définie dans SEFAGI (cf. figure 3.3). Une étape de réification s'ensuit, qui construit le code correspondant à la plate-forme cible. Cette étape se fonde également sur le générateur de SEFAGI (cf. section 3.2.2). Les caractéristiques de la plate-forme sont des informations contextuelles.

La postition du service dans le modèle fonctionnel permet également de construire automatiquement le menu de l'utilisateur. Il est constitué de l'ensemble des services accessibles. Les services accessibles sont ceux dont les transitions sont évaluées à vrai.

Il est également important de fournir une méthode de chargement des interfaces graphiques sur le terminal "à chaud". SEFAGI requiert l'import manuel d'une archive contenant les interfaces disponibles. SECAS doit permettre d'ajouter ou supprimer des interfaces en fonction de l'évolution du contexte. C'est pourquoi notre architecture se fonde sur OSGi [all04] et les web services. OSGi est un middleware permettant d'ajouter ou retirer des archives (bundles) à chaud. Il existe aujourd'hui des versions pour plates-formes PC et pour terminaux mobiles CDC.

3.3.6 Conclusion SECAS

Ce travail a fait l'objet de la thèse de doctorat de Tarak Chaari [Cha07], de stages de master 2 recherche, de stages d'élèves ingénieurs, de projets de fin d'études d'ingénieurs. Nous avons également collaboré avec le Professeur Augusto Celentano de Venise lors de son congé sabbatique au LIRIS; il est coauteur de plusieurs de nos publications. Ce projet SECAS a donné lieu à sept publications : deux articles de revues [1, 3], trois articles dans des conférences internationales [6, 25, 42], et deux articles dans des conférences nationales [20, 21]. Nous avons présenté ces travaux dans le cadre de l'action Adaptation du GDR

ASR [41], lors de la réunion à Perpignan en octobre 2006. Nous participons également aux réunions du GT Mobilité du GDR I3.

Un prototype de la plate-forme SECAS a été implanté en Java et sur le middleware OSGi. Il utilise les standards Web services et XML-RPC, jugé plus adapté que SOAP pour les terminaux à faible capacité. Notre prototype comporte l'ensemble des modules de la couche d'adaptation et une simplification de la couche de contexte. La plate-forme SECAS représente actuellement environ 15000 lignes de code Java réparties en 79 classes.

Les tests ont nécessité le développement d'une couche application. Nous avons utilisé une application médicale de gestion de dialyses à domicile dont les spécifications proviennent du projet SICOM. Ces tests nous ont permis de valider le bon fonctionnement de la plate-forme. Les adaptations se déroulent comme prévu, et les phases d'adaptation n'induisent pas de temps d'attente prohibitifs. Nous n'avons pas pu réaliser de tests en grandeur nature. Il serait intéressant d'effectuer des tests sur une application plus complexe, et avec un ensemble de règles d'adaptation plus fourni. C'est une critique que l'on peut d'ailleurs très souvent faire au domaine de recherche de l'adaptation pour les systèmes pervasifs : il nous manque toujours l'application de référence qui permettrait de réaliser de réels benchmarks sur nos propositions.

3.4 Conclusion sur l'adaptation au contexte

3.4.1 Positionnement par rapport à l'état de l'art

Dans la littérature, on peut trouver de nombreux travaux sur des langages de descriptions d'interfaces, qu'on aurait tendance à rapprocher de notre description abstraite de fenêtres élaborée dans le projet SEFAGI. UIML [AM02] se fonde sur des modèles de dialogue pour décrire les composants graphiques d'une interface, les données présentées à l'utilisateur et les interactions entre l'utilisateur et l'interface. AUIML [CCT+03] est un dérivé d'UIML qui sépare les données des composants d'affichage. Ce langage a pour cible les PC standards. SunML [PDFP03] est un autre langage dérivé d'UIML qui se concentre sur les services avec lesquels les interfaces interagissent. Ce langage propose de nombreux opérateurs pour composer et projeter des descriptions SunML sur de nombreux langages cibles (HTML, C++, VoiceXML...). XIML [PE02] et USIXML [CV05] modélisent tous les aspects d'une interface utilisateur : les interactions avec l'utilisateur, la présentation, les tâches à accomplir, les données à afficher, les plates-formes cibles. Ces langages offrent des descriptions riches et précises d'interfaces multi-plates-formes mais le grand nombre de détails nécessaires augmente considérablement leur complexité et réduit d'autant leur utilisabilité. PlasticML [Rou03] tente de réduire la complexité des langages existants en séparant l'interface du code sous-jacent, mais il est dédié aux interfaces web.

Les nombreuses recherches en gestion du contexte qui ont été menées les années passées se sont principalement concentrées sur l'acquisition, le traitement, la modélisation et la transmission du contexte à l'application. Le travail le plus

connu dans le domaine est celui de Dey et son équipe [SDA99]. Dey a défini une architecture pour l'acquisition, l'interprétation et l'assemblage de données contextuelles provenant de sources diverses à base de context widgets. La modélisation du contexte est également un domaine prolixe. On peut citer CoOL [SLPF03], les travaux de l'équipe de Henricksen en modélisation objet [HIR02], les travaux de [WZGP04] sur les ontologies, la proposition 5W1H basée sur XML de [JW03], ou encore les travaux de Kirsch Pinheiro [Pin06] dédiée aux environnements coopératifs.

L'adaptation de l'application au contexte est une étape en aval des travaux précédents: le contexte est disponible à l'application ; à elle maintenant de savoir le prendre en compte. Bien sûr la modélisation du contexte choisie a des implications sur l'adaptation de l'application. La problématique de l'adaptation de l'application elle-même peut être vue selon trois facettes : l'adaptation des interfaces utilisateur, l'adaptation des données, et l'adaptation des services rendus.

- L'adaptation des interfaces utilisateurs est traitée sous deux angles. Le premier angle concerne les langages de description d'interfaces et la génération automatique de code associée, comme présenté ci-dessus. La seconde, nommée plasticité des interfaces, est un sujet porteur et important. Il inclut l'adaptation au terminal et aux dispositifs d'entrée/sortie disponibles ainsi qu'aux préférences de l'utilisateur [SCF+05]. Certains travaux mettent tout particulièrement l'accent sur les systèmes de réalité augmentée, ou de virtualité augmentée, comme par exemple dans les travaux de [BRC05] ou [STB04].
- L'adaptation des données concerne principalement l'adaptation aux capacités du réseau ou du terminal cible. Certains travaux de gestion des droits d'accès peuvent aussi entrer dans ce cadre. Les adaptations effectuées consistent en une modification du format, de la qualité, ou en une substitution de données originelles pour que la donnée fournie corresponde au contexte. De nombreux travaux portent sur ce domaine. On peut citer [LL02], [SB02] ou [BBP05].
- L'adaptation des services rendus est étudiée selon différents angles. Dans la communauté des composants, l'adaptation est vue comme une sélection du meilleur schéma d'interaction [CTLR06]. Ce sont les spécifications du composant en termes de fonctions et de connecteurs qui permettent de définir l'adéquation du composant. Dans le communauté des agents [ALP04], l'adaptation est vue comme une coopération d'éléments de décision pour la sélection et le déploiement de code. Dans la communauté intergiciels, l'adaptation est répartie entre les éléments de l'application et l'intergiciel lui-même [ABTB05]. En architectures orientées services, des propositions augmentent les descriptions des services d'un contexte pour assurer une meilleure sélection du service approprié [LVVOGM07], d'autres insèrent des wrappers permettant une adaptation dynamique du dialogue entre deux services, l'un client et l'autre serveur [BMT+07].

Certaines approches originales proposent des adaptations sur plusieurs facettes. Par exemple, [VO02] propose un système d'accès progressif aux informations et aux fonctionnalités par un système de masquage/démasquage progressif, ou en fonction des possiblités du système en regard des préférences utilisateur

[RVOGM06].

3.4.2 Points forts et travaux futurs

Dans le projet SEFAGI, nous avons défini une méthode de description d'interface graphiques qui peut être utilisée par tout utilisateur final, par le biais de notre assistant. Ces descriptions sont courtes, simples et donc rapides à construire. Elles sont également indépendantes de toute plate-forme cible. Un générateur de code permet ensuite de dériver des descriptions précédentes le code des fenêtres correspondantes, et ce pour plusieurs types de plates-formes. Nous avons aujourd'hui implanté trois types de plates-formes.
Le projet SECAS se positionne dans le domaine des applications pervasives, et propose une méthode et un prototype pour l'adaptation des applications à base de service selon trois facettes : les services, les données et les interfaces graphiques. A notre connaissance, c'est la seule plate-forme qui s'intéresse aux trois facettes de façon unifiée, les autres travaux se concentrant habituellement sur l'une ou l'autre des facettes. A partir d'une description succincte des services disponibles et d'un ensemble de règles d'adaptation, SECAS fournit automatiquement une vue de l'application correspondant à la situation contextuelle de chaque utilisateur. Pour cela, SECAS effectue des adaptations à chaud.

Les résultats de nos travaux sur l'adaptation et la gestion du contexte peuvent être résumés comme suit:
côté utilisateur:

- un outil simple et rapide de description d'interfaces graphiques (dans le cas de SEFAGI)
- une application auto-adaptative pouvant évoluer en fonction des changements de contexte de l'utilisateur

coté administrateur:

- une adaptation des applications dirigée par des langages déclaratifs et ne requiérant plus de développements spécifiques pour
  - les applications (modèle fonctionnel)
  - le contexte et les API des plates-formes
  - la politique d'adaptation (règles et opérateurs)
- une gestion des informations de contexte (couche gestion de contexte) standardisée et externalisée de l'application

On peut cependant regretter quelques points, et nous pourrons donc poursuivre nos investigations pour ces projets. Dans le cas de SEFAGI, nous n'avons pas implanté le générateur générique ni les descriptions externalisées de plates-formes conjointes, et ce par manque de temps et de moyens humains. De même, il faudrait généraliser la notion de composant graphique, pour l'étendre à des objets plus complets et peut-être plus complexes, se rapprochant par exemple des COMETs de [CCD+04].

Pour SECAS, le point qui me semble le plus limitant est le fait que notre architecture d'adaptation est pour l'instant centralisée, bien que fondée sur les services Web qui permettraient une décentralisation des adaptateurs et autres composants de la couche d'adaptation. Il faudra réfléchir à l'optimisation du placement des adaptateurs de services ou de données. Les réponses à ces questions vont dépendre entre autres de l'architecture de l'application adaptée, et

des droits d'accès aux serveurs hébergeant les services de cette application. En effet, dans certains cas il vaudra mieux localiser l'adaptateur sur la même machine que celle hébergeant le service correspondant, ou au contraire il faudra préférer centraliser les adaptateurs. Des techniques intermédiaires par proxies d'adaptation doivent aussi être envisagées. D'autre part, les ensembles d'opérateurs que nous avons définis ne sont certainement pas exhaustifs, et des tests sur des applications de plus plus grande envergure nous permettraient d'élargir les possibilités.

Nous avons également commencé un travail de formalisation plus avancée des composants d'une plate-forme d'adaptation telle que SECAS, ceci pour augmenter les capacités de validation des résultats des adaptations effectuées, mais également pour permettre des mises à jour à tous les niveaux en éliminant les remises en cause du code. Ainsi, le stage de master de Mohamed Zouari [Zou07] a abouti à la définition de trois ontologies : une ontologie de contexte, une ontologie de description de l'application, et une ontologie de description de la politique d'adaptation. Chaque ontologie a deux niveaux : un niveau générique indépendant de tout contexte, système d'adaptation ou application, et un niveau spécifique détaillant les spécificités de l'environnement utilisé. Un premier travail sur la validation des résultats d'adaptation a été fait ; il faut cependant approfondir ce travail. Il faudra également passer à une étape d'implantation des ontologies et des outils les manipulant, ce qui demandera, dans le cas de SECAS, des modifications non négligeables du code existant.

3.4.3 Vue synthétique

Le tableau 3.3 fournit une vue synthétique du travail effectué pour l'adaptation au contexte. Les informations suivantes sont répertoriées : participants et sujet de leur travail, financements associés, publications.

Thème, thésard rattaché, publications	**Sous-thème**	**Participant au sous-thème**	**Contrats**
SEFAGI [15, 8, 19, 24, 43]	générateur d'interfaces pour BD	PFE Chaari et Stefan 2002	SICOM 00-03
	état de l'art	biblio DESS Jeandot 2002	
	générateur d'interfaces sur services Web	M2R Chaari 2003	
	service d'accès à une base de données	mastère Cadot 2003	
	prototype pour J2ME CLDC MIDP	PFE Perez 2003	
	prototype pour J2ME CDC	PFE Elloumi 2004	
	services Web et application	M2R Tantravahi 2004	
	générateur générique	M2R Elloumi 2005	
SECAS thèse Chaari 2003-07 [1, 3, 6, 20, 21, 41, 42, 25]	adaptation de services	Pr invité A. Celentano 2004-05	
	prototype adaptation de services v0	PFE De Leon Perez 2005	
	exemples de services	PFE Richem et Branciard 2006	
	adaptation de services	PFE Zouari 2006	
	suivi de versions	M2R Doumat 2006	
	modélisation d'une application	M2R Fraga 2006	
	adaptation d'interfaces graphiques	PFE Sahnoun 2007	
	adaptation de contenu	PFE Prisacaru (encad. TChaari) 2007	
	politiques d'adaptation	M2R Zouari 2007	

TAB. 3.3 – *Adaptation au contexte : vue synthétique*

Chapitre 4

Conclusion

4.1 S'inspirer du domaine (bio)médical

Le dossier médical subit depuis ces quinze dernières années une grande transformation, qui n'est d'ailleurs toujours pas complètement terminée. Nous passons du dossier sous forme papier au dossier électronique. Cette transformation est nécessaire pour atteindre un double objectif : augmenter la qualité des soins tout en réduisant les coûts. En effet, l'informatisation du dossier médical permet d'assister le praticien dans sa démarche, de mieux suivre chaque patient dans son parcours de soins en transmettant efficacement les informations à tous les partenaires du parcours de soin... et donc de réduire les erreurs et les redondances. Il a aussi un intérêt d'un point de vue épidémiologique, puisque le traitement statistique des données recueillies à la source permet d'avoir une image fiable et à jour de la santé des populations. Il a finalement un intérêt économique car il permet de mieux évaluer l'activité des structures médicales (hôpitaux ou autres) ou même la pratique médicale.

Ce passage du support papier au support électronique est difficile. Outre les réticences des utilisateurs qui tendent néanmoins à disparaître, la complexité de la médecine et du para-médical rend l'exercice complexe. Ainsi, la médecine est encore loin d'être une science exacte, elle est très diversifiée (chaque spécialité et chaque corps de métier ayant ses méthodes et modes de travail), elle est très collaborative (le soin d'un patient n'est pas le fait exclusif d'une personne, mais d'une collaboration intra et inter-équipes), elle est sensible (les informations traitent de personnes et sont donc soumises à de nombreuses lois et réglementations nationales et internationales), elle peut être complexe (cas de patients multi-pathologiques, utilisation de nouvelles techniques, intégration d'essais thérapeutiques dans les soins)...

L'évolution du dossier médical est soumises aux évolutions constantes de la technologie, aux besoins des partenaires de santé, mais également aux directives des institutions qui ont joué un rôle moteur important ces dernières années. Les avancées massives en termes d'outils de communication et de mobilité (assistants personnels, réseaux voix et données, architectures distribuées) ont accéléré la montée en puissance de la télémédecine (hospitalisation à do-

micile, mobilité du praticien ou du patient). En France, les obligations de la carte vitale, du dossier médical personnel (DMP) etc. ont orienté et contraint fortement les propositions.

Ceci explique pourquoi les recherches en systèmes d'information peuvent se nourrir des besoins du domaine médical, et plus particulièrement des problématiques liées à l'informatisation du dossier médical. Les projets auxquels nous avons participé se sont tout particulièrement intéressés à la télémédecine (TIPHAD, COQUAS, SICOM et DocPatient), ou à l'informatisation de la filière d'urgences (HRP3). Ces projets nous ont permis de travailler sur la structuration des informations du dossier médical, sur la saisie de ces informations, sur leur transmission et sur leur restitution à l'utilisateur. Ces problématiques ont été traitées dans un cadre de partage d'information et de mobilité, ce qui soutient nos travaux sur l'adaptation de documents et leur partage, et sur l'adaptation au contexte dans des environnements pervasifs.

On peut également citer le projet SITRANS qui concerne le suivi des expériences à base de puces à ADN, et qui est donc plus lié au domaine biomédical qu'au dossier médical. La complexité et le volume des données manipulées a soulevé des problématiques de saisie et de restitution aux utilisateurs. Nous avons d'ailleurs proposé de travailler sur l'utilisation d'interfaces de navigation à base de topic maps, en s'inspirant d'un travail antérieur mené dans l'équipe pour la consultation du dossier médical. Nous avons publié deux articles concernant ces travaux [7, 45].

Nous avons aussi dû travailler sur des problématiques plus spécifiques, comme l'héritage au niveau des objets. Les systèmes à objets classiques conçoivent un héritage entre classes, qui consiste en un héritage structurel et fonctionnel de propriétés. Cet héritage n'est pas suffisant pour représenter toute la sémantique des classifications de termes médicaux. Prenons comme exemple la CIM-10, classification de maladies et problèmes de santé. Cette classification sert principalement à coder les diagnostics et symptômes d'un patient. Elle est habituellement représentée par une table relationnelle ou une classe objet et a pour attributs le code et le libellé de la classification (H80, Otosclerose). Cependant, cette classification est hiérarchique à trois niveaux (Chapitre maladies de l'oreille => H80 Otosclérose =>H80.2 Otosclérose cochléaire). La description d'une maladie au niveau le plus profond de la hiérarchie doit hériter de la description de sa maladie mère et récursivement jusqu'au premier niveau de l'arbre. Pour représenter ces niveaux hiérarchiques, nous avons dû définir un mécanisme d'héritage entre instances, et non entre classes [48, 54].

Ces travaux applicatifs nous ont amenés à publier dans des revues et conférences à mi-chemin entre les systèmes d'information et le domaine applicatif (Methods of Information in Medicine, Medical Infomatics - MedInfo, Medical Informatics Europe - MIE, Journées Francophones d'Informatique Médicale...). J'ai également co-organisé et co-édité le workshop Health Pervasive Services en 2006. J'ai co-organisé une session invitée "Information Systems" dans la conférence "Operational Research Applied to Health Services" en 2007. Je suis en cours de co-édition de deux numéros spéciaux de revues: un numéro spécial "Pervasive Health Care Services and Technologies" pour le journal *Internatio-*

nal Journal of Telemedicine and Applications prévu pour le printemps 2008 et un numéro spécial "Information Systems and Operational Research applied to Healthcare" pour le journal *Int. Journal International Journal of Healthcare Information Systems and Informatics* prévu pour l'automne 2008. Je participe également au GDR STIC-Santé conjoint au CNRS et à l'INSERM. J'ai également été membre de comités de programmes de conférences et workshops en systèmes d'information appliqués au domaine médical.

4.2 Perspectives

Dans cette section, je vais d'abord présenter le contexte dans lequel j'évolue. Je tracerai ensuite des perspectives de recherche.

La restructuration de notre laboratoire au 1er octobre 2006 a fait apparaitre une équipe "Systèmes d'Information Pervasifs" comprenant lors de sa création trois professeurs et trois maîtres de conférences venant d'horizons divers. Cette équipe avait une forte accointance avec le domaine d'application des systèmes d'information médicaux. Le départ du professeur André Flory et celui de Christine Verdier ont considérablement réduit les capacités d'encadrement de recherche appliquées au médical dans l'équipe et le laboratoire, pourtant réputé pour ses actions dans ce domaine d'application. En obtenant une habilitation à diriger des recherches, j'ai l'intention de renforcer mon activité sur les systèmes pervasifs, et de poursuivre leur application à la santé. Ce domaine est plus que jamais d'actualité, avec les fortes pressions gouvernementales à développer le partage des données médicales et la télémédecine en général. Le maintien de personnes à domicile, que ce soient des personnes malades ou dépendantes, est aussi un sujet social brûlant. Les avancées en systèmes pervasifs aideront à faire de ce maintien à domicile une solution plus sûre et plus adéquate à de nombreux cas. De façon plus large, la mobilité du professionnel de santé et du patient, la distance entre les différents acteurs requièrent de plus en plus de technologies des systèmes pervasifs.
Nous devons donc poursuivre nos travaux sur l'adaptation dans les systèmes pervasifs et leur application au domaine de la gestion de l'information médicale. J'espère également pouvoir susciter de l'engouement pour cette thématique parmi de jeunes maitres de conférences et ainsi permettre à la nouvelle équipe systèmes d'information pervasifs du LIRIS de prendre de l'ampleur.

Nous avons principalement étudié les systèmes pervasifs sous l'angle des services qu'ils procurent. Les services ont pu revêtir plusieurs formes : services logiciels de calcul, services actionneurs ayant une action sur le monde physique, services d'interrogation de l'environnement ou plus généralement services fournisseurs de données. Tous ces types de services peuvent être vus comme des boîtes noires fournissant des données sur sollicitation. Cette vision rejoint la notion de "dataspace" [FHM05] qui considère l'environnement de l'utilisateur comme un espace de données réparties dans des sources co-existantes. L'objectif des travaux dans le cadre des dataspace est de fournir un ensemble d'outils de base permettant le traitement uniforme d'un dataspace, quel que soit le degré de relations existant entre les différentes sources de cet espace. M. Franklin et ses co-auteurs ont identifié les nombreux challenges à relever dans ce cadre. On peut

y retrouver les trois thèmes que j'aimerais développer dans un avenir proche : la dynamicité, la répartition de l'adaptation et la prise en compte du contexte.

En ce qui concerne la dynamicité, on peut à juste titre remarquer que nos travaux actuels prennent en compte la dynamicité du contexte d'utilisation, mais plus difficilement celle des services et données disponibles, ou autrement dit l'apparition et la disparition de nouvelles sources. Dans ce cadre, deux thèmes peuvent être étudiés.
Tout d'abord, il faudra aller vers plus de dynamicité dans la découverte d'un nouvel environnement. Les systèmes pervasifs ont pour ambition de prendre en compte les apparitions et disparitions de services, qu'ils soient connus ou non à l'avance. Outre l'aspect description d'un nouveau service qu'il serait également intéressant d'étudier, nous avons commencé à travailler sur la construction déclarative d'applications pervasives. C'est l'objet du projet SoCQ (Services-oriented Continuous Queries) initié il y a un an et dont l'objectif est de permettre à l'utilisateur de construire une application ad hoc à la SQL, en se fondant sur une représentation homogène de toutes les sources de données disponibles dans l'environnement, que la source soit une base de données classique, un flux de données continu ou un service applicatif. Sur cette nouvelle thématique, le professeur Jean-Marc Petit et moi-même co-encadrons la thèse de Yann Gripay. Ce travail a déjà fait l'objet d'une publication dans une conférence nationale [18] et d'une présentation orale lors d'une journée du GDR I3 en juin 2007 [40]. Un premier prototype de système de gestion de données non conventionnelles a été réalisé. Nous développons actuellement un environnement-jeu pour tester les capacités de ce système.
Ensuite, l'intégration de nouvelles applications ou de nouvelles fonctionnalités dans un système d'information existant doit induire des procédures proches de la composition de services. Ce sujet est largement étudié dans la littérature. Cependant, je désire revoir les propositions actuelles pour une meilleure prise en compte du contexte. En effet, chaque service peut aujourd'hui être assorti d'un ensemble de couples attribut-valeur pour spécifier la situation contextuelle qui leur convient. Les travaux en modélisation du contexte proposent de plus en plus des outils et méthodes fondés sur les ontologies, et je pense que la richesse sémantique induite mérite d'être intégrée dans le "profil" des services à orchestrer.

Par ailleurs, nos propositions actuelles se fondent sur un outil d'adaptation centralisé. Cette centralisation a les avantages et les inconvénients habituels, à savoir, simplicité relative de mise en oeuvre, lourdeur et goulot d'étranglement... Pour pallier à ces inconvénients, une répartition de l'adaptation peut être envisagée. L'adaptation est en fait un processus à deux niveaux : un niveau de décision qui détermine l'ensemble des adaptations à appliquer, et un niveau de réalisation qui applique réellement l'adaptation déterminée. On peut voir la répartition de l'adaptation à ces deux niveaux. Distribuer la réalisation de l'adaptation permettrait d'effectuer les adaptations au plus proche des services applicatifs concernés, et ainsi réduire l'utilisation du réseau. Cela se fait dans d'autres domaines, comme par exemple dans certaines gestions de caches distribués ou dans des travaux sur les proxies. D'autre part, répartir la décision d'adaptation nécessiterait une réorganisation des décisions d'adaptation pour en extraire des parties locales et des parties distribuées. Les parties distribuées requièrent ensuite des procédures de négociation collaborative. Les nombreux

travaux en réseaux pair à pair devraient pouvoir servir de point de départ à la modélisation d'une telle décision. Des heuristiques dépendant des caractéristiques de l'application devraient également réduire le champ d'étude général.

La prise en compte du contexte mérite finalement notre attention. Outre l'adaptation pilotée par le contexte d'utilisation, nous devons également étudier le champ de l'adaptation pilotée par les données. Pour étayer mes propos, j'aimerais tout d'abord expliquer le type de projet dans lequel nous imaginons de tels travaux. Nous travaillons à la mise en place d'un projet concernant l'accès international au dossier médical. Ce projet se fonde sur l'émergence de centres régionaux regroupant l'ensemble des informations médicales des patients de la région. Ces centres régionaux ne sont accessibles qu'aux praticiens de la même région. Le but du projet est de permettre à tout médecin européen d'accéder au dossier régional d'un patient, notamment quand intervient un incident de santé lors d'un voyage. Outre les problématiques d'architecture et de sécurité, nous traiterons tout particulièrement les questions relatives à l'adaptation du contenu du document. Cette adaptation concerne par exemple la traduction (passage d'une langue à une autre), mais également le traitement des équivalences de termes du domaine (médicaments non internationaux remplacés par des produits similaires par exemple). Pour aller plus loin, une étape d'adaptation aux données serait utile. En effet, il va falloir effectuer un résumé de l'information stockée dans le centre régional. Ce résumé devra être pilotable par le cas pathologique du patient (histoire médicale et problème actuel), c'est-à-dire par les données.

Chapitre 5

Bibliographie

5.1 Consultation et saisie adaptées d'informations

[Ade98] Brad Adelberg. Nodose - a tool for semi-automatically extracting semi-structured data from text documents. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 283–294. ACM Press, 1998.

[AEG00] Eugene Agichtein, Eleazar Eskin, and Luis Gravano. Combining strategies for extracting relations from text collections. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 86–95, 2000.

[AKS99] Naveen Ashish, Craig A. Knoblock, and Cyrus Shahabi. Selectively materializing data in mediators by analyzing user queries. In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems, Edinburgh, Scotland, September 2-4, 1999*, pages 256–266. IEEE Computer Society, 1999.

[AQM+97] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[Bad03] Youakim Badr. *Couplage Documents et Bases de données : Etude et réalisation.* PhD thesis, INSA de Lyon, 2003.

[BBB00] Ron Bourret, Christof Bornhövd, and Alejandro P. Buchmann. A generic load/extract utility for data transfer between xml documents and relational databases. In *WECWIS*, pages 134–143, 2000.

[BDGR97] Edouard Bugnion, Scott Devine, Kinshuk Govil, and Mendel Rosenblum. Disco: Running commodity operating systems on scalable multiprocessors. *ACM Trans. Comput. Syst.*, 15(4):412–447, 1997.

[BFS00] Peter Buneman, Mary F. Fernandez, and Dan Suciu. Unql: A query language and algebra for semistructured data based on structural recursion. *VLDB J.*, 9(1):76–110, 2000.

[BGL+99] Chaitanya K. Baru, Amarnath Gupta, Bertram Ludäscher, Richard Marciano, Yannis Papakonstantinou, Pavel Velikhov, and Vincent Chu. Xml-based information mediation with mix. In Alex Delis,

Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 597–599. ACM Press, 1999.

[Bou05] Mohamed Salah Boumediene. *Définition d'un système générique de partage de données entre systèmes existants.* PhD thesis, INSA de Lyon, 2005.

[Bri06] Sandra Bringay. *Les annotations pour supporter la collaboration dans le dossier patient électronique.* PhD thesis, Université de Picardie Jules Verne, 2006.

[CFR+01] Donald D. Chamberlin, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu (ed.). Xquery: A query language for xml, 2001.

[CMM01] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 109–118. Morgan Kaufmann, 2001.

[DBO05] Cécile Frérot Marie-Paule Jacques Didier Bourigault, Cécile Fabre and Sylwia Ozdowska. Syntex, analyseur syntaxique de corpus. In *Actes des 12èmes journées sur le Traitement Automatique des Langues Naturelles, Dourdan, Franc*, volume 2, pages 17–20, 2005.

[DFS99] Alin Deutsch, Mary F. Fernandez, and Dan Suciu. Storing semistructured data with stored. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 431–442. ACM Press, 1999.

[ECJ+99] David W. Embley, Douglas M. Campbell, Y. S. Jiang, Stephen W. Liddle, Yiu-Kai Ng, Dallan Quass, and Randy D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng.*, 31(3):227–251, 1999.

[FFK+98] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy, and Dan Suciu. Catching the boat with strudel: Experiences with a web-site management system. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 414–425. ACM Press, 1998.

[FRV96] Daniela Florescu, Louiqa Raschid, and Patrick Valduriez. A methodology for query reformulation in cis using semantic knowledge. *Int. J. Cooperative Inf. Syst.*, 5(4):431–468, 1996.

[Gro89] Maurice Gross. The use of finite automata in the lexical representaion of natural language. In *Proceedings of the LITP Spring School on Theoretical Computer Science*, pages 34–50. Springer-Verlag, 1989.

[IFF+99] Zachary G. Ives, Daniela Florescu, Marc Friedman, Alon Y. Levy, and Daniel S. Weld. An adaptive query execution system for data

integration. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 299–310. ACM Press, 1999.

[ISO02] ISO. Topic maps, document description and processing language, 2nd ed., 2002.

[JCB98] Vincent Brunie Sawsan El Kassar Pierre Zweigenbaum Jean Charlet, Bruno Bachimont and Jean-François Boisvieux. Hospitexte: towards a document-based hypertextual electronic medical record. *Journal of the American Medical Informatics Association*, 5:713–717, 1998.

[Kar01] Lauri Karttunen. Applications of finite-state transducers in natural language processing. *Implementation and Application of Automata : 5th International Conference, CIAA 2000, London, Ontario, Canada, July 24-25, 2000, Lecture Notes in Computer Science*, 2088:34–46, 2001.

[MIKS00] Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, and Amit P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[Mus99] Ion Muslea. xtraction patterns for information extraction tasks: A survey. In Mary E. Califf, editor, *AAAI-99 Workshop on Machine Learning for Information Extraction, Orlando, Florida.* american association for artificial intelligence, 1999.

[Ouz04] Mourad Ouziri. *Utilisation des Topic Maps pour l'interrogation et la génération de documents virtuels : Application au domaine médical.* PhD thesis, INSA de Lyon, 2004.

[Rou02] Marie-Christine Rousset. Knowledge representation for information integration. In Mohand-Said Hacid, Zbigniew W. Ras, Djamel A. Zighed, and Yves Kodratoff, editors, *Foundations of Intelligent Systems, 13th International Symposium, ISMIS 2002, Lyon, France, June 27-29, 2002, Proceedings*, volume 2366 of *Lecture Notes in Computer Science*, pages 1–3. Springer, 2002.

[RS97] Mary Tork Roth and Peter M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 266–275. Morgan Kaufmann, 1997.

[Sil00] Max Silberztein. Intex: An fst toolbox. *Theor. Comput. Sci.*, 231(1):33–46, 2000.

[SPB05] Ludwig Seitz, Jean-Marc Pierson, and Lionel Brunie. Encrypted Storage of Medical Data on a Grid. *Methods of Information in Medicine*, (2):198–202, jan 2005.

[Wie96] Gio Wiederhold. Glossary: Intelligent integration of information. *J. Intell. Inf. Syst.*, 6(2/3):281–291, 1996.

5.2 Adaptation au contexte

[ABTB05] Dhouha Ayed, Nabiha Belhanafi, Chantal Taconet, and Guy Bernard. Deployment of component-based applications on top of a context-aware middleware. In Peter Kokol, editor, *IASTED Conf. on Software Engineering. Innsbruck, Austria, February 15-17, 2005*, pages 414–419. IASTED/ACTA Press, 2005.

[all04] The OSGi alliance. Osgi service platform core specification and service compendium, release 4. http://www.osgi.org/, August 2004.

[ALP04] Jean-Paul Arcangeli, Sebastien Leriche, and Marc Pantel. Development of flexible peer-to-peer information systems using adaptable mobile agents. In *15th International Workshop on Database and Expert Systems Applications (DEXA 2004), with CD-ROM, 30 August - 3 September 2004, Zaragoza, Spain*, pages 549–553. IEEE Computer Society, 2004.

[AM02] Helms J. Abrams M. User interface markup language (uiml) 3.0 specification draft, , document version 08 february 2002, 2002.

[BBP05] Girma Berhe, Lionel Brunie, and Jean-Marc Pierson. Distributed content adaptation for pervasive systems. In *International Symposium on Information Technology: Coding and Computing (ITCC 2005), Volume 2, 4-6 April 2005, Las Vegas, Nevada, USA*, pages 234–241. IEEE Computer Society, 2005.

[BCvH+03] Jonathan Billington, Søren Christensen, Kees M. van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno, and Michael Weber. The petri net markup language: Concepts, technology, and tools. In Wil M. P. van der Aalst and Eike Best, editors, *ICATPN*, volume 2679 of *Lecture Notes in Computer Science*, pages 483–505. Springer, 2003.

[BMT+07] Djamal Benslimane, Zakaria Maamar, Yehia Taher, Mohammed Lahkim, Marie-Christine Fauvet, and Michael Mrissa. A multilayer and multi-perspective approach to compose web services. In *21st International Conference on Advanced Information Networking and Applications (AINA 2007), May 21-23, 2007, Niagara Falls, Canada*, pages 31–37. IEEE Computer Society, 2007.

[BRC05] Oliver Brdiczka, Patrick Reignier, and James L. Crowley. Automatic development of an abstract context model for an intelligent environment. In *3rd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops), 8-12 March 2005, Kauai Island, HI, USA*, pages 35–39. IEEE Computer Society, 2005.

[CCD+04] Gaëlle Calvary, Joëlle Coutaz, Olfa Dâassi, Lionel Balme, and Alexandre Demeure. Towards a new generation of widgets for supporting software plasticity: The "comet". In Rémi Bastide, Philippe A. Palanque, and Jörg Roth, editors, *Engineering Human Computer Interaction and Interactive Systems, Joint Working Conferences EHCI-DSVIS 2004, Hamburg, Germany, July 11-13, 2004, Revised Selected Papers*, volume 3425 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2004.

[CCDG05] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. Context is key. *Commun. ACM*, 48(3):49–53, 2005.

[CCT+03] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[Cha07] Tarak Chaari. *Adaptation d'applciations pervasives dans des environnements multi-contextes*. PhD thesis, INSA de Lyon, 2007.

[CTLR06] Daniel Cheung, Jean-Yves Tigli, Stephane Lavirotte, and Michel Riveill. Wcomp: a multi-design approach for prototyping applications using heterogeneous resources. In *17th IEEE International Workshop on Rapid System Prototyping (RSP 2006), 14-16 June 2006, Chania, Crete, Greece*, pages 119–125. IEEE Computer Society, 2006.

[CV05] Adrien Coyette and Jean Vanderdonckt. A sketching tool for designing anyuser, anyplatform, anywhere user interfaces. In *Human-Computer Interaction - INTERACT 2005, IFIP TC13 International Conference, Rome, Italy, September 12-16, 2005, Proceedings*, volume 3585 of *Lecture Notes in Computer Science*, pages 550–564. Springer, 2005.

[DSA01] Anind K. Dey, D. Salber, and G.D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCI) Journal*, 16(2-4):97–166, 2001.

[FHM05] Michael J. Franklin, Alon Y. Halevy, and David Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34(4):27–33, 2005.

[HIR02] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In Friedemann Mattern and Mahmoud Naghshineh, editors, *Pervasive Computing, First International Conference, Pervasive 2002, Zürich, Switzerland, August 26-28, 2002, Proceedings*, volume 2414 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2002.

[JW03] Seiie Jang and Woontack Woo. Ubi-ucam: A unified context-aware application model. In Patrick Blackburn, Chiara Ghidini, Roy M. Turner, and Fausto Giunchiglia, editors, *Modeling and Using Context, 4th International and Interdisciplinary Conference, CONTEXT 2003, Stanford, CA, USA, June 23-25, 2003, Proceedings*, volume 2680 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2003.

[LL02] Wai Yip Lum and Francis C. M. Lau. On balancing between transcoding overhead and spatial consumption in content adaptation. In Ian F. Akyildiz, Jason Yi-Bing Lin, Ravi Jain, Vaduvur Bharghavan, and Andrew T. Campbell, editors, *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking, MOBICOM 2002, Atlanta, Georgia, USA, September 23-28, 2002*, pages 239–250. ACM, 2002.

[LVVOGM07] Céline Lopez-Velasco, Marlène Villanova-Oliver, Jérôme Gensel, and Hervé Martin. Registre de services web pour le développement d'applications. In *Actes du XXVème Congrès INFORSID, Perros-Guirec, France, 22 au 25 mai 2007*, pages 521–536, 2007.

[MON90] J. Morejon, R. Oudrhiri, and Pascal Negros. Etude d'un niveau méta de spécification et de génération dans un outil d'aide à la conception. In Patrick Valduriez, editor, *Sixièmes Journées Bases de Données Avancées, 25-28 Septembre 1990, Montpellier (Informal Proceedings)*, pages 103–119, 1990.

[PDFP03] Anne-Marie Pinna-Dery, Jérémy Fierstone, and Emmanuel Picard. Component model and programming: A first step to manage human computer interaction adaptation. In Luca Chittaro, editor, *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI 2003, Udine, Italy, September 8-11, 2003, Proceedings*, volume 2795 of *Lecture Notes in Computer Science*, pages 456–465. Springer, 2003.

[PE02] Angel Puerta and Jacob Eisenstein. Ximl: a common representation for interaction data. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 214–215, New York, NY, USA, 2002. ACM Press.

[Pin06] Manuele Kirsch Pinheiro. *Adaptation contextuelle et personnalisée de l'information de conscience de groupe au sein des systèmes d'information coopératifs.* PhD thesis, Université Joseph Fourier - Grenoble I, 2006.

[Rou03] José Rouillard. Plastic ml and its toolkit. In *Conference HCI International 2003. Heraklion, Crete, Greece, 2003*, volume 4, pages 612–616. Lawrence Erlbaum associates, 2003.

[RVOGM06] Angela Carrillo Ramos, Marlène Villanova-Oliver, Jérôme Gensel, and Hervé Martin. Gestion des préférences utilisateurs pour les systèmes d'information ubiquitaires. In *Actes du XXIVème Congrès INFORSID, Hammamet, Tunisie, 31 mai - 4 juin, 2006*, pages 767–782, 2006.

[Sat01] Mahadev Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[SB02] Kevin Kamel Andrzej Kochut Christopher Kommareddy Tamer Nadeem Pankaj Thakkar Bao Trinh Adel M. Youssef Moustafa Youssef Ronald L. Larsen A. Udaya Shankar Ashok K. Agrawala Suman Banerjee, Sulabh Agarwal. Rover: Scalable location-aware computing. *IEEE Computer*, 35(10):46–53, 2002.

[SCF+05] Jean-Sebastien Sottet, Gaëlle Calvary, Jean-Marie Favre, Joëlle Coutaz, Alexandre Demeure, and Lionel Balme. Towards model driven engineering of plastic user interfaces. In Jean-Michel Bruel, editor, *Satellite Events at the MoDELS 2005 Conference, MoDELS 2005 International Workshops, Doctoral Symposium, Educators Symposium, Montego Bay, Jamaica, October 2-7, 2005, Revised Selected Papers*, volume 3844 of *Lecture Notes in Computer Science*, pages 191–200. Springer, 2005.

[SDA99] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pages 434–441, 1999.

[SLPF03] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool: A context ontology language to enable contextual interoperability. In Jean-Bernard Stefani, Isabelle M. Demeure, and Daniel Hagimont, editors, *Distributed Applications and Interoperable Systems, 4th IFIP WG6.1 International Conference, DAIS 2003, Paris, France, November 17-21, 2003, Proceedings*, volume 2893 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2003.

[SM03] D. Saha and A. Mukherjee. Pervasive computing: a paradigm for the 21st century. *Computer, IEEE*, 36(3):25–31, 2003.

[STB04] Kinan Samaan and Franck Tarpin-Bernard. Task models and interaction models in a multiple user interfaces generation process. In Pavel Slavík and Philippe A. Palanque, editors, *Task Models and Diagrams for User Interface Design: Proceedings of the Third International Workshop on Task Models and Diagrams for User Interface Design - TAMODIA 2004, November 15 - 16, 2004, Prague, Czech Republic*, pages 137–144. ACM, 2004.

[sun07] sun. Java me technology apis and docs, 2007.

[VO02] Marlène Villanova-Oliver. *Adaptabilité dans les systèmes d'information sur le web : modélisation et mise en oeuvre de l'accès progressif.* PhD thesis, Université Joseph Fourier - Grenoble I, 2002.

[Wei91] Mark Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, 1991.

[WZGP04] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), 14-17 March 2004, Orlando, FL, USA*, pages 18–22. IEEE Computer Society, 2004.

[Zou07] Mohamed Zouari. Politique d'adaptation d'applications dans les systèmes pervasifs. Master's thesis, INSA de Lyon, 2007.

Chapitre 6

Curriculum Vitae

Ce Curriculum Vitae décrit mes activités de recherche depuis la soutenance de ma thèse. Il concerne donc la période 1997-2007, hormis mes deux congés maternité!

6.1 Emplois

Depuis septembre 1998 Maître de conférences 27ème section
INSA Lyon Dpt Télécommunications, services & usages
Laboratoire LIRIS, axe Systèmes d'information communicants, équipe Systèmes d'information pervasifs
Titulaire de la PEDR depuis 2004

1996-97 et 1997-98 Attachée Temporaire d'Enseignement et de Recherche 27ème section
IUT Informatique Bourg en Bresse
Laboratoire LISI

1993-96 Allocataire de Recherche MRT
Laboratoire LISI

6.2 Diplômes

Janvier 1997 Docteur en Informatique
INSA Lyon
"Modèles et interfaces génériques : application aux systèmes d'information médicaux"
Directeur de thèse : Professeur André Flory
Jury : Pr Claude Chrisment, Pr André Flory, Pr Colette Rolland (rapporteur), Pr Paul Rubel, Pr Michel Schneider (rapporteur), Pr Alain Vénot
Mention Très honorable

Juillet 1993 DEA Informatique
DEA Informatique de Lyon
"Etude d'un héritage au niveau des objets : application au dossier médical"
stage de DEA/PFE effectué au LISI, INSA de Lyon, sous la direction du Professeur André FLORY
Mention Bien

Juin 1993 Ingénieur en Informatique
INSA Lyon

6.3 Publications

Une vue synthétique de mes publications est fournie dans le tableau 6.1. Les nombres en parenthèses comptent mes publications se référant à mon travail de thèse. Parmi ces publications, on trouvera des articles dans des revues ou conférences purement informatiques. D'autres ont une audience dédiée à l'informatique (bio)médicale ; ces publications émanent de contrats de recherche associant des praticiens de la santé ou des biologistes, qui nous ont amenés à prouver l'intérêt de nos propositions dans le cadre applicatif.

5 articles dans des revues internationales
9 (+4) publications dans des conférences internationales
3 (+2) articles dans des revues nationales
6 (+2) publications dans des conférences nationales
5 (+1) publications dans des workshops
2 chapitres de livres
4 éditions invitées
2 manuels
7 (+5) autres

TAB. 6.1 – *Synthèse des publications (les publications relatives à mes travaux de thèse sont entre parenthèses)*

6.3.1 Articles dans des revues internationales

[1] Tarak Chaari, Frédérique Laforest, and Augusto Celentano. Adaptation in Context-Aware Pervasive Information Systems: The SECAS Project. *Int. Journal on Pervasive Computing and Communications(IJPCC)*, 3(3-4), 2007.

[2] Frédérique Laforest and Christine Verdier. Documents and topic maps: an original way to manage medical records. *International Journal of Healthcare Information Systems and Informatics (IJHISI) Special Issue on Health Information Linkage and Integration*, oct-dec 07, 2007.

[3] Tarak Chaari, Dejene Ejigu, Frédérique Laforest, and Vasile-Marian Scuturici. A comprehensive approach to model and use context for adapting applications in pervasive environments. *Int. Journal on Systems and Software*, 3, 2007.

[4] Frédérique Laforest and André Flory. Using weakly structured documents to fill in a classical database. *J. Database Manag.*, 12(2):3–13, 2001.

[5] Stéphane Frenot and Frédérique Laforest. Medical record management systems : Criticisms and new perspectives. *Methods of Information in Medicine*, 38(2):89–95, 1999.

6.3.2 Publications dans des conférences internationales

[6] Tarak Chaari, Dejene Ejigu, Frédérique Laforest, and Vasile-Marian Scuturici. Modeling and using context in adapting applications to pervasive environments. In *Proceedings of the IEEE Int. Conference on Pervasive Services (ICPS'06)*. IEEE, 2006.

[7] Frédérique Laforest, Anne Tchounikine, Tarak Chaari, Hubert Charles, and Federica Calevro. Sitrans: A web information system for microrray experiments. In *Proceedings of the Conference on Medical Informatics Europe (MIE'05)*, 2005.

[8] Tarak Chaari and Frédérique Laforest. Sefagi: Simple environment for adaptable graphical interfaces - generating user interfaces for different kinds of terminals. In *Proceedings of the Int. Conf. on Enterprise Information Systems (ICEIS'05), Miami, USA*, pages 232–237, 2005.

[9] Youakim Badr, Frédérique Laforest, and André Flory. Druid: Coupling user written documents and databases. In *Proceedings of the Int. Conf. on Enterprise Information Systems (ICEIS'03), Angers, France*, pages 191–196, 2003.

[10] Youakim Badr, Frédérique Laforest, André Flory, and Marguerite Sayah. Transformation rules from semi-structured xml documents to database model. In *Int. Conf. on Computer Systems and Applications (AICC-SA'01)*, pages 181–184. IEEE Computer Society, 2001.

[11] Frédérique Laforest and André Flory. Medical records and electronic documents: a proposal. In *Proceedings of the Int. Conf. on Medical Informatics (MedInfo'01), London, England*, 2001.

[12] Frédérique Laforest and André Flory. The electronic medical record: Using documents for information capture. In *Proceedings of the Conf. on Medical Informatics Europe (MIE'00), Hannover, Germany*, 2000.

[13] Frédérique Laforest and Anne Tchounikine. A model for querying annotated documents. In Johann Eder, Ivan Rozman, and Tatjana Welzer, editors, *Advances in Databases and Information Systems, Third East European Conference (ADBIS'99), Maribor, Slovenia*, volume 1691 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1999.

[14] André Flory, Stéphane Frénot, and Frédérique Laforest. An intelligent system for drug prescription support. In *Proceedings of the int. Conference on Intelligent Systems (ICIS'99), Denver, Colorado, USA*, 1999.

6.3.3 Articles dans des revues nationales

[15] Tarak Chaari and Frédérique Laforest. Génération et adaptation automatiques des interfaces utilisateurs pour des environnements multiterminaux. le projet sefagi. *Ingénierie des Systèmes d'Information, n° spécial Systèmes d'information pervasifs*, 9(2):11–38, 2004.

[16] Frédérique Laforest and Youakim Badr. Modèle de couplage de documents structurés et de bases de données. le projet druid. *Ingénierie des Systèmes d'Information, n° spécial Bases de données semi-structurées*, 8(5-6):109–126, 2003.

[17] Frédérique Laforest, Stéphane Frénot, and Nada Al Masri. Dossier médical semi-structuré pour des interfaces de saisie multi-modales. *Documents Numériques, n° spécial dossiers numériques*, 6(1-2):29–46, 2002.

6.3.4 Publications dans des conférences nationales

[18] Yann Gripay, Frédérique Laforest, and Jean-Marc Petit. Towards Action-Oriented Continuous Queries in Pervasive Systems. In *Bases de données Avancées 2007 (BDA'07)*, October 2007.

[19] Frédérique Laforest. Interfaces adaptatives. In *Actes de la Conf. Gestion et Ingénierie des Systèmes Hospitaliers (GISEH'06), Luxembourg*, 2006.

[20] Tarak Chaari, Frédérique Laforest, and André Flory. Adaptation des applications au contexte en utilisant les services web : le projet secas. In *Actes de la Conf. Ubiquité et Mobilité (UbiMob 2006), Grenoble, France*, 2006.

[21] Tarak Chaari, Frédérique Laforest, and André Flory. Adaptation des applications médicales à des contextes multiples. In *Actes des journées francophones d'informatique médicale (JFIM 2005), Lille, France*, 2005.

[22] Nathalie Bricon-Souf, Marie-Catherine Beuscart-Zéphir, Frédérique Laforest, Harry Karadimas, Françoise Anceaux, André Flory, Eric Lepage, and Régis Beuscart. Tiphad : Technologies de l'information pour l'hospitalisation à domicile. In P.; Venot A.; Degoulet P. Beuscart, R.; Zweigenbaum, editor, *Télémédecine et e-santé*, volume 13. Springer, 2002.

[23] André Flory, Frédérique Laforest, and Arnaud Weill. Utilisation des documents semi-structurés pour la représentation et le stockage du dossier médical. In R. Beuscart R. Baud M. Fieschi, O. Bouhaddou, editor, *L'informatique au service du patient*, volume 12, pages 229–240. Springer, 2000.

6.3.5 Publications dans des workshops

[24] Tarak Chaari, Brahim Elloumi, and Frédérique Laforest. A generic description language for the automatic generation of pervasive medical user interfaces: the sefagi project. In *ICPS Workshop : Health Pervasive Services (HPS 2006), Lyon, France*, 2006.

[25] Tarak Chaari, Frédérique Laforest, and Augusto Celentano. Service-oriented context-aware application design. In *Proceedings of the First International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP 2005)*, 2005.

[26] Frédérique Laforest and Mohamed Boumédiene. Study of the automatic construction of xml documents models from a relational data model. In *DEXA Workshop : Web Semantics (WebS 2003), Prague, Czech Republic*, pages 566–570. IEEE Computer Society, 2003.

[27] Youakim Badr, Marguerite Sayah, Frédérique Laforest, and André Flory. Capturing data using xml paragraph-centric document. In Maurizio Len-

zerini, Daniele Nardi, Werner Nutt, and Dan Suciu, editors, *VLDB Workshop : 8th International Workshop on Knowledge Representation meets Databases (KRDB 2001), Roma, Italy*, volume 45 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

[28] Frédérique Laforest and Anne Tchounikine. Indexing semi-structured documents for context-based information retrieval in a medical information system. In *DEXA Workshop : First International Workshop on Document Analysis and Understanding for Document Databases (DAUDD 1999), Florence, Italy*, pages 593–597, 1999.

6.3.6 Chapitres de livres

[29] Frédérique Laforest and André Flory. Using weakly structured documents at the user-interface level to fill in a classical database. In *Advanced Topics in Database Research, Vol. 1*, pages 190–210. IDEA Group publishing, 2002.

[30] Frédérique Laforest and André Flory. The electronic medical record: Using documents for information capture. In *Studies in Health Technologies of Information*, volume 77, pages 617–621. IOS Press, 2000.

6.3.7 Editeur invité

[31] Paul Harper and Frédérique Laforest, editors. *Joint advances in Information Systems and Operational Research applied to Healthcare Special Issue of the Journal of Healthcare Information Systems and Informatics (IJHISI)*. IGI publishing, automn 2008.

[32] Laurence Yang, Sajid Hussain, Frédérique Laforest, and Christine Verdier, editors. *Pervasive Health Care Services and Technologies Special Issue of the on-line International Journal of Telemedicine and Applications (IJTA)*. Hindawi Journals, spring 2008.

[33] Frédérique Laforest and Frédéric Le Mouël, editors. *n° spécial Adaptation et gestion du contexte de la revue Ingénierie des Systèmes d'Information (ISI)*. Hermès - Lavoisier, 2006.

[34] Frédérique Laforest and Christine Verdier, editors. *Proceedings of the 1st International Workshop on Health Pervasive Services (HPS), Lyon, France*. IEEE, 2006.

6.3.8 Manuels

[35] R. Dayan, editor. *Manuel de gestion*, volume 1 & 2. Agence Universaitaire Francophone, 2° edition, 2005.

[36] André Flory and Frédérique Laforest. *Les bases de données relationnelles*. Economica, 3° edition, 2005.

[37] André Flory and Frédérique Laforest. *Les bases de données relationnelles*. Economica, 2° edition, 2002.

[38] R. Dayan, editor. *Manuel de gestion*, volume 1 & 2. Agence Universaitaire Francophone, 1° edition, 1999.

[39] André Flory and Frédérique Laforest. *Les bases de données relationnelles*. Economica, 1° edition, 1996.

6.3.9 Autres communications

[40] Yann Gripay, Frédérique Laforest, and Jean-Marc Petit. Traitement de requêtes sur données non conventionnelles (données, services, flux). In *Journées du thème Masses de donnees et accès à l'information du GDR I3*, juin 2007.

[41] Tarak Chaari and Frédérique Laforest. Adaptation d'applications à de nouveaux contextes d'utilisations: le projet secas. In *Journée de l'action Adaptation du GDR ASR du CNRS*, octobre 2006.

[42] Tarak Chaari and Frédérique Laforest. L'adaptation dans les systèmes d'information sensibles au contexte d'utilisation : approche et modèles. In *Actes de la conf. Génie Electrique et Informatique (GEI 2005), Sousse, 2005*, pages 56–61, 2006.

[43] Tarak Chaari, Frédérique Laforest, and André Flory. Génération automatique d'interfaces graphiques pour la saisie et la consultation de données : application au domaine médical. In *Proceedings of the International Conf. on Sciences of Electronic, Technology of Information and Telecommunications (SETIT 2004), Sousse, Tunisia*, 2004.

[44] Mohamed Boumediene and Frédérique Laforest. Contrôle d'accès aux documents xml en filtrant les dtd. In *poster at the International Conf. on Sciences of Electronic, Technology of Information and Telecommunications (SETIT 2004), Sousse, Tunisia*, 2004.

[45] Hubert Charles, Jean-Michel Fayard, Frédérique Laforest, and Anne Tchounikine. Système d'information transcriptome pour la plate-forme de la génopole rhône-alpes. In *poster aux Journées de post-génomique de la Doua (JPGD 2002), Lyon, France*, 2002.

[46] Frédérique Laforest. Druid : Document and rule-based user interface for databases. In *Journée du Groupe de Travail Documents Multimédia du GDR I3*, juin 2001.

6.3.10 Publications se reportant à mon travail de thèse

Journaux et revues

[47] Frédérique Laforest André Flory and Christine Verdier. Propositions pour un système de gestion de l'information médicale. *Les Cahiers de Sociologie et Démographie Médicales*, 36(3):281–309, 1996.

[48] Frédérique Laforest and Christine Verdier. Héritage entre objets. *Ingénierie des Systèmes d'Information*, 3(2-3):327–347, 1995.

Conférences internationales et nationales

[49] André Flory, Stéphane Frénot, and Frédérique Laforest. A hypermedia-based medical records management system. In *Proceedings of the Int. Conf. on Medical Informatics (MedInfo'98), Seoul, Korea*, 1998.

[50] Frédérique Laforest, Stéphane Frénot, and André Flory. A new approach for hypermedia medical records management. In *Proceedings of the Conf. on Medical Informatics Europe (MIE'96), Copenhagen, Denmark*, 1996.

[51] Frédérique Laforest and André Flory. The impact of new technologies on future medical information systems. In *Proceedings of the 6th Int Conf of System Science in Health Care (SSHC'96), Barcelona, Spain*, 1996.

[52] Stéphane Frénot, Frédérique Laforest, and André Flory. An intelligent system to help expert users : application to drug prescription. In *Proceedings of the Int conf on Artificial Intelligence and Expert Systems Applications (EXPERSYS'95), San Francisco, USA*, pages 61–66, 1995.

[53] Stéphane Frénot and Frédérique Laforest. Un système de gestion d'architextes. In *Actes du Congrès Informatique des organisations et systèmes d'information et de décision (INFORSID'98), Montpellier, France, 12-15 mai, 1998*, pages 305–325, 1998.

[54] Frédérique Laforest and Christine Verdier. Héritage entre objets : application à la définition de l'information médicale. In *Actes du Congrès Informatique des organisations et systèmes d'information et de décision (INFORSID'94), Aix en Provence, France*, pages 435–450, 1994.

Autres publications

[55] Frédérique Laforest. Generic models : a new approach for information systems design. In *3rd Basque International Workshop on Information Technology (BIWIT'97), Biarritz, France*, pages 189–196, 1997.

[56] Frederique Laforest. *Modèles et interfaces génériques - Application aux systèmes d'information médicaux.* PhD thesis, INSA de Lyon, janvier 1997.

[57] Frédérique Laforest. Les logiciels médicaux génériques : Vers la prise en compte d'une spécificité de la médecine. In *Colloque de l'Association Rhodanienne pour l'Avancement de l'Econométrie (ARAE), Villeurbanne, France*, 1996.

[58] André Flory, Stéphane Frénot, and Frédérique Laforest. Une nouvelle approche pour la présentation multimédia de l'information médicale. In *Journée du Collège Rhône-Alpin de l'Information Médicale (CRAIM), Valence, France*, 1996.

[59] André Flory, Frédérique Laforest, and Christine Verdier. La dimension temporelle dans l'information médicale. In *journées annuelles de la Société Suisse d'Informatique Médicale : Quelle information pour des soins de qualité?, OMS, Genève, Suisse*, pages 40–44, 1995.

[60] Frédérique Laforest. Etude d'un héritage au niveau des objets : application au dossier médical. Master's thesis, INSA de Lyon, 1993.

6.4 Encadrements de recherche

Je conçois la recherche comme un travail collaboratif, regroupant différents types de participants (professeurs, chercheurs permanents, doctorants, stagiaires de master ou autres) rassemblés autour d'une problématique commune, avec chacun des objectifs personnels. J'organise des réunions de projet très fréquentes (hebdomadaires ou bimensuelles) pour que les travaux des uns puissent être compris et critiqués par les autres. Les sections suivantes donnent la liste détaillée de ces encadrements.

6.4.1 Thèses de doctorat co-encadrées

Depuis le début de ma carrière de maître de conférences, je co-encadre un à deux doctorants en parallèle. Je suis co-auteur de toutes les publications rédigées sur ces travaux de thèse.

Date	**Etudiant**	**Situation actuelle**
sept 07	Tarak Chaari	post-doc au LAAS, Toulouse
déc 05	Mohamed Boumediene	ingénieur développement, Lyon
déc 03	Youakim Badr	maître de conférences à l'INSA de Lyon
En cours	**Etudiant**	
06-09	Yann Gripay	

TAB. 6.2 – *Thèses co-encadrées*

Thèses soutenues

Septembre 2007 Tarak Chaari
Thèse en co-encadrement avec le Pr André Flory. Bourse du gouvernement Tunisien (2 ans) puis ATER.
Titre : "Adaptation d'applications pervasives dans des environnements multi-contextes".

Décembre 2005 Mohamed Boumediene
Thèse en co-encadrement avec le Pr André Flory. Bourse de la Région Rhône-Alpes dans le cadre du projet SICOM.
Titre: "Définition d'un système générique de partage de données entre systèmes existants".

Décembre 2003 Youakim Badr
Thèse en co-encadrement avec le Pr André Flory. Ingénieur réseau à l'université de Beyrouth, Liban puis ATER.
Titre: "Couplage Documents et Bases de données : Etude et réalisation".

Thèse en cours

Soutenance prévue en 2009 Yann Gripay.
Thèse en co-encadrement avec le Pr Jean-Marc Petit. Bourse MRT.
Sujet: "Requêtes dans les systèmes pervasifs intégrant des sources de données non conventionnelles".

6.4.2 Stages de DEA et de Master 2 Recherche encadrés

Chaque année, j'ai encadré un à deux stages de DEA ou master. Ces stages ont permis soit d'étudier une idée naissante, soit d'approfondir une problématique spécifique entrant dans le cadre d'un projet en cours. La majorité des stages de master encadrés sont de spécialité informatique. Deux sont des formations différentes: l'une en bioinformatique et l'autre en analyse des systèmes

de santé. Le tableau 6.3 donne la liste des treize stages de DEA et master 2 recherche que j'ai encadrés.

Année	Etudiant	Sujet
2007	M. Zouari	Modélisation de politiques d'adaptation
2007	M. Hindawi	Adaptation pour un dossier médical européen
2006	A. Cros	Développement d'un système d'extraction de données de puces à ADN de SITRANS pour les bases internationales (master Approches Mathématiques et Informatiques du Vivant de Lyon)
2006	N. Fraga	Modélisation du schéma fonctionnel d'une application et étude de sa cohérence
2006	R. Doumat	Gestion de versions et interfaces utilisateur adaptables
2005	B. Elloumi	Plate-forme générique de génération automatique des interfaces graphiques pour des environnements multi-terminaux
2004	P. Tantravahi	Méthode de conception d'applications par services Web
2003	T. Chaari	Le projet SEFAGI: simple environment for adaptable graphical interfaces
2002	I.S. Oprescu	Etude de l'intégration d'outils de traitement de la langue pour l'extraction de données dans des documents médicaux
2001	L. Varron	Tests du prototype DRUID et mise en place d'exemples (master Méthodes d'analyse des systèmes de santé)
2001	M. Boumediene	Etude de la construction automatique de modèles de document XML à partir d'un modèle de données
2000	J. Garcia	Liaison d'une base de données relationnelle à une interface utilisateur fondée sur les documents semi-structurés
1999	A. Weill	Une nouvelle approche pour l'interrogation de documents structurés, fondé sur la corrélation base de documents / base de données

TAB. 6.3 – *Stages de master recherche et DEA encadrés*

6.4.3 Autres projets ou stages encadrés

J'ai encadré un à deux projets de fin d'études d'ingénieur chaque année (sauf pendant mes congés maternité!) pour mes projets de recherche. Les projets de fin d'études (PFE) que j'ai encadrés sont des projets des départements Informatique (IF) ou Télécommunications (TC) de l'INSA de Lyon, mais également de l'ENIS de Sfax, Tunisie. Les meilleurs ont poursuivi par un stage de master sous ma direction. Les autres stages que j'ai encadrés sont de types assez diversifiés. Ils sont répertoriés dans le tableau 6.4.

Année	Etudiant	Type du stage	Projet
2007	L. Zeng	PFE TC	SoCQ
2007	A. Sahnoun	PFE Sfax	SECAS
2006	M. Zouari	PFE Sfax	SECAS
2006	P.-Y. Branciard, S. Richem	PFE TC	SECAS
2005	D. de Leon Perez	PFE TC	SECAS
2005	M. Sinaceur	PFE TC	SITRANS
2004	B. Elloumi	PFE Sfax	SEFAGI
2003	B. Perez	PFE IF	SEFAGI
2003	C. Salce	PFE IF	DRUID
2003	V. Cadot-Libercier	stage mastère informatique INSA	SEFAGI
2002	T. Chaari	PFE Sfax	SEFAGI
2002	D. Stefan	PFE IF	SEFAGI
2002	J. Sastre	PFE IF	DRUID
2002	L. Duby	stage DUT statistiques	SITRANS
2002	C. Keime, C. Knibbe, A. Laugraud, C. Truntzer	projetCollectif INSA Bioinformatique	SITRANS
2002	B. Jeandot	DESS ingénierie documentaire	SEFAGI
2001	A. Alussi	stage INSA informatique	DRUID
2000	J.-F. Bellet	stage de fin d'études ingénieur CNAM	DRUID

TAB. 6.4 – *Autres stages encadrés*

6.5 Rayonnement

6.5.1 Editeur invité et organisation de manifestations

Mes activités en tant qu'éditeur de numéros spéciaux de revues ou de proceedings sont récentes, même si elles sont déjà au nombre de quatre. Elles concernent toutes les systèmes d'information, et trois d'entre elles sont tournées volontairement vers les applications au domaine médical.
J'ai participé à la création et à l'organisation de manifestations dans des domaines très proches de mon activité de recherche, sur les thèmes des systèmes pervasifs. J'ai tout particulièrement apprécié le travail afférent à l'organisation des trois workshops APIS, SOMITAS et HPS. La prise en main de numéros spéciaux de revues constitue en fait un travail très similaire, que j'ai également toujours plaisir à mener. Les retombées de telles activités sont de plus importantes en terme de rayonnement.

- **Editeur invité**
- Paul Harper, Frédérique Laforest (ed.), Joint advances in Information Systems and Operational Research applied to Healthcare; Special Issue of the Journal of Healthcare Information Systems and Informatics (IJHISI). IGI publishing, scheduled for automn 2008.
- Laurence T. Yang, Sajid Hussain, Frédérique Laforest, Christine Verdier (ed.), Pervasive Health Care Services and Technologies; Special Issue of the on-line International Journal of Telemedicine and Applications (IJTA).

Hindawi Journals, scheduled for spring 2008.
- F. Laforest, F. Le Mouel (ed.) N° spécial Adaptation et gestion du contexte; revue RTSI série Ingénierie des Systèmes d'Information (ISI), volume 11 n°5, 2006, Hermès Lavoisier, Paris.
- **Création et organisation de workshops**
- Créatrice et co-organisatrice du workshop Adaptation in Pervasive Information Systems APIS (conjoint à CAiSE), 2008. J'ai proposé l'idée d'un tel workshop à S. Lavirotte de Nice. Nous avons rédigé la proposition de workshop à la conférence conjointe. Je gère le site web du workshop, fais la publicité du workshop sur différentes listes de diffusion... http://liris.cnrs.fr/apis
- Co-créatrice et co-organisatrice du workshop Software Organisation and MonIToring of Ambient Systems SOMITAS (conjoint à Amby-Sys), 2008. Je gère le site web du workshop. J'ai également participé à la rédaction de la proposition de workshop à la conférence conjointe. Je fais la publicité du workshop sur différentes listes de diffusion. http://somitas.org
- Co-créatrice et co-organisatrice du workshop Health Pervasive Services HPS (conjoint à ICPS), 2006. J'ai co-géré l'ensemble de la chaîne de ce workshop, faisant office à la fois de program chair et d'organisatrice. J'ai par exemple négocié avec les IEEE l'édition des actes, défini les droits d'inscription en collaboration avec ICPS, constitué le comité de programme, rédigé la proposition de workshop, etc. http://liris.cnrs.fr/hps06
- **Comités d'organisation**
- Membre du comité d'organisation de la conférence internationale Int. Conf. Pervasive Services ICPS, Lyon, juin 2006. J'ai été chargée de l'organisation de la soirée de gala.
- Membre du comité d'organisation de la conférence nationale Inforsid 2000. J'ai participé à l'organisation de la soirée de gala, à la recherche de sponsors et à la constitution du cartable de "goodies" des participants.

6.5.2 Membre de comités de lecture et de programmes

Je suis membre du comité de lecture d'un journal national (ISI). J'ai été membre de 16 comités de programme de conférences et de 7 workshops ou ateliers, tant dans des événements purement informatiques proches de mon domaine de recherche (IE, ICPS, ER, Inforsid...) que dans des manifestations d'informatique dédiées au domaine médical (MIE, MedInfo...). J'ai également servi de relectrice pour de nombreuses conférences que je ne citerai pas ici, et aussi pour le journal Int. Journal on Information Fusion, Elsevier.

- **Comité de lecture d'une revue**
- Revue Ingénierie des Systèmes d'Information, Hermès, Paris
- **Comités de programmes de conférences**
- Int. conf. on Research Challenges in information science, RCIS 2008
- Int. Conf. on Intelligent Environments, IE 2007 et 2008
- Int. Conf. Pervasive Services ICPS 2006
- Medical Informatics Europe MIE 1999, 2000, 2002, 2003, 2005, 2006
- Medical Informatics MedInfo 2001, 2004, 2007

- Int. Conf. on Conceptual Modelling ER 1999
- Conférence nationale Inforsid 1999, 2001, 2007
- Int. Symp. on Methodologies for Intelligent Systems 2002
- **Comités de programmes de workshops**
- Workshop on methods and design models for health information systems(MeDEHIS@ICDIM07), oct 2007, Lyon
- Workshop on Ontologies-based Techniques for DataBases and Information Systems and Semantic Web and Databases ODBIS-SWDB@VLDB07), Vienna, Austria, September 2007
- Workshop on Knowledge Management and Discovery for Ubiquitous and Pervasive Applications (KUPA@DEXA07), Regensburg, Germany, September 2007
- Workshop on Smart Homes for Tele-Health (SmarTEL@AINA07), Niagara Falls, Canada, mai 2007
- Atelier Objets, Composants et Modèles dans l'Ingénierie des Systèmes d'Information (OCM-SI@Inforsid07), Perros Guirec, mai 2007
- Atelier Prise en Compte de l'Utilisateur dans les Systèmes d'Information (PeCUSI@Inforsid07), Perros Guirec, mai 2007
- Atelier OSGI@Ubimob06, Paris, octobre 2006

6.5.3 Groupes de Recherche et Groupes de Travail

J'ai toujours participé aux GDR correspondant à mes problématiques de recherche, d'abord dans le cadre du GT documents multimédia, puis dans des groupes traitant de mobilité, d'adaptation ou de gestion de données et services. Je suis également membre du GDR STIC-santé qui regroupe des chercheurs appliquant leurs travaux à la santé.

- GDR I3 thème 4 : Masses de données et accès à l'information. Présentation de nos travaux sur la gestion de données non conventionnelles le 15 juin 2007 (par Y. Gripay)
- GDR ASR/GSP action Adaptation Dynamique. Présentation de nos travaux sur l'adaptation de services Web le 3 octobre 2006 (par T. Chaari)
- PRC-GDR I3, GT Mobilité et Ubiquité
- PRC-GDR I3, GT Documents Multimédia. Présentation de nos travaux sur les documents semi-structurés dans DRUID le 8 juin 2001.
- GDR STIC-santé, GT Systèmes d'Information Médicaux

6.6 Contrats

L'ensemble des contrats auxquels j'ai participé ont une application directe au domaine (bio)médical, impliquant bien sûr des chercheurs en informatique, mais aussi des chercheurs dans d'autres disciplines et des personnes du domaine d'application (biologistes, hôpitaux, médecins, associations d'hospitalisation à domicile, éditeurs de logiciels médicaux...). Le travail effectué dans le cadre de ces projets m'a permis d'acquérir une expérience réelle des besoins et spécificités de ce domaine. Ce domaine nous beaucoup apporté, tant en source d'inspiration

qu'en cas pratiques pour la validation de nos propositions. Bien entendu, nos propositions, même si elles ont été inspirées par les besoins de ce domaine, ont une visée plus large et restent indépendantes de tout domaine d'application. Le tableau 6.5 liste l'ensemble des contrats auxquels j'ai participé et donne le type de responsabilités que j'ai eu dans ce cadre. La suite de cette section présente notre apport dans chaque projet.

Années	**Acronyme**	**Rôle**	**Type de financement**
06-09	HRP3	**responsable** d'un workpackage et d'un groupe de travail	cluster Région Rhône-Alpes
02-05	DocPatient	invitée	HTSC Région Haute Picardie
01-04	COQUAS	participante	RNTS (technologies pour la santé)
02-03	SITRANS	**porteur** du projet	Inter-EPST bioinformatique et CDD INRIA
00-03	SICOM	**responsable** d'un workpackage et co-directrice de thèse	Thématique prioritaire
99-01	TIPHAD	participante	ACI

TAB. 6.5 – *Contrats*

2006-2009 HRP3 Cluster GOSPI Région Rhône-Alpes
Titre : Hôpitaux en réseau : Prévoir, Partager, Piloter
Responsabilité : **Responsable** d'un workpackage (système d'information) et d'un groupe de travail (système d'information et de décision)
Partenaires académiques: PRISMA (Lyon), LASPI (Saint Etienne), GRAPHOS (Lyon), CIS (Saint Etienne), LSR-IMAG (Grenoble), MA2D (Lyon), COACTIS (Saint Etienne)
Autres partenaires:ARH Rhône-Alpes, CH Saint Joseph Saint Luc, CHU Grenoble, CHU Saint Etienne, CH Valence
Descriptif : Il a pour but l'étude et le prototypage d'un système de pilotage de la filière d'urgences. Notre objectif est de définir le système d'information support à ce pilotage. Ce système d'information est par essence distribué et doit intégrer des outils mobiles. C'est donc un bon exemple de système pervasif.

2002-2005 DocPatient HTSC Haute Picardie
Titre : Le dossier patient électronique documentaire
Responsabilité : Participante **invitée** au projet depuis fin 2003
Partenaires académiques: Laria (Amiens), CRIISEA (Amiens), ECCHAT (Amiens)
Autres partenaires : CHU Amiens, société UniMédecine, AP-HP, GIP Télémédecine de Picardie
Descriptif : Il a pour but de spécifier les caractéristiques d'un système d'information pour la gestion de dossiers patients sous forme documentaire. Les participants à ce projet ont étudié notre approche DRUID et nous ont alors demandé de participer à leur projet. Nous avons donc été intégrés lors de la seconde année pour effectuer une évaluation en grandeur réelle de DRUID et de

son intégration dans les spécifications de DocPatient. Nous avons profité de ce projet pour mettre à jour notre prototype, et effectuer de nouveaux tests avec des documents fournis par les partenaires. Mon second congé maternité m'a cependant empêchée de participer à la dernière année de ce projet.

2001-2004 COQUAS Réseau National Technologies pour la Santé
Titre : COordination pour la QUAlité des Soins
Responsabilité : Participante au projet
Partenaires académiques : CERIM (Lille), LAMIH (Valenciennes)
Autres partenaires : CHRU de Lille, CHU de Rouen, CH de Valenciennes, Roubaix, Armentières, Montreuil, AP-HP, CRAM Picardie, sociétés Planet-HealthCare, CegetelRSS
Descriptif : Ce projet fait suite au projet TIPHAD. Il nous a permis de poursuivre notre travail sur l'architecture DRUID d'extraction de données dans des documents faiblement structurés. L'association de suivi de patients à domicile partenaire du projet nous a également fourni des documents supports pour les tests de DRUID.

2002-2003 SITRANS Inter-EPST bioinformatique et CDD INRIA
Titre : Système d'Information pour le TRANScriptome
Responsabilité : **Porteur** du projet
Partenaires académiques : BF2I (Lyon), plate-forme transcriptome au DTAMB (Lyon)
Descriptif : Il a pour but l'étude et le développement d'un système d'information pour la gestion et la comparaison des expériences à base de puces à ADN de la plate-forme transcriptome. Outre ce travail d'ingénierie assuré par un CDD INRIA d'un an, ce projet a ouvert des perspectives de recherche dans la thématique de la recherche d'informations. Les outils de navigation dans des cartes de concepts que nous avons proposés séduisent les biologistes qui doivent analyser ces données.

2000-2003 SICOM Thématique prioritaire Région Rhône-Alpes
Titre : Conception de Systèmes d'Information Communicants pour la Santé
Responsabilité : **Responsable** du workpackage "système d'information" et co-directrice de la thèse financée par la Région dans ce cadre (M. Boumediene)
Partenaires académiques : CITI (Lyon), LASS (Lyon), LBTI (St Etienne), Cardiologie et soins intensifs (Lyon), lab. de physiologie (Lyon)
Autres partenaires : Réseaux de soins ONCORA et AURAL, sociétés Al'X, Alliance Santé, Baxter, MAPI, DEF Systèmes, CNET Grenoble, France Télécom Lyon
Descriptif : J'ai été coordinatrice du lot " système d'information ". Nous avions pour objectif de définir une architecture de système d'information de télémédecine indépendant de la pathologie traitée. Ce projet nous a permis de définir dans un premier temps la problématique des interfaces multi-plates-formes, objet du projet de recherche SEFAGI, puis de l'étendre à l'adaptation des services, qui est le sujet principal du projet SECAS. La thèse de Mohamed Boumédiene

a été financée par une bourse de recherche région obtenue dans le cadre de ce projet.

1999-2001 TIPHAD Action Concertée Incitative
Titre : Technologies d'Intercommunication Pour l'Hospitalisation A Domicile
Responsabilité : Participante au projet
Partenaires académiques : CERIM (Lille), PERCOTEC (Lille), LASS (Lyon), AP-HP
Autres partenaires : société Planet-HealthCare
Descriptif : Ce projet avait pour objectif de définir un système d'information permettant le support à l'hospitalisation à domicile. En particulier, il nous a permis de travailler sur notre architecture DRUID de saisie de données dans des documents faiblement structurés pour alimenter une base de données.

6.7 Tâches administratives

Mes tâches administratives concernent à la fois la recherche et l'enseignement. En recherche, je participe à la vie du laboratoire en tant que membre élue au conseil de laboratoire, et je suis membre de la commission de spécialistes de 27ème section. J'ai également été webmestre du site de l'ancien DEA DISIC.
En enseignement, j'ai été recrutée lors de la création du Département Télécommunications. Nous nous sommes donc partagé toutes les tâches afférant à la création d'une telle structure : définition du programme d'enseignement, définition du fonctionnement du département, élaboration des profils des demandes de postes... Cela a été un travail très important durant quatre années (1998 à 2002). Depuis 2002, je suis coordinatrice des enseignements du domaine informatique du département et co-responsable de la gestion de l'emploi du temps des enseignements (j'ai également co-développé le logiciel ad hoc correspondant).

- Membre élue de la Commission de Spécialistes d'Etablissement de 27ème section en 2001-2004 (suppléante) et 2005-2008 (titulaire) de l'INSA Lyon
- Membre élue du conseil de laboratoire du LIRIS depuis février 2007
- Membre élue du conseil de laboratoire du LISI 1994-2002
- Webmaster du site web du DEA DISIC (1998-2003)

- Membre du Comité de Pilotage du Dpt Télécommunications pour sa création et sa mise en place (ouverture en septembre 1998 lors de mon recrutement) jusqu'à sa stabilisation en 2002
- Membre du conseil de Département du Dpt Télécommunications de l'INSA de Lyon 1998-2001
- Coordinatrice des enseignements du domaine Informatique du Département Télécommunications de l'INSA Lyon depuis sa création (septembre 1998)
- Responsable ou co-responsable des emplois du temps d'enseignement du Département Télécommunications de l'INSA Lyon depuis 1999

– Co-concepteur et co-développeur du logiciel de gestion de l'emploi du temps du Dpt Télécommunications de l'INSA Lyon

6.8 Enseignement

J'ai effectué quelques vacations durant ma thèse de doctorat, en bases de données ou en systèmes d'information. Ensuite, durant mon poste d'ATER en IUT informatique, j'ai enseigné principalement les systèmes d'exploitation, le génie logiciel, le multimédia, les interfaces homme-machine.
Ensuite, dans le cadre de mon poste de maître de conférences, j'ai enseigné et enseigne les systèmes d'exploitation, l'algorithmique, le langage C, le langage java, les bases de données, le génie logiciel... sous forme de cours magistraux, travaux dirigés et travaux pratiques. Je conduis également des enseignements plus proches de ma recherche dans le cadre du module systèmes d'information pervasifs au Département Télécommunications (cours magistral et travaux pratiques) et en master 2 recherche (cours magistral). J'ai aussi donné un cours proche de nos travaux sur les documents en DEA en 2000 et 2001.

Pour tous ces enseignements, j'ai monté entièrement l'ensemble des supports de cours, TD et TP, et bien entendu les sujets d'examen. La mise en place d'un nouveau Départment (Télécommunications de l'INSA) a nécessité de définir les matières, leurs objectifs, les volumes, les contenus, leur relations avec les matières des autres domaines (comme par exemple traitement du signal ou réseaux pair à pair)...

Premier cycle IUT Informatique
Modélisation objet, génie logiciel, interfaces homme-machine, documents et hypertextes (World Wide Web), composants distribués, indexation sémantique d'images.

Second cycle Ingénieurs Télécommunications INSA de Lyon
Architecture de l'ordinateur et systèmes d'exploitation, Algorithmique et programmation, Systèmes d'information et génie logiciel, Bases de données, Java, Systèmes distribués, Web dynamique, projets tutorés, projets de fin d'études, systèmes d'information pervasifs.

Troisième cycle DEA et master M2 recherche
2007-2008, Master Informatique Lyon M2Recherche. Unité d'enseignement mutuelle : Systèmes d'information pervasifs et interopérables. F. Laforest, D. Benslimane.
2006-2007, Master Informatique Lyon M2Recherche. Unité d'enseignement mutuelle : Systèmes d'information pervasifs, collaboratifs et interopérables. P. Ghodous, F. Laforest.
2004-2005 et 2005-2006, Master Informatique Lyon M2Recherche. Unité d'enseignement mutuelle : Systèmes d'information pervasifs - architecture et cohérence des données. Y. Amghar, F. Laforest.
1999-2002, DEA Documents multimédia, Images et Systèmes d'Information

Communicants, cours de recherche : Indexation et interrogation de sources d'informations hétérogènes. S. Calabretto, F. Laforest.

Chapitre 7

Sélection d'articles

J'ai choisi de sélectionner un article par projet. Ainsi, on pourra lire les articles suivants :

Projet QDE [28]
Frédérique Laforest and Anne Tchounikine. Indexing semi-structured documents for context-based information retrieval in a medical information system. In *DEXA Workshop : First International Workshop on Document Analysis and Understanding for Document Databases (DAUDD 1999), Florence, Italy*, pages 593–597, 1999.

Le projet Query Documents by Example a été mené en 1998-99. Cet article explique la modélisation des documents et des requêtes, et donne l'algorithme permettant l'appariement des requêtes et documents.

Projet DRUID [16]
Frédérique Laforest and Youakim Badr. Modèle de couplage de documents structurés et de bases de données. le projet druid. *Ingénierie des Systèmes d'Information, n° spécial Bases de données semi-structurées*, 8(5-6):109–126, 2003.

Cet article est le plus récent que nous avons publié sur l'extraction de données dans des documents faiblement structurés. Il est également le plus complet. Il a été publié dans un numéro spécial de la revue RTSI-ISI dédiée aux documents semi-structurés.

Projet SEFAGI [15]
Tarak Chaari and Frédérique Laforest. Génération et adaptation automatiques des interfaces utilisateurs pour des environnements multiterminaux. le projet SEFAGI. *Ingénierie des Systèmes d'Information, n° spécial Systèmes d'information pervasifs*, 9(2):11–38, 2004.

Cet article n'est pas le plus récent sur le projet SEFAGI, mais il présente de façon détaillée nos travaux sur l'adaptation d'interfaces graphiques à des environnements multi-terminaux. Il a été publié dans un numéro spécial de la revue RTSI-ISI sur les systèmes d'information pervasifs.

Projet SECAS [3]
Tarak Chaari, Dejene Ejigu, Frédérique Laforest, and Vasile-Marian Scuturici. A comprehensive approach to model and use context for adapting applications in pervasive environments. *Int. Journal on Systems and Software*, 3, 2007.

L'article sélectionné ici donne une bonne vision de l'adaptation au contexte d'applications legacy que nous avons définie dans le projet SECAS. Cet article est également le fruit d'une collaboration avec une autre thèse de l'équipe (thèse de Dejene Ejigu) sur la modélisation du contexte d'utilisation.

Projet SoCQ [18]
Yann Gripay, Frédérique Laforest, and Jean-Marc Petit. Towards Action-Oriented Continuous Queries in Pervasive Systems. In *Bases de données Avancées 2007 (BDA'07)*, October 2007.

Ce dernier article présente les premiers travaux effectués pour la construction déclarative d'applications pervasives. Il explique comment nous avons défini une représentation homogène de sources de données non conventionnelles et donne des pistes pour la rédaction et l'évaluation de requêtes.

[28]
Frédérique Laforest and Anne Tchounikine. Indexing semi-structured documents for context-based information retrieval in a medical information system. In *DEXA Workshop : First International Workshop on Document Analysis and Understanding for Document Databases (DAUDD 1999), Florence, Italy*, pages 593–597, 1999.

Indexing Semi-Structured Documents for Context-based Information Retrieval in a Medical Information System

Frédérique Laforest Anne Tchounikine
Laboratoire d'Ingénierie des Systèmes d'Information
INSA, Bat 401, 20 Avenue A. Einstein, 69621 Villeurbanne Cedex, France
E-mail: frede@lisiflory.insa-lyon.fr, atchouni@if.insa-lyon.fr

Abstract

Most of Information Retrieval models for documents are intended for SGML-like structured documents. In the context of medical informatics, the Patient Record document needs a looser structuration process as an a priori structure can hardly be defined. Thus we first propose an authoring tool that allows to annotate embedded information, i.e. to give them a context, with qualifiers that are stored in a thesaurus rather than in SGML-like DTD. The retrieval process in the Patient Records collection takes into account the flexibility of the qualifying process while reformulating the queries.

1. Introduction

Management of documents, including storage and efficient Information Retrieval (IR) aspects, is now recognized as an important field of research. Earlier IR models focus on the content of documents and provided systems based on keywords. Sets of keywords are manually composed or obtained by frequency calculus on full-text analysis. The growing use of authoring standards like SGML[2] or XML [10] which allow to hierarchically structure documents, has relaunched the interest in Information Retrieval research, making it desirable the use of structural tags to enhance document querying. Some experiments [9, 4] recommend to match structured document into DB so as to be able to benefit from powerful ad-hoc query languages provided by the DBMS. On the other side, [5, 3] recommend the definition of a new paradigm for manipulating and querying the new data represented through documents. In the context of medical informatics, the study of the "Patient Record" (PR) electronic document leads us to look for the loosest structuration and retrieval model. Indeed, as addressed in the first part of this article, it is neither possible, nor desirable for users to impose a structuring model for PRs. Section 2 presents the context of our research i.e. the PR specificities. In section 3, we propose a document production model in which the user does not structure his document, but rather *qualifies* the contained information. Section 4 focuses on our IR model that takes into account the flexibility of the indexing and allows queries on the PRs composition.

2. Background

2.1. The Patient Record

Definition. The Patient Record (PR), also known as "Medical Record", has been defined as follows:*"The medical record is the memory in which all the data necessary for the surveillance and to take in charge the patient are stored"* [6]. The PR is the main tool for the centralization and the coordination of the medical activity in hospital. The PR is multi-author as it is filled by various persons from the medical staff (physicians, nurses, medical secretaries) and accessed by different persons with respect of confidentiality restrictions due to the sensitive character of information. The PR is unique and non-modifiable as it recounts the medical story of the patient and diseases, accidents, even bad diagnoses or errors cannot be removed. For all these reasons, the Patient Record is one of the main repositories for the knowledge needed in medical activities.

A document view of the PR. Early electronic PR were reduced to the "data" part of the Medical Information System (MIS) using the various classifications provided by the medical discipline (classification of diseases, medicine data-banks...). Still many studies [12, 8] argue that a document view of the medical record avoids a constraining and unreacheable standardisation of information and thus could provide a more viable solution.

We can see the PR as a "hyperdocument" composed of 3 types of multimedia documents. First type are highly structured documents which contain only formatted information and can be stored in a database (e.g. some analyzis results

forms). Second type are structured documents whose information is not formatted [8]. They can be stored using a predefined Document Type Definition (DTD) and belong to the classical SGML-based documents. For example, "end of hospitalization" reports are normalized for administration needs and do have a pre-defined structure. In the last type, rely semi-structured, or un-structured documents for which DTDs can hardly be defined. For example, clinicians notes on the patients have a structure which is very specific to each note and cannot be foreseen as a whole. The few experiments that have been made for the structuration of this last type of documents [1, 7] result in a stupendous number of DTD so as to cover the domain.

2.2. Outlines for Information Retrieval

We here outline how the very specific semantics of the medical information contained in the document can be used, or influence, the building of a query.

The chronological aspect. Information contained in the PR is strongly related to temporal aspects. As addressed before, information is always added in the document, but never modified nor deleted. Indeed, for legal reasons, even erroneous information (diagnosis errors, cancelled prescriptions...) must remain in it. The order in which information pieces succeed one another in the document reflects a chronology of facts and thus the medical history of the patient. Moreover, the composition of the PR can be seen as the image of the way the medical staff deals with the patient case, and then becomes a real trace of a reasoning process. For example, the request searching for cases of prescription of a certain drug followed by a particular analysis is in the same time a query on the chronology of acts and on the follow-up of a medical case. But this query is also equivalent to a query on the way the PRs are composed. This frequent case-based searching corresponds to studies that focus the medical art for epidemiological or learning purposes. We think that the PR seen as a ordered list of events can be a good basis for these studies.

The contextual aspect. Another aspect of the medical information is its "context-sensitivity". The mentioning of a drug in a paragraph relating a prescription does not have the same meaning in a prescription paragraph and in an allergy paragraph. Thus, the meaning, or the semantics, of an information always depends on its writing context. As a consequence, asking for a specific information cannot be understood without making its context clear. This aspect is obviously not specific to medical documents. But in this special field, an Information Retrieval system that would be based on keywords would not only be completely useless, but nearly absurd.

To conclude this part, we can say that the retrieval in our documentary base focuses on the structure of the PRs and uses the structure to be able to understand this query. Despite, the PR remains a semi-structured document. For all these reasons, we need at the same time (i)a non-constraining production process, and (ii)a retrieval process that relies on the content and the organization of PRs and is robust to the flexibility of the production process.

3. Population of the documents collection

The PR, written by several authors, represents the entire medical history of a person, and for this reason it can only exist in a single version, although mentioning different dates. The element that federates the information pieces in the PR is the subject of this document, i.e. the patient himself. Each PR in our documents collection thus will be associated to the meta-data "Patient" as it is registered in the Administrative Information System of the hospital. If the PR has no a priori structure, and the information no a priori types, the information still has a well-determined semantics according to the moment or the place where they appear. Moreover, PRs represent a whole which cannot be divided into sub-records according to a criterion like the type of information or its meaning.

3.1. The content

Data. Data may be selected in any database of the information system : patient or staff databases, medical terms classification, drugs data-banks... Depending on their source they may have a strong type, a definition domain, associated methods. At the production time, the author selects a data in the appropriate database. This data is automatically surrounded by its identification. This identification can be seen as a reference and is used in this way at the consultation time. `<DATA DBID=BCB OID=124 > aspirin </DATA>` refers to the object "aspirin" identified number 124 from the BCB drugs bank.

Free information. Information which is not a data, is "free information". It is captured freely, as in a classic document authoring tool. Information freely captured can be of any type but at present we have restricted our study to textual information.

3.2. The context

Logical Units. In the medical area, it is classic to distinguish 4 medical types of information ([11]) : S (Subjective) refers to the information the patient provides; O (Objective) refers to information the physician discovers (clinical examination or other analysis results); A (Assessment) represents the diagnosis of the doctor and more widely his thinking about the case of the patient; P (Plan) represents all the prescriptions given to the patient. They form the

4 Logical Units (LU) of PRs documents. The insertion of any information in a document must be made in the context of one of these logical units. Thus, a LU is created by a person of the medical staff each time he wants to register something about the patient case. He chooses the type of LU he wants to add according to the semantics of its content. The meta-data associated to a LU is its author and its date of creation and are automatically inserted. For example, the author could write the following sentences:

```
<O Author="Dr No" Date="12/12/92"> The patient presents an undeniable tobacco addiction</O>
<A Author="Dr No" Date="12/12/92"> We may suspect a lung cancer</A>.
```

Qualified text. We call "qualified text" a text which can be given sort of a "title", and this without pointing out a logical structure for this text. The text qualifier inserted arround the information does not give a structure to the text but is used to give a sense, i.e. to qualify, the information that follows. An author could write the following paragraph:

```
<O author="Dr No" Date="12/12/92"> the patient presents an undeniable <habit> tobacco addiction</habit></O>
<A author="Dr No" Date="12/12/92"> We may <Hypothesis> suspect a lung cancer</Hypothesis> </A>
```

The two pieces of information "tobacco addiction" and "lung cancer" are respectively said to be a habit and a hypothesis. The author is free to put his first qualifier (`<habit>`) in any place of the sentence: after the word "undeniable" or before the word "presents": if he wants his text to be correctly qualified, the only requirement is that the words "tobacco" and "addiction" rely in-between the qualifiers. Thus, as a difference with classic SGML tags, qualifiers can be inserted anywhere in the text. The aim is more to provide a way to annotate information in a document instance than to build a model of document. There is no constraint on the type of the information that follows (it could be free information or data of any type), nor on the way qualifiers can eventually be nested one into the other (e.g. `<f><g> information </g></f>`). In this last case, we consider that the information is multi-qualified i.e. is qualified by all the qualifiers that surround it. As mentioned above, qualifiers do not provide a structure for the PRs. This means that there is no SGML-like DTD associated to the PRs. Despite, qualifiers have strong semantics. We do not need a simple list of qualifiers to be provided to the authors, but rather a real dictionary where the semantic relationships between qualifiers would be explicit. Thus, we propose to build a thesaurus of pre-defined qualifiers, linked with the traditional synonymy and generalization/specialization relationships. In the production process, the thesaurus will be used to help the author in choosing his qualifiers ; then the thesaurus will have a major role in the retrieval process. As a summarized discussion, we can say that the main differences between SGML tags and qualifiers are:

- SGML-like tags are defined so that their places in the document are regulated. They are provided for structuring purposes. On the contrary, qualifiers are defined for annotation purposes. They do not aim at providing a structure of documents but are intended to explicit the sense of information pieces.
- SGML-like tags are defined in a DTD where composition in the only relationship provided. Qualifiers are stored in a thesaurus which allows for the use of synonymy and generalization/specialization relationships between qualifiers. Composition is not on purpose as any composition is allowed.
- SGML-like tags are defined a-priori and cannot be adapted to each author. Qualifiers may be created and inserted in the thesaurus, so that each author can have an adapted tool.

3.3. Documents Descriptors

We shall try now to build up what is called a document descriptor in documentary systems. The descriptor aims to be a summarized description of the documents i.e. what will be indexed to ease the retrieval process.

The tree representation. The PR can be represented as a tree in which the nodes are the different components of the document and edges are composition or reference links. This representation is drawn on classic hypertext representations. This tree can have any depth, but we can distinguish 4 levels that are the following:

- Level 1: the root, i.e. the PR itself
- Level 2: Logical Units: there can be only one node of this type per branch (i.e. LU cannot nest)
- Level 3: Qualifiers: they can nest, so that the depth of this level is any. Qualifiers always contain information: they cannot be placed at a leaf of a tree. In those levels, the nodes are labeled using their identifier in the thesaurus and not their formal names (e.g. the `<allergy>` qualifier will be referenced in the tree as "f").
- Level 4: Information: this level merges information that originates from the content into 2 parts. One is a list that contains the key-words (items) provided by a full-text indexing mechanism computed on free information. The second part consists in all the DB used to reference data.

In these trees, links directed to a data are reference links, while all other links are composition links.

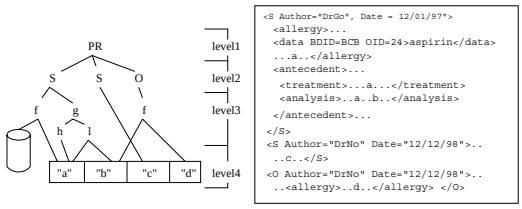


Figure 1. Tree representation of a document

4. The Information Retrieval process

Our approach for information retrieval can be summarized as follows: both documents and queries are represented using a tree-based representation ; the correspondence function between the query tree and documents in the collection is based on pattern matching on flattened trees. In the following we first describe how queries are formulated and then we describe the pattern generation for documents and queries and lastly the matching process.

4.1. Queries by example

As queries are case-based, they have the form of documents and it is straightforward to propose a query-by-example language. This means that the query is formulated as a document using LU, qualifiers, data and free information. If the user is searching for "cases where a patient announced an allergy to A treated with B and where more investigations showed an allergy to C", then the query stated is :

```
<Q> <S> <allergy>A</allergy> <treatment>
B</treatment> </S>
<O> <allergy>C</allergy></O> </Q>
```

4.2. Trees implementation

We use a flattened version of trees to implement the index on documents and to make pattern matching between a query and the documents.

Flattening principles. The aim is to produce a string of tokens corresponding to a flat representation of each tree. The flattening process simply consists in a deep-first parsing of a tree. It follows 3 rules :

- encountering a start tag for a LU or a qualifier node results in the creation of a starting token as `'s'` *node identifier* (e.g. sf) ;
- encountering an item or data results in the creation of a token for this leaf, which is *leaf identifier*;
- encountering an end tag for a LU or a qualifier node results in the creation of a "ending" token as `'e'` *node identifier*.

Thus, the token string produced is a synthesized version of the document where identifiers of LU, qualifiers, items and data only remain (empty words do not appear).

The documents index. Each document descriptor is flattened as described above. Our system also contains an index of the items and data used; each index entry addresses a list of pointers to the flattened descriptors in which the entry appears. In this way, each document descriptor is pointed to by as many index entries as the number of items and data it contains. Similarly, each descriptor addresses its corresponding PR in the documents collection. This index speeds

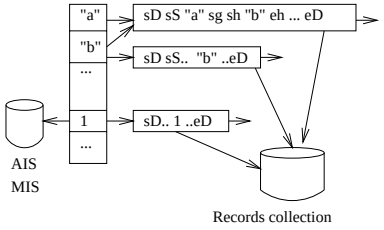


Figure 2. The documents collection indexing

up the query answering as it restricts the set of documents to submit to the pattern matching process to the set of documents containing the items and data contained in the query.

The query pattern. The answers to a query must contain the same LU within the items, identically qualified, and in the same order. Nevertheless, relevant documents may also contain other components, and this in-between the terms of the query. In figure 3, the document drawn in (c) is an answer to the (a) query. In other terms, the tree of the query

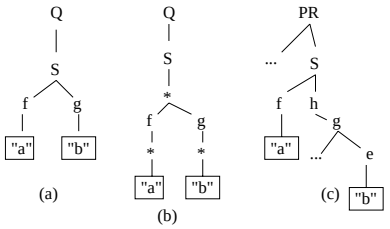


Figure 3. Query pattern

must be a sub-tree of the document tree. Thus, we have to add "virtual" nodes in the query tree to represent the fact that the qualifiers and items we are looking for can stand in

variable depth and/or width in the matching trees. This step is shown on Figure 3 (b). The pattern to be found in relevant documents has to be modified to ensure it takes care of the depth and width variability of the answers. It is re-written into the following pattern: sD * sS * sf * "a" * ef * sg * "b" * eg * eS * eD where "*" is any series of tokens, i.e. any sub-tree of the document. Let us notice that the existence and depth or width of these sub-trees do not allow us to conclude on the relevance degree of the documents. As a matter of fact, the depth and width of the trees depends on the precision how the documents and queries have been produced, characteristic which is totally dependent on the author's way of working.

4.3. Evaluation of queries

We have to match a query pattern containing virtual tokens to a descriptors base indexed by a list of items and/or data.

Pattern matching. The first step is to search the entries in the index that correspond to the items or data tokens of the query. It restricts the number of pattern matching processes to attempt. Then we have to compare each descriptor pointed out by the entry with the pattern of the query.

Query reformulation. Previous step computes the most relevant documents i.e. containing exactly the same tokens, in the same order, but with various depth or width. In the second step, we have to find "similar" documents i.e. documents that were not qualified like the query asked for, but with qualifiers that are semantically close. In that purpose, we have to reformulate thanks to the thesaurus. The use of specialized qualifiers does not modify the accuracy of the answer, whereas the use of generalized terms reduces it. At this stage of our study, we did not focus on the calculation of accuracy. It is obvious that this problem will have to be addressed deeper because synonymy, generalization and specialization are a direct consequence of the way documents are produced and qualified.

5. Conclusion

In this article, we have proposed a new way for the authoring and selection of documents. The logical structure of documents is reduced to the use of 4 different logical unit types, while information pieces can be qualified by authors. Qualification is a very flexible process based on the selection of qualifiers in a thesaurus which also contains synonymy and generalization/specialization relationships between qualifiers. Querying documents consists in query-by-example. The query focuses on the chronology and the content of the events related by the PR, thus consisting in a tree pattern matching. For the pattern matching process, documents and queries are represented with documents descriptors which can be compared. A documents index allows for speeding up the matching process as it reduces the documents descriptors collection to the subset of documents containing the items and data present in the query. The next step of this study is to provide an accuracy measure of answered documents. We first have to define an order in the query reformulation tools (see paragraph 4.3.2.) and secondly we would like to study in which measure providing documents that answer only one part of the query can be relevant. The definition of the end-user tool and interface of such a system can also be a good research deal.

References

[1] L. Alschuler and al. The Kona proposal. http://www.mcis.duke.edu/standards/HL7/.

[2] M. Bryan. *SGML, an Author's Guide to the Standard Generalized Markup Language.* Addison-Wesley, 1988.

[3] P. Buneman, S. Davidson, and G. Hillebrand. A query language and optimization techniques for unstructured data. In *SIGMOD proceedings*, pages 505–516. ACM, june 1996.

[4] V. Christophides, S.Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *ACM SIGMOD Conference*, Minneapolis, may 1994.

[5] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language for xml. In *International WWW Conference*, 1999.

[6] C. M. Donald and W.M.Tiernet. Computer-stored medical records: future role in medical practice. *Journal of American Medical Association*, 1988.

[7] H. S. S. I. Group. SGML/XML as an interchange format for HL7 V2.3 messages. http://www.mcis.duke.edu/standards/HL7/.

[8] F. Laforest, S. Frenot, and A. Flory. A hypermedia-based medical records management system. In *MedInfo*, Seoul, Korea, August 1998.

[9] M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, and A. Guttman. Document processing in a relationnal database system. *ACM TOIS*, 1(25), April 1983.

[10] W3C. *eXtensible Markup Language (XML), XML-Link, XS (XML-DSSSL-o).* 1997.

[11] L. Weed. Medical records, medical education and patient care. the problem oriented record as a basic tool. In *The Press of CaseWestern Reserve University, Cleveland, Ohio*, 1969.

[12] P. Zweigenbaum and al. From text to knowledge: a unifying document-centered view of analyzed medical language. In *Methods of Information in Medicine*, 1998.

Projet DRUID

[16]
Frédérique Laforest and Youakim Badr. Modèle de couplage de documents structurés et de bases de données. le projet druid. *Ingénierie des Systèmes d'Information, n° spécial Bases de données semi-structurées*, 8(5-6):109–126, 2003.

Modèle de couplage de documents structurés et de bases de données

Le projet DRUID

Frédérique Laforest, Youakim Badr

LIRIS,
INSA, Bat Blaise Pascal
7 avenue Recteur Capelle
69621 Villeurbanne Cedex
frederique.laforest@insa-lyon.fr, youakim.badr@insa-lyon.fr

RÉSUMÉ. *Dans cet article, nous présentons DRUID, un système de saisie d'information sous forme de documents semi-structurés, lié à une base de données relationnelle. Les documents saisis par l'utilisateur sont des documents centrés paragraphes, c'est-à-dire que les balises encadrent des portions de texte libre en langue naturelle, et non des données unitaires. Le DRUID Core Module utilise un ensemble de règles permettant d'extraire de ces paragraphes les données unitaires prévues dans la base de données. Les règles sont composées d'une extension de la syntaxe XSL pour la manipulation de la structure du document, ainsi que de transducteurs à états finis pour l'appel à un processeur de langue naturelle. Les données ainsi extraites sont tout d'abord déposées dans un document centré données puis dans la base de données. L'intérêt d'un tel système est d'associer à la fois la souplesse des interfaces documentaires, et la puissance des langages d'interrogation des bases de données relationnelles. Chaque requête à la base permet d'avoir une vue tabulaire des informations demandées, ainsi qu'un accès aux documents sources de l'information.*

ABSTRACT. *In this paper, we present the DRUID system that links document-based information capture to relational database querying. End-users capture paragraph-centric documents where tags embrace paragraphs of free text. The DRUID Core Module employs a set of rules to extract relevant data from free text paragraphs. Some rules are written as XSL extension elements for document structure management, and others invoke finite state transducers for natural language processing and information extraction. Extracted data are stored in both a data-centric document and in the database. This system is interesting because it associates documents flexibility with databases querying efficiency. Queries to the database return the classical data tables as well as links to the source documents.*

MOTS-CLES: *documents structurés, extraction de données, interface utilisateur, base de données, XML*

KEYWORDS: *structured documents, data extraction, user interface, database, XML*

1. Introduction

Les bases de données et les bases de documents ont été créées pour des applications totalement différentes. Dans le premier cas, il s'agit de fournir des supports de stockage pour traiter et interroger des données dont la création et la mise à jour sont fréquentes. Dans le second cas il s'agit de retrouver des documents à l'aide de mots clés, ces documents contenant des informations stables rarement modifiées. L'arrivée d'Internet et du Web a rapproché ces deux mondes en permettant d'une part de remplir les documents à l'aide de données d'une base et d'autre part d'indexer les documents. Ces deux approches sont opposées en ce sens que l'une (données) relève des informations structurées alors que l'autre (documents) fait partie des approches qui traitent aujourd'hui des données semi-structurées.

Pour la gestion du dossier médical, les systèmes classiquement proposés sont fondés sur des bases de données. Pour le médecin, cette approche se traduit par des saisies dans des grilles. La variété des cas médicaux s'accommode mal de systèmes figés et rigides. Cependant, l'utilisation de bases de données pour l'analyse et le calcul est aujourd'hui la seule approche permettant d'accomplir des recherches et traitements efficaces. D'un autre côté, le document électronique est proche du dossier médical papier. Malgré les travaux entrepris à ce jour, l'utilisation directe du document n'est pas encore possible : les systèmes d'interrogation de bases documentaires sont peu précis.

Les systèmes classiques actuels associent documents et données de la façon suivante : (1) les documents sont rédigés selon les règles de qualité imposées par l'entreprise. (2) des informations concernant tout nouveau document sont stockées dans une base de données, à l'aide d'une grille de saisie. Cette grille contient des informations contextuelles (auteur, date, version…) et des mots clés permettant de classer ce document. La base de données est donc un index complexe permettant de retrouver des documents.

L'objectif de notre système est différent : nous voulons mettre à jour une base de données " classique " représentant un système d'information à partir de documents saisis librement par l'utilisateur, et non fournir un index d'accès à des documents. La différence par rapport aux systèmes classiques est double. Il faut : (1) éliminer l'étape fastidieuse du remplissage de la grille de saisie, (2) intégrer dans la base de données non seulement des informations contextuelles, mais également des portions de contenu permettant une interrogation sur les valeurs contenues dans les documents.

Une manière d'extraire les informations des documents consisterait à imposer un balisage très précis des informations à insérer dans la base de données, comme illustré sur la figure 1. Ces documents sont appelés des documents centrés données.

Les recherches actuelles du domaine se fondent principalement sur ce type de document. Pour notre part, nous nous plaçons plutôt dans le cadre de documents où les informations à recueillir sont balisées de manière plus "lâche", pour assurer une souplesse lors de la rédaction des documents. Nous nous intéressons à des documents où le balisage encadre des paragraphes et non des informations unitaires (figure 1), nous les appelons des documents centrés paragraphes. Le travail du processus d'analyse ne se restreint alors pas à l'extraction d'informations balisées, mais à la recherche d'informations dans un paragraphe en langage pseudo-naturel. Les termes en gras dans le document orienté paragraphes sont les informations que le système doit retrouver afin de remplir la base de données.

< rencontre id='245'> <diagnostic> **bronchite**, **25/5/00** Le patient se plaint d'une **toux rauque** et de **fièvre** modérée </diagnostic> <antécédent> **père**, **infarctus du myocarde 1997** </ antécédent > <prescription> Donner **3 comprimés** d'**aspirine** 3 fois par jour pendant **10 jours** </prescription> </rencontre>	<rencontre id='245'> <diagnostic> **bronchite** </diagnostic> <date> **25/5/00** </date> <symptôme> <maladie> **toux rauque** </maladie> <maladie> **fièvre** </maladie> </symptôme> <antécédent> <maladie > **infarctus du myocarde** </maladie> <parenté> **père** </parenté> <date>**1997**</date> </antécédent> <prescription> <dosage> **3** </dosage> <unitéDosage> **3** </unitéDosage> <médicament mID='34'>**aspirine** </médicament> <durée>**10**</durée> <unitéDurée> **jour** </ unitéDurée > </prescription> < /rencontre>

Figure 1: Un document rédigé sous une forme centrée données (à droite) et le même document rédigé sous une forme centrée paragraphes (à gauche)

Dans cet article, nous présentons tout d'abord le système DRUID, d'un point de vue fonctionnel. Nous présentons ensuite de manière détaillée le DRUID Core Module, instance principale de notre architecture. Nous expliquons ensuite les choix d'implantation qui ont été faits pour le prototype que nous avons développé. Avant de conclure, nous comparons notre système avec les études les plus proches que nous avons trouvées dans la littérature.

2. Fonctionnalités du système DRUID

Le système DRUID peut se décrire selon les fonctionnalités suivantes :

Pour l'utilisateur final :

- Une saisie de documents orientés paragraphes, conformes à un modèle de documents. Cette saisie se fait sous forme de paragraphes de texte libre, qui sont ensuite balisés manuellement. Dans nos essais, on trouve environ 6 à 8 balises par document. La validation du document centré paragraphes entraîne sa transformation automatique en un document centré données.
- Une interrogation d'une base de données relationnelle remplie à partir du document centré données : avec une interface graphique, l'utilisateur construit des requêtes SQL sur la base de données. Les résultats comportent à la fois les données tabulaires classiques, mais également des liens vers les documents sources des données. Ceci permet à l'utilisateur de consulter le contexte des données retournées par le système, et ainsi de mieux interpréter les réponses reçues de la base.
- Une validation des extractions effectuées. Dans notre prototype actuel, nous demandons à l'utilisateur de valider le document centré données. Au besoin, il peut faire des modifications sur ce document. La validation du document centré données entraîne l'alimentation automatique de la base.

Pour l'adaptation au domaine :

DRUID est un système générique permettant la saisie de documents orientés paragraphes, leur transformation en documents orientés données et l'alimentation d'une base de données relationnelle avec ce dernier document. Pour l'utiliser dans un certain domaine, il est nécessaire d'effectuer en amont une étape de définition des éléments suivants :

- Le modèle de données pour la base. Dans le cas où l'informatisation est nouvelle dans l'organisation cible, il faut définir le modèle de la base de données relationnelle. Dans le cas où cette base existe, il faut l'augmenter de colonnes pour stocker les références aux documents sources de données.
- Les modèles de documents adaptés au domaine. Il faut également définir les modèles de documents permettant l'alimentation de la base et la saisie par l'utilisateur final. Pour cela, nous avons défini un algorithme qui, à partir du modèle de la base de données, construit les modèles de documents adéquats [Laforest, 2003].
- Les règles spécifiques au domaine. L'extraction de données depuis les documents centrés paragraphes se fonde sur des règles de transformation et des motifs d'extraction, qu'il faut rédiger manuellement. Nous avons spécifié un langage de haut niveau pour la spécification des motifs.
- La définition des dictionnaires du domaine. Les dictionnaires standards de la langue française sont intégrés à DRUID. Par contre, il est nécessaire de fournir tout dictionnaire supplémentaire spécifique au domaine d'activité concerné (dictionnaire de médicaments par exemple).

Dans notre prototype DRUID, nous avons intégré les modules suivants :

- une interface utilisateur final lui permettant la rédaction de nouveaux documents, la validation des documents centrés données générés, et l'interrogation de la base
- le module principal concerne la transformation de documents centrés paragraphes en documents centrés données et le stockage des données dans une base relationnelle. Il comporte trois sous-modules :

  1- Un processeur de transformations dont l'objectif est d'extraire les données de chaque document centré paragraphes saisis et de produire un document centré données pour chacun. L'extraction elle-même est sous-traitée à un module d'extraction. Ce processeur y ajoute une sélection des paragraphes, un traitement des données extraites et une restructuration du document brut.

  2- Un processeur d'alimentation qui crée les tuples pour la base de données à partir des documents centrés données construits par le processeur de transformations.

  3- Un processeur d'extraction qui extrait effectivement les données des portions de texte libre. Il se fonde sur une spécification de motifs d'extraction et sur un processeur de langue naturelle (NLP) qui exécute ces motifs.
- un module d'alimentation de la base de données à partir de documents centrés données permet d'inclure les nouvelles données dans la base.
- un système de gestion de bases de données gère la base des données extraites et stocke le fond documentaire. La base de données est relationnelle et structurée. Ce n'est pas un index de mots clés, mais elle est modélisée selon un modèle de données du domaine (tables patient, médecin, médicament, prescription… pour un dossier médical). Nous y avons cependant ajouté des liens vers les documents sources des données par le biais de colonnes contenant des références aux documents. Le fond documentaire contient les documents entiers, sans aucune fragmentation ni indexation. Ce stockage aurait pu être fait sur un système de fichiers ; nous profitons du SGBD pour gérer les droits d'accès aux documents.
- un module de compilation de motifs d'extraction en transducteurs à états finis permet de réécrire les motifs rédigées manuellement dans notre langage de haut niveau en entrées au processeur de langue naturelle.

3. DRUID Core Module

Comme indiqué ci-avant, le DRUID Core Module comporte trois sous-modules, que nous allons détailler dans cette section. Chacun des trois sous-modules (processeurs de transformations, d'extraction et d'alimentation) se base sur des règles pour mener à bien leur mission. Nous verrons donc également les différents

types de règles utilisés par chaque sous-module, ainsi que les langages choisis pour les implanter.

En entrée du DRUID Core Module sont fournis des documents centrés paragraphes. En sortie du DRUID Core Module, on trouve des instructions au SGBD pour mettre à jour la base de données. En interne au DRUID Core Module, des documents centrés données sont construits (voir figure 2). Ces documents sont ajoutés au fond documentaire et pourraient être utilisés comme source de données, et alors être directement interrogés avec les langages d'interrogation dédiés à XML.

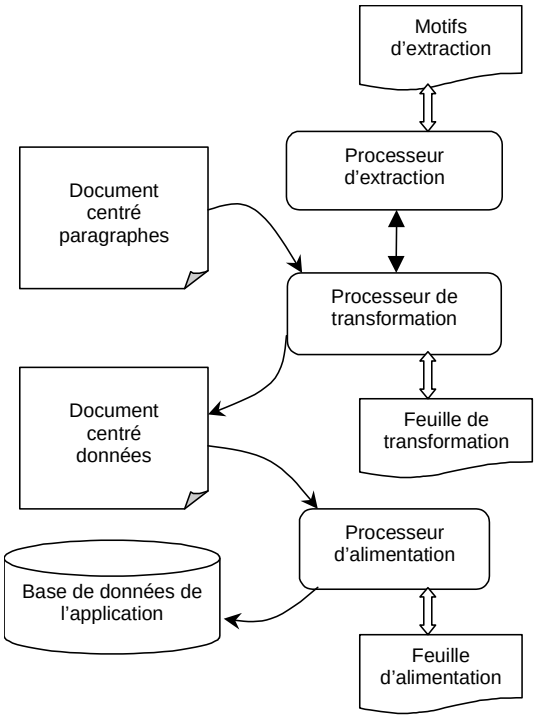


Figure 2 : architecture du DRUID Core Module

3.1. Processeur de transformations

3.1.1. Présentation du processeur de transformations

Le processeur de transformations analyse les paragraphes balisés d'un document centré paragraphes et construit le document centré données correspondant. Le processeur exécute les instructions qui se trouvent dans une feuille de

transformations. Une feuille de transformations contient une séquence de règles qui décrivent comment transformer un document centré paragraphes en un document centré données. Le document centré données ainsi construit est stocké dans le fond documentaire et transmis au processeur d'alimentation.

Pour chaque DTD de documents centrés paragraphes, il existe une feuille de transformation et une DTD de documents centrés données. Dans une feuille de transformations, on distingue quatre types de règles :

- Les règles de sélection permettent de repérer, à l'aide des balises, les paragraphes dans un document centré paragraphes,

- Les règles d'extraction indiquent quels motifs d'extraction utiliser et les types de retour attendus. Ces règles permettent d'activer le processeur d'extraction, et de récupérer ses résultats,

- Les règles de manipulation permettent de manipuler les informations extraites en appliquant des opérations arithmétiques, chaînes de caractères ou booléens,

- Les règles de restructuration permettent de construire un document conforme à une DTD de document centré données en utilisant des opérateurs de manipulation d'arbres.

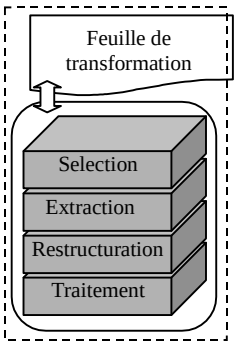


Figure 3 : Processeur de transformations

Nous avons choisi d'implanter le processeur de transformations comme une extension d'un processeur XSLT (Clark, 1999). Ainsi, nous avons pu reprendre un outil existant et ne développer que la partie spécifique supplémentaire.

3.1.2. Règles des feuilles de transformation

Les règles sont écrites dans une extension du langage de description des règles XSL. La figure 4 donne l'extension de la feuille de style XSL pour les feuilles de transformation. Une feuille de transformations contient un élément *transform* par type de paragraphe. L'attribut *paragraph* de cet élément définit la balise XML des paragraphes concernés, et constitue donc la règle de sélection. Le corps de chaque élément *transform* peut contenir une ou plusieurs règles de restructuration pour

ajouter des éléments XML et attributs, une ou plusieurs règles de manipulation pour traiter les informations balisées, et des règles d'extraction associées au paragraphe en question. La figure 5 montre la syntaxe formelle des règles d'extractions (*extract*) et ses sous-éléments (*slot*). Il existe deux variantes de l'élément *extract* ; dans le cas où l'extraction retourne une seule valeur, l'élément *extract* est réduit à un élément XML vide ; dans le cas où l'extraction retourne plusieurs valeurs, l'élément *extract* contient un élément *slot* par information retournée. Un élément *extract* possède une liste d'attributs : l'attribut *pattern* spécifie le nom du slot à retourner, l'attribut *lookupkey* retourne l'ID du slot (au lieu de sa valeur) en consultant le dictionnaire, l'attribut *taggable* permet de baliser ou non la valeur retournée, et enfin l'attribut *att* met la valeur d'un slot comme attribut dans l'élément XML qui l'engendre.

```
                                <!-- Category: top-level-element -->
<ext:transform paragraph = qname >
        un ou plusieurs règles d'extractions, règles de manipulation, règles de
        restructuration
</ext:transform>
                                <!-- Category: instruction -->
<ext:extract pattern = qname lookupkey = ('yes' | 'no') taggable = ('yes' | 'no')  att = qname>
un ou plusieurs éléments slot
</ext:extarct>

<ext:slot name = qname        lookuokey = ('yes' | 'no')      taggable = ('yes' | 'no') />

<ext:construct  frame = qname type = (' fregment' | 'defragment') >
                        un ou plusieurs éléments extract
</ext:construct>
```

Figure 4 : extension de XSL pour la rédaction de feuilles de transformation

Le dernier élément, parmi les extensions proposées à XSL, est l'élément *construct*. Il permet soit de mettre dans le document centré données uniquement les données extraites, soit de conserver l'intégralité du texte du paragraphe en y ajoutant des balises repérant les données extraites.

La figure 5 illustre cette extension par un exemple de feuille de transformation, incluant règles de sélection et d'extraction.

```
….
< ext:transform paragraph="prescription">
  < ext:extract pattern="R_prescription">
    < ext:slot name=" dosage " />
    < ext:slot name=" unitéDosage " />
    < ext:slot name=" médicament " att="mID" />
    < ext:slot name=" médicament " />
    < ext:slot name=" durée " />
    < ext:slot name=" unitéDurée " />
```

```
  </ ext:extract>
</ ext:transform>
< ext:transform paragraph =" antécédent ">
  < ext:antécédent >
    < ext:extract pattern="R_ maladie" />
    < ext:extract pattern="R_ parenté" />
    < ext:extract pattern="R_date" />
  </ ext:antécédent >
 </ ext:transform >
....
```

Figure 5 : Règles d'extractions extraites de la feuilles de transformation pour les paragraphes prescription et antécédent.

3.2. Processeur d'extraction

3.2.1. Présentation du processeur d'extraction

Nous avons choisi de réutiliser le moteur de traitement de la langue naturelle (NLP) Intex (Silberztein, 2000). Notre processeur d'extraction est une interface entre le processeur de transformations et le NLP. Il reçoit un paragraphe rédigé en langue naturelle et doit retourner des termes extraits du paragraphe grâce à l'application d'un motif d'extraction. Avant de faire effectuer l'extraction par le NLP, le processeur d'extraction lance des étapes préliminaires : il demande au NLP de faire (i) une validation du texte (que du texte et pas de caractères de contrôle), (ii) un pré-traitement pour identifier les phrases, délimiter les mots composés non ambigus, et marquer les mots contactés ou élidés, (iii) une analyse lexicale pour appliquer les dictionnaires de la langue et du domaine, et (iv) une désambiguïsation des mots correspondant à plusieurs entrées lexicales dans les dictionnaires. Après ces étapes préliminaires, le processeur d'extraction fournit au NLP le paragraphe à analyser et le motif d'extraction correspondant pour effectuer l'extraction proprement dite.

Les motifs d'extraction sont rédigées par un expert du domaine. Pour aider à la gestion de ces motifs, nous avons défini un langage de spécification à quatre niveaux que nous détaillons ci-après. Les motifs d'extraction sont très dépendants du domaine d'application, et nécessitent un travail préalable non négligeable. Ils intègrent des références à des dictionnaires de la langue française ainsi qu'à des dictionnaires de termes spécifiques au domaine. Ces motifs ne sont pas directement compréhensibles par le processeur de langue naturelle, mais doivent être transformées en transducteurs à états finis (FST). Il est difficile pour un humain de rédiger des FSTs pour des cas complexes, et le nombre de FST nécessaires est important. C'est pourquoi nous avons préféré définir un langage de spécification de plus haut niveau et développer un compilateur permettant de traduire les motifs

définis en FST. Les motifs d'extraction sont conservés pour réutilisation, les FSTs correspondants sont stockés dans un dictionnaire.

3.2.2. Langage de haut niveau pour la spécification de motifs d'extraction

Le langage de haut niveau que nous avons défini permet à l'utilisateur d'écrire simplement des motifs d'extraction d'information. En regard des classiques expressions régulières, ce langage ajoute les fonctionnalités suivantes :

- définir des méta-mots comme <maladie> pour faire référence à des dictionnaires de termes

- faire référence à un motif existant dans la définition d'un nouveau motif

- rédiger en plusieurs couches les motifs, et donc fournir un résultat plus lisible à l'utilisateur qui rédige ou réutilise des motifs.

Pour la rédaction des motifs d'extraction, nous avons défini quatre couches :

La couche 1 est composée d'un ensemble fini de *termes*. Un terme est soit une séquence de lettres délimitée par des séparateurs, soit un méta-mot décrivant un nombre <NB>, un mot <MOT>, un mot vide <E> ou une référence à un dictionnaire <dico>. Un terme peut être nommé, c'est-à-dire que le terme identifié doit être retourné balisé par le nom donné : medic [<medicamentDic>] indique que le terme du dictionnaire medicamentDic doit être balisé par le mot medic.

La couche 2 est composée d'un ensemble fini d'*expressions*. Une expression est une concaténation de termes séparés par des espaces ou des tabulations. On dit qu'une expression est valide dans un paragraphe si on y trouve une séquence de termes correspondant à cette expression.

La couche 3 est constituée d'un ensemble fini de *segments*. Un segment est une liste d'expressions alternatives, exprimant les différentes formes de rédaction d'un tel segment. On dit qu'un segment est valide dans un paragraphe si une de ses expressions est valide dans ce paragraphe.

La couche 4 est constituée d'un ensemble fini de *motifs*. Un motif est un ensemble fini non ordonné de segments. Certains segments sont obligatoires, d'autres sont optionnels. Des contraintes sur les segments peuvent être indiquées, pour réduire le nombre factoriel de combinaisons possibles (cardinalité, ordre entre n segments…). On dit qu'une règle est valide pour un paragraphe si l'ensemble de ses segments obligatoires est valide dans ce paragraphe.

Par exemple, pour définir un motif d'extraction de prescription (figure 6), on définit (i) un motif (pattern) indiquant qu'un paragraphe de prescription contient les segments (slots) dosage, médicament, et optionnellement les segments fréquence, durée et unité pour la durée, (ii) une ou plusieurs expressions indiquant la définition de chaque segment avec ses différentes formes d'écriture possibles. Le segment dosage indique qu'il y a trois expressions possibles pour repérer le dosage tel que '½ comprimé', '1 cachet', '1 et ½ gélule' etc. Les termes comprimé, cachet et gélule

sont recensés dans un dictionnaire et repérés par le meta-mot <unité>. De même, le meta-mot <période> regroupe les termes tels que jour, semaine, mois, etc.

```
expression expDosage1= dosage [<NB> "/" <NB>] <unité> ;
expression expDosage2= dosage [<NB> "et" <NB> "/" <NB>] <unité> ;
expression expDosage3= dosage [<NB>] <unite> ;
slot dosage= expDosage1 | expDosage2 | expDosage3 ;
expression expuDosage1= <NB> "/" <NB> unitéDosage[<unité>];
expression expuDosage2= <NB> unitéDosage[<unité>] ;
slot unitéDosage = expuDosage1| expuDosage2;
slot médicament =  médicament[<medical>] ;
slot durée = 'pendant' durée [<NB>] <période>  ;
slot unitéDurée = 'pendant' <NB> unitéDurée[<période>] ;
 …
pattern prescription = dosage,  médicament, durée?, unitéDurée?,  fréquence? ;
```

Figure 6 : modèle d'un motif d'extraction pour le paragraphe prescription

3.2.3. Transformation des motifs en FST

Le NLP que nous avons choisi pour l'implantation de notre système utilise les transducteurs à états finis (FST) comme format d'entrée des motifs d'extraction. Ces FST sont abrupts à lire pour des cas non triviaux, et c'est pourquoi nous avons préféré fournir à l'utilisateur concepteur un langage de haut niveau pour la rédaction des motifs. La traduction de motifs en FST nécessite l'utilisation d'un compilateur. Nous avons développé un tel compilateur. Il prend en entrée des motifs dans le langage de spécification et retourne en sortie les mêmes motifs traduits en FST. Ce compilateur suit les étapes classiques d'analyse lexicale, syntaxique et sémantique. Il lit les scripts contenant les spécifications des motifs et compile chaque motif d'extraction en un FST en utilisant une procédure récursive bottom up et ceci en incluant à chaque couche les FSTs de couche inférieure : une FST est écrite pour chaque terme, puis pour chaque expression (les FSTs des termes inclus sont mis en série), puis pour chaque segment (les FSTs des expressions correspondantes sont mis en parallèle), et enfin pour chaque motif (on met en parallèle tous les ordres possibles de segments). La figure 7 présente le FST correspondant à la traduction du segment dosage rédigé dans notre langage de spécification à quatre couches.

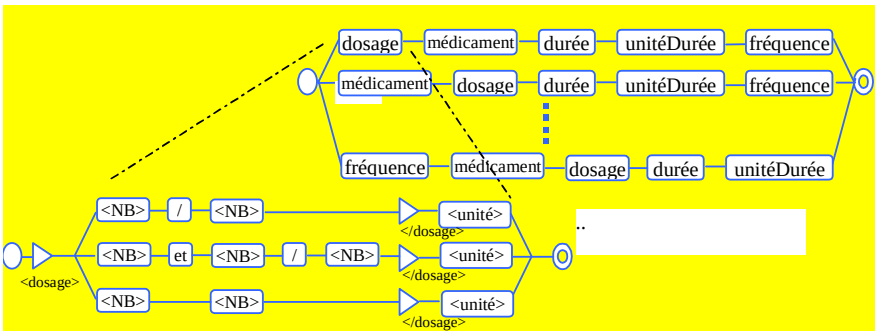


Figure 7 : FST correspondant au segment Dosage de la figure 6

3.3. Processeur d'alimentation

Le processeur d'alimentation permet d'enregistrer les données des documents centrés données dans la base de données relationnelle. Nous avons réutilisé le travail de R. Bourret (XML-DBMS). Ce processeur utilise des règles d'alimentation écrites dans un format XML pour construire une vue objet du document centré données et d'autres pour relier cette vue au modèle de la base de données. Pour plus d'informations, voir (Bourret, 2000). La figure 8 donne un exemple de feuille d'alimentation pour l'exemple des prescriptions.

```
<ClassMap>
 <ElementType Name= "prescription" />
 <ToClassTable>   <Table Name="PrescriptionTable" />  </ToClassTable>
<RelatedClass  >
  <PropertyMap>
     <Attribute Name=" dosage "/>
     <ToColumn> <Column Name="dosage-column"/>  </ToColumn>
 </PropertyMap>
 <PropertyMap>
     <Attribute Name=" unitéDosage "/>
     <ToColumn> <Column Name="udosage-column"/>   </ToColumn>
 </PropertyMap>
<PropertyMap>
     <Attribute Name=" médicamernt "/>
     <ToColumn> <Column Name="Med-column"/>   </ToColumn>
 </PropertyMap>
     ….
```

Figure 8 : Exemple de feuille d'alimentation pour les prescriptions

L'extrait de la feuille d'alimentation montre que l'élément XML *prescription* est relié à la table *PrescriptionTable* dans la base de données et que les éléments XML *dosage*, *unitéDosage*, *médicament…* sont reliés respectivement aux colonnes *dosage-column*, *udosage-column* et *Med-column* dans la table *PrescriptionTable*. Les données extraites du document centré données de la figure 1 alimentent ces colonnes avec les valeurs (*3, comprimé, aspirine*).

4. Prototype

Nous avons développé un prototype qui intègre la saisie, la transformation et l'alimentation des données. Il est écrit en java et se fonde sur les standards XML. Dans cette section, nous présentons rapidement l'implantation de chacun des modules.

- *Interface graphique documentaire* : L'interface utilisateur est écrite en java et utilise la bibliothèque Swing pour les interactions. Elle est écrite avec la technologie Enterprise Java Beans et fonctionne sur une plate-forme weblogic. Les opérations XML de manipulation et validation de documents utilisent le standard DOM, et plus particulièrement l'implantation Apache Xerces (Xerces). Pour plus d'informations, voir (laforest et al., 2002, Badr et al., 2001).

- *Processeur de transformations* : Le processeur de transformations est construit sur XSLT. Le processeur Saxon (Kay, 2002) a été utilisé. Nous l'avons étendu pour intégrer les spécifications des règles des feuilles de transformation. Ce processeur de transformations utilise également JAXP1.1 (JAXP) pour manipuler les documents centrés paragraphes et les documents centrés données. Les fonctionnalités de parsing sont effectuées à l'aide de l'implantation Apache Xerces.

- *Processeur d'extraction et processeur de langue naturelle* : Nous avons choisi le processeur de langue naturelle INTEX (Silberztein, 1993) car il permet de créer des dictionnaires personnels et donc d'intégrer des mots techniques du domaine cible, mais également parce qu'il permet de spécifier des règles d'extraction contenant des termes nommés. De plus, il utilise les FST aussi bien pour décrire les motifs d'extraction que pour modéliser les dictionnaires. Le processeur d'extraction est rédigé dans le langage Java.

- *Compilateur de spécifications de motifs en FSTs* : Nous avons développé un compilateur en java pour transformer les spécifications de motifs d'extractions en quatre couches en finite state transducers. Ce compilateur se fonde sur la bibliothèque JavaCC (Sun, 1997).

- *Processeur d'alimentation* : Comme indiqué précédemment, nous avons réutilisé le middleware XML-DBMS (Bourret, 2000) pour effectuer l'alimentation de la base de données à partir d'un document centré données.

Le prototype a été testé sur une base documentaire constituée de 18 documents centré paragraphes dans le domaine de la cardiologie. Nous avons conçu 60 motifs d'extraction qui couvrent la recherche des informations vitales telles que : les

résultats des examens cliniques, les symptômes dans les ECG, écographies, l'historique du patient, la prescription, et les coordonnées du patient, de la clinique et du médecin traitant. Les règles d'extraction écrites selon le langage de spécification génèrent après compilation 463 FST (expressions et slots).

5. Positionnement par rapport à l'état de l'art

Les recherches les plus proches de notre travail concernent les wrappers. De nombreuses études ont été menées sur le sujet pour la construction de wrappers de pages HTML (Crescent *et al.,* 2001), de documents XML (Muslea, 1999), ou pour les textes en langue naturelle (Agichtein *et al.,* 2000). Cependant, la plupart des wrappers ne parviennent pas à effectuer une extraction efficace des données dans le cas où la structure n'est pas complètement régulière. Or dans les applications réelles, il est difficile de trouver des formes syntaxiques régulières, comme des délimiteurs de portions de texte ou de définir des connaissances linguistiques complètes pour extraire correctement des données.

A l'opposé, notre processeur d'extraction permet d'un part la recherche d'informations dans des paragraphes narratifs ou de style télégraphique et d'autre part a recours à des dictionnaires linguistiques et techniques, ce qui augmente le taux d'extraction des données.

Un autre domaine de recherche ayant trait à notre problématique concerne l'interrogation et la gestion de données dans des documents XML, et plus précisément dans des documents centrés données. Cela consiste la plupart du temps en la mise en correspondance de données irrégulières avec un schéma traditionnel de base de données. Ce type de mapping est proposé par de nombreux systèmes, tels Stored (Deutch *et al.,* 1999), UnQL (Buneman *et al.,* 1996), Lorel (Abiteboul *et al.,* 1997), struQL (Fernandez *et al.,* 1998) ou OQL-Doc (Abiteboul *et al.,* 1996).

Dans notre approche, nous utilisons une base de données relationnelle et pouvons donc profiter de la maturité de SQL pour manipuler les données avec des requêtes complexes (étude statistique, analytique, mining, etc) et pour sélectionner les documents correspondants.

De nos jours, les documents centrés paragraphes sont de plus en plus utilisés. La structure de tels documents définit des régions d'intérêt et les données pertinentes extraites de ces régions doivent être conformes à une sémantique précise. Peu de systèmes ont été conçus pour la gestion de tels documents. NoDoSE (Adelberg, 1998) fournit un système semi-automatique pour l'extraction de données dans des instances d'un type de document. Il supporte complètement les documents en texte libre, les documents HTML et en partie les documents XML centrés paragraphes. Les données extraites sont stockées dans un schéma relationnel traditionnel. Sur la base d'ontologies, (Embley *et al.,* 1998) formule des règles permettant l'extraction

de données pertinentes et applique un système de reconnaissance pour construire des valeurs de tuples d'attributs qui sont insérés dans un schéma de base de données. Cette approche est limitée à des ontologies relativement petites et à des ensembles de mots clés petits.

Notre système partage avec NoDoSE la même vision basée sur l'extraction des données a partir de documents pour alimenter une base de données. Mais les deux systèmes restent différents : nous utilisons les documents comme interface de saisie et alimentons une application existante, alors que NoDoSE n'alimente pas une base de données, il présente les données extraites sous forme structurée dans des fichiers. Bien que notre travail soit basé sur des dictionnaires pour extraire les informations, le travail d'Embly est plus ambitieux : il utilise des ontologies pour modéliser une application et ensuite pour générer automatiquement les motifs d'extraction. Ces motifs ne couvrent pas bien la richesse du langage humaine ; l'intervention de l'être humain est indispensable pour lever les ambiguïtés.

Enfin, nous mentionnerons l'utilisation d'automates à états finis pour les processeurs de langue naturelle (Gross, 1989) et pour l'extraction d'information. De nombreuses propositions d'outils ont été faites dans la communauté linguistique, tels FS5 (Karttunen *et al.*, 2001) et Xerox (Karttunen, 2000). Dans la communauté française, INTEX (Silberztein, 1993) semble être l'outil le plus utilisé.

6. Conclusion

Le système que nous proposons associe une base documentaire à une base de données. Il définit un outil permettant à l'utilisateur de travailler sur des documents, sans se préoccuper de la manière dont les données sont intégrées dans la base de données. Ce système permet d'analyser des documents semi-structurés, afin d'en extraire les informations qui doivent être stockées dans la base de données. Pour cela, il s'appuie sur des règles de transformation, des règles d'alimentation et des motifs d'extraction. Des requêtes sur la base de données permettent de retrouver les documents, ou d'effectuer des traitements statistiques sur les données. La technique proposée dans ce travail est générique et peut s'appliquer à de nombreux domaines, dès lors qu'ils manipulent des documents centrés paragraphes, et qu'une interrogation plus fine que par mots-clés est nécessaire. Les règles et les motifs nécessaires sont très dépendants de l'application cible et doivent être rédigés pour chaque implantation.

Les documents centrés données sont ajoutés au fond documentaire et pourraient être utilisés comme source de données, et alors être directement interrogés avec les langages d'interrogation dédiés à XML. Nous n'avons pas aujourd'hui implanté ceci, mais il semble qu'aucune difficulté technique n'est à redouter. Il serait cependant intéressant de le faire pour comparer l'utilisation d'une base de données relationnelle à l'utilisation d'une base de documents structurés.

Dans l'état actuel de notre travail, les règles d'alimentation sont rédigées manuellement par le concepteur des règles d'extraction. Nous travaillons à la rédaction automatique de ces règles. Ceci est relié à la génération automatisée des DTD de documents centrés paragraphes et centrés données à partir du schéma de la base de données, pour laquelle nous avons développé un algorithme (Laforest *et al.*, 2003).

Dans un avenir proche, nous voulons étudier plus précisément les possibilités d'automatisation de la construction des feuilles de transformation et d'alimentation. Des squelettes de ces feuilles pourraient certainement être produits automatiquement à partir du schéma de la base de données sous-jacente et ainsi réduire le travail de l'utilisateur chargé de l'adaptation du système à un domaine particulier.

Notre système ne permet pas de gérer des images, du son ou de la vidéo dans la base de données, mais se restreint pour l'instant à la gestion d'informations textuelles. Il permet d'intégrer une analyse des commentaires associés aux objets multimédia, mais ne permet pas une indexation du contenu de ces objets. Il serait cependant intéressant de coupler notre approche à des analyses d'images et de vidéo, et permettre ainsi une indexation par le contenu et par les commentaires. Une première étude a été menée dans ce sens (Badr *et al.*, 2003).

D'autres pistes d'étude peuvent également être explorées, comme la spécification de règles sémantiques pour la validation des données saisies (ne pas accepter une prescription du style « 3 cachets de sirop ») ou l'interrogation de la base par l'exemple, en fournissant un document modèle (il faudrait alors définir une distance entre documents pour retourner les documents les plus proches du document modèle).

7. Bibliographie

Abiteboul S., Cluet S., Christophides V., Milo T., Moerkotte G., and Simeon J.. Querying Documents in Object Databases (OQL-Doc), 1996.

Abiteboul S., Quass D., Widom J., and Wiener J.L.. The lorel query language for semi-structured data. International Journal on Digital Libraries 1997.

Adelberg B. NoDoSE: a tool for semi-Automatically Extracing Structured and Semistructured Data from Text Documents. Proceeding of ACM SIGMOD international conference on Management of data, 1998, pp 283-94.

Agichtein E., Eskin E. and Gravano L. Combining Strategies for Extracting Relations from Text. Collections, in the Proceedings of the 2000 ACM SIGMOD Workshop on Data Mining and Knowledge Discovery (DMKD 2000)

Badr Y., Chbeir R., Flory A. Toward an automatic image description. Proc. Int conf. on computer science, software engineering, information technology, e-business and applications CSITeA'03 pp. 242-7, 2003

Badr Y., Sayah M., Laforest F., Flory A. Transformation rules from semi-structured XML documents to database model. AICSSA 2001, Beirut, Lebanon. June 26-29, 2001

Bourret, R., Bornhövd, C., Buchmann, A.P.: A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. 2nd Int. Workshop on Advanced Issues of EC and Web-based Information Systems (WECWIS), San Jose, California, June, 2000

Buneman P., Davidson S., Hillebrand G., and Succiu D. A query language and optimization techniques for unstructured data (UnQL). In Proc. SIGMOD Conf., 1996.

Clark J. (ed.) “XML transformations (XSLT) version 1.0” W3C, November 1999, http://w3c.org/TR/xslt

Crescenzi V., Mecca G., Merialdo P. RoadRunner: Towards Automatic Data Extraction from Large Web Sites, 27th International Conference on Very Large Databases 2001.

Deutsch A., Fernandez M., Suciu D. Storing semistructured data with STORED, In Proc of the ACM SIGMOD International Conference on Management of Data , 1999.

Embley D., Campbell D., Smith R., Liddle S.: Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. CIKM 1998: 52-59

Fernandez M., Florescu D., Kang J., Levy A., Suciu D. Catching the boat with Strudel: experiences with a web-site management system. In Proc of ACM-SIGMOD Int. Conference on Management of Data , 1998

Gross M. The Use of Finite Automata in the Lexical Representation of Natural Language. Electronic Dictionaries and Automata in Computational Linguistics. In Lecture Notes in Computer Science, pages 34-50, Springer-Verlag, Berlin, Germany, 1989.

JAXP Java Community ProcessSM. Java APIs for XML parsing (JAXP). Project home page. http://java.sun.com/xml/xml_jaxp.html.

Karttunen L., Koskenniemi K., Van Noord G. (editors), Finite State Methods in Natural Language Processing. FSMNLP 2001. Extended Abstracts. ESSLLI Workshop, Helsinki 2001

Karttunen L.. Applications of Finite-State Transducers in Natural Language Pr of CIAA-2000. Lecture Notes in Computer Science. Springer Verlag.

Kay M. SAXON XSLT Processor 7.2, http://saxon.sourceforge.net/ , August 2002.

Laforest F., Boumediene M. Study of the automatic construction of XML documents models from a relational data model. WebS workshop, DEXA’2003, Prague, sept 2003

Laforest F., Flory A. Using Weakly Structured Documents at the User-Interface Level to Fill in a Classical Database. Chapter in: Advanced Topics in Database Research, K. Siau (ed.) Idea Group Publishing, 2002.

Muslea I. Extraction patterns for information extraction tasks: A survey. In AAAI-99 Workshop on machine learning for information extraction, 1999.

Robie J., Lapp J., Schach D. XML Query Language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html, 1998

Silberztein M. Dictionnaires Electroniques et Analyse Lexicale du Français--Le Système INTEX, Masson, Paris, France. 1993.

Silberztein M. INTEX: an FST toolbox. Theoretical Computer Science, (234)33-46. 2000.

SunTest JavaCC - A Java Parser Generator, 1997

Xerces Apache XML Project. Xerces XML Parser for Java. http://xml.apache.org/xerces-j

Projet SEFAGI

[15]
Tarak Chaari and Frédérique Laforest. Génération et adaptation automatiques des interfaces utilisateurs pour des environnements multiterminaux. le projet sefagi. *Ingénierie des Systèmes d'Information, n° spécial Systèmes d'information pervasifs*, 9(2):11–38, 2004.

Génération et adaptation automatiques des interfaces utilisateurs pour des environnements multi-terminaux

Projet SEFAGI : Simple Environment For Adaptable Graphical Interfaces

Tarak Chaari, Frédérique Laforest

Laboratoire d'InfoRmatique en Images et Systèmes d'information, CNRS FRE 2672
Institut National des Sciences Appliquées de Lyon
Bâtiment Blaise Pascal (502)
20 Avenue Albert Einstein
69621 Villeurbanne cedex
{tarak.chaari, frederique.laforest}@insa-lyon.fr

RÉSUMÉ. *Le projet SEFAGI (Simple Environment For Adaptable Graphical Interfaces) s'intéresse à l'adaptation des applications à des environnements multi-terminaux. Notre objectif est double : nous voulons (i) permettre à l'utilisateur final (ou à un utilisateur-concepteur) de décrire ses interfaces, et lui générer entièrement le code correspondant et (ii) rendre la génération du code de l'interface adaptable à tous types de terminaux. Ces interfaces doivent interagir avec des traitements distants variés (calculs mathématiques, mise à jour d'une base de données, téléchargement d'une image…). Ceci inclut la gestion des interactions entre composants graphiques et services distants, ainsi que la mise en place d'adaptations à différents niveaux. Nous avons spécifié, conçu et développé une plate-forme complète qui nous a permis de valider les principes élaborés dans ce travail.*

ABSTRACT. *The SEFAGI project (Simple Environment For Adaptable Graphical Interfaces) is interested in designing an architecture to adapt applications to multi-terminal environments. We have a double objective: we aim at (i) allowing end-users describe the user interfaces they need, and entirely generate the corresponding code and (ii) make code generation adaptable to any kind of terminal. It has to provide interactions between graphical components and several types of distant services (mathematical computing, database update, image download...) and to make several adaptations at different levels. We have specified, designed and developed a complete platform that has allowed us to validate the principles of our proposal.*

MOTS-CLÉS : *IHM, génération automatique, adaptation de contenu, systèmes d'information pour terminaux mobiles, services WEB, XML, terminaux mobiles.*

KEYWORDS: *GUI, automatic generation, content adaptation, information systems for mobile terminals, WEB services, XML, mobile terminals.*

1. Introduction

L'évolution technologique des terminaux mobiles et leur disponibilité sur le marché nous incitent à intégrer ce type d'appareils dans nos systèmes d'information. En effet, l'utilisateur veut avoir l'information le plus vite possible, partout et à tout moment. Ainsi un nouveau domaine d'exploitation des systèmes ambulatoires apparaît. Le téléphone mobile ne sert plus uniquement à de simples communications vocales ou textuelles. Les assistants personnels ne sont plus de simples gadgets de planning et d'organisation. Les ordinateurs de poche ne sont plus isolés de l'Internet. Ces appareils interagissent maintenant avec des services implantés sur des serveurs d'applications divers.

Cependant, plusieurs problèmes ont surgi lors de l'exploitation de ces terminaux dans les systèmes d'information pervasifs (Birnbaum, 1997). Premièrement, ces appareils ont des performances très lointaines de celles d'un vrai serveur ou d'une station de travail. En effet, la plupart d'entre eux ne dépassent pas 3 Mo de mémoire vive et morte réunies. De plus, la bande passante de transmission est faible et la taille de l'écran est réduite. Enfin, le handicap majeur reste la diversité de ces appareils. Le fait de concevoir plusieurs versions de la même application pour chaque terminal semble une solution pénible et non adéquate vue la croissance étonnante des types de terminaux.

Dans ce contexte, notre objectif est double : nous voulons (i) permettre à l'utilisateur de décrire ses interfaces, et lui générer entièrement le code correspondant et (ii) rendre la génération du code de l'interface adaptable à tout type de terminaux (taille, résolution, nombre de couleurs de l'écran, mémoire disponible, configuration logicielle…). Il serait également intéressant d'étudier les possibilités et spécificités apportées par les différentes modalités d'interaction disponibles (clavier, souris, stylet sur écran tactile…), mais notre travail ne se penche pas sur ces aspects pour le moment. Ces interfaces doivent interagir avec des traitements distants variés (calculs mathématiques, mise à jour d'une base de données, téléchargement d'une image…). En partant d'une description générique (indépendante des terminaux cibles) des composants constituant les interfaces graphiques d'une application, notre architecture génère la totalité du code source des différents écrans constituant cette application. La génération se fait d'une façon adaptative respectant les contraintes matérielles et logicielles imposées par les terminaux cibles. Ensuite lors de l'exploitation des interfaces utilisateurs générées, nous proposons une solution d'adaptation dynamique des données échangées entre le terminal hébergeant ces interfaces et les serveurs de traitements. Cette adaptation doit conserver la valeur sémantique de la donnée afin qu'elle soit toujours utile pour l'utilisateur final de l'application. Nous avons prédéfini un ensemble de composants et un ensemble de traitements génériques (visualisation d'images, requêtes sur une base de données…) pour faciliter la tâche de création de l'interface à l'utilisateur. Ainsi ce dernier choisit seulement les composants constituant son interface et les traitements qu'il veut exécuter parmi ceux prédéfinis. Les associations entre les composants et les

entrées/sorties des traitements se font automatiquement par le système. Cependant, l'utilisateur peut intervenir et modifier ces associations.

Nous avons spécifié, conçu et développé une plate-forme complète qui nous a permis de valider les principes élaborés dans ce travail. La *modularité* et l'*adaptation* ont été les principes de base pour résoudre notre problématique, que ce soit au niveau du code de notre plate-forme ou du code des interfaces générées par cette plate-forme. La modularité consiste en premier lieu à classifier les terminaux selon leurs configurations matérielles et logicielles et leurs profils d'utilisation. De plus, elle permet de séparer les différentes facettes des applications notamment les données, les traitements et les interfaces graphiques. Les traitements sont exécutés par un serveur. Ensuite on adapte les données issues de ces traitements au terminal cible. De même un système d'adaptation d'interfaces graphiques garantit l'affichage adéquat des données. L'architecture que nous proposons facilite l'intégration d'applications sur une large variété de terminaux mobiles, encourageant ainsi l'exploitation de l'informatique mobile.

Dans les interfaces homme – machine nous distinguons deux niveaux d'interaction avec l'utilisateur : le niveau physique (dispositifs matériels) et le niveau logique (composants logiciels). Nous entendons par niveau logique les associations entre composants logiciels (en termes de fonctionnalités plus que de présentation) et services. Ce niveau logique est un intermédiaire entre les spécifications des modèles tâche-utilisateur et l'implantation réelle des interfaces. Le long de notre travail nous ne nous intéressons pas au niveau physique. Par exemple, en entrée on ne s'intéresse pas à comment la zone texte est remplie (au clavier ou à travers un logiciel de reconnaissance vocale) mais plutôt aux fonctionnalités du composant logiciel « zone texte » lui-même. De même, en sortie on ne s'intéresse pas à la taille physique de l'écran mais à la taille du conteneur graphique responsable de l'affichage sur l'écran physique. Ainsi, dans la suite de cet article, le terme « écran » désigne une fenêtre affichée sur l'écran physique du terminal.

La littérature (Nigay, 1995) différencie deux types d'interfaces : les interfaces en entrée (dispositifs matériels ou outils logiciels pour la saisie) et les interfaces en sortie (dispositifs matériels ou outils logiciels pour la présentation des données). Dans ce travail nous ne faisons pas cette différence. En effet, les critères de choix d'un composant - qu'il soit d'entrée ou de sortie - sont les mêmes : l'ergonomie et le confort d'exploitation du composant par l'utilisateur final.

Dans cet article, nous commençons classiquement par présenter un état de l'art des travaux relatifs à notre problématique. La section 3 est le cœur de l'article. Nous y présentons d'abord le principe du projet SEFAGI, puis l'architecture de cette plate-forme. Ensuite, nous détaillons le langage de description générique des interfaces graphiques que nous avons défini ainsi que le processus de génération et d'adaptation des interfaces décrites dans ce langage. Avant de conclure, nous terminons par une présentation de l'implantation effectuée illustrée par un exemple complet.

2. Etat de l'art

Le problème d'adaptabilité ou d'adaptation n'est pas nouveau dans les systèmes d'information. En effet, plusieurs travaux ont touché ce domaine. Dans (Layaïda, 1999) on trouve la définition : « Par adaptabilité, on entend les moyens automatiques ou semi-automatiques qui permettent aux contenus […] d'être utilisables sur des terminaux ayant des caractéristiques et des ressources très variées. ». Sur l'utilité de l'adaptation, M. Satyanarayanan a dit dans (Satyanarayanan, 2001) « L'adaptation est nécessaire quand il y a une disparité significative entre l'offre et la demande d'une ressource. ». Beaucoup de travaux ont montré l'intérêt de l'adaptabilité - ou plasticité (Calvary *et al*, 2002) - des interfaces homme – machine aux changements du contexte d'évolution de ces interfaces (Calvary *et al*, 2001). Dans ces travaux on distingue deux grands types d'adaptation : l'adaptation statique et l'adaptation dynamique. Dans ce paragraphe, nous présentons ces deux méthodes d'adaptation avec leurs avantages et leurs inconvénients. Pour chaque type, nous mentionnons quelques projets existants. Finalement, nous faisons un petit tour d'horizon des technologies actuelles utilisées dans les systèmes d'information pervasifs.

2.1. Adaptation statique vs adaptation dynamique

2.1.1. Adaptation statique

L'adaptation statique ou manuelle consiste à préparer plusieurs versions de la même ressource avant son exploitation. Cette ressource peut être un document, une image, une vidéo ou même une application. Initialement, elle a été faite pour être exploitée par les terminaux standards. Ensuite on applique des transformations manuelles (généralement un codage manuel) sur cette ressource pour qu'elle puisse être exploitée par d'autres types de terminaux.

Cette technique a été adoptée pour plusieurs développements au début de l'apparition des terminaux mobiles. (Larson, 2002), (Massé, 2002) et (Benjaminsen *et al*, 2002) sont des projets d'applications qui suivent l'approche d'adaptation statique. Une première version de ces applications existait déjà pour les micro-ordinateurs standards. Une version équivalente dédiée aux terminaux mobiles a été mise en œuvre de façon manuelle.

Avec une adaptation statique, un développement dédié au type du terminal assure la sûreté de fonctionnement et d'exploitation de la ressource. Elle présente aussi une solution simple et efficace où chaque fournisseur s'occupe de l'adaptation de la ressource pour ses terminaux. Cependant, elle présente l'inconvénient de la difficulté d'évolution des applications. En effet, l'adaptation de la ressource doit passer par toutes les étapes classiques de création d'un logiciel notamment la spécification, la conception, l'implémentation et le test. Le temps de développement est devenu un facteur critique dans les nouveaux systèmes d'information. En outre, l'explosion des

types de terminaux et l'absence de normes et de considérations communes diminuent l'utilité de cette solution.

2.1.2. Adaptation dynamique

Contrairement à l'approche précédente, l'adaptation dynamique effectue les transformations sur la ressource au cours de son utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques matérielles du terminal demandant la ressource.

Soluphone Santé (TeraCom, 2002), OPERA (Lemlouma *et al*, 2001) et MUSA (Menkhaus, 2002) sont des applications où les données sont adaptées dynamiquement au terminal cible. Elles présentent une adaptation des pages WEB pour des différents terminaux en agissant sur la composition logique, spatiale, temporelle et hypertexte de ces documents.

Avec une adaptation dynamique, on gagne un temps de développement important puisque la solution est générée automatiquement à la demande de l'utilisateur. De plus, l'adaptation dynamique adresse le problème de diversité des terminaux mobiles. Cependant, des problèmes d'adaptation peuvent surgir lors de la connexion d'un terminal non reconnu par le système. En effet, on n'est pas vraiment sûr que la ressource adaptée va fonctionner à cent pour cent sur le terminal. En outre, le temps perdu pour l'adaptation de la ressource peut gêner l'utilisateur puisque la connexion sans fil est déjà très limitée en bande passante, ce qui augmente la latence de transmission.

2.2. La solution WEB du HTML au WML

Les solutions précédentes présentent des interfaces graphiques sous forme de pages WEB. La première tentative d'exploitation de ce genre de solutions sur les terminaux mobiles est le WAP (WAP Forum, 1999a). Cette technologie nous a offert un ensemble de services supplémentaires en rendant l'Internet disponible sur ces petits appareils. Les documents échangés sont au format WML (WAP Forum, 1999a). D'autres langages de scripts viennent donner plus de dynamisme à ce genre de format comme WMLScript (WAP Forum, 1999b) qui permet d'exécuter des primitives simples sur le terminal. Cependant, l'affichage est généralement restreint à des données textuelles. Un autre protocole IMODE (Mori, 2000) donne plus de possibilités d'interaction avec l'utilisateur et assure un affichage plus évolué à travers son format d'échange C-HTML (compact HTML) (Kamada, 1998).

Ces technologies web évitent le problème de diversité des terminaux puisque la majorité d'entre eux sont dotés de navigateurs qui peuvent afficher le flux WML ou C-HTML envoyé par le serveur. Cependant, l'absence de normalisation et la diversité des versions de ces langages nous obligent à effectuer des adaptations ad-

hoc à chaque type de terminaux. Par exemple, plusieurs navigateurs n'offrent pas la possibilité d'utiliser les cadres, d'autres ne permettent pas d'exécuter toutes les versions des scripts…

Les langages WEB sont très limités et les scripts additionnels ne permettent pas d'exécuter des traitements complexes. Les interfaces graphiques résultant de ce type de solutions présentent une faible interaction avec les utilisateurs. De plus, on ne peut pas accéder aux ressources matérielles des terminaux, donc il est par exemple difficile d'adapter l'interface à la taille de l'écran (qui reste inconnue).

2.3. Microsoft .net vs Sun Java

Microsoft a proposé une solution permettant de développer des applications complètes pour les terminaux mobiles. Elle s'est inspirée de la technologie .NET (Narayanaswamy, 2002) pour les micro-ordinateurs standards pour la porter sur les petits appareils. Microsoft .NET Compact Framework (Yao, 2002) est un outil permettant de générer des interfaces utilisateurs où on peut utiliser un ensemble de composants graphiques assurant une haute interaction avec l'utilisateur et avec des serveurs de traitements distants. Contrairement aux solutions WEB, les interfaces graphiques programmées donnent plus de fonctionnalités et plus de modalités d'interaction pour l'utilisateur (Thevenin *et al*., 2000). Ainsi ce dernier sera un acteur permanent du système et son rôle n'est plus passif comme dans le cas des pages WEB où le nombre d'entrées est négligeable devant le volume des sorties.

Le problème de l'outil de Microsoft est qu'il génère des interfaces graphiques très dépendantes de la plate-forme Microsoft. Il faut avoir le système d'exploitation Windows CE sur le terminal pour espérer exploiter ces applications. En outre, Microsoft .NET Compact Framework permet de générer seulement une partie du code. On aura donc toujours besoin d'un développement pour avoir des interfaces graphiques complètement fonctionnelles. Par exemple, la liaison entre l'appel d'un service et l'ensemble des composants graphiques concernés doit être codée manuellement. Ceci ne correspond pas à nos objectifs, dans lesquels on a spécifié que la génération des interfaces utilisateurs doit être faite d'une manière automatique sans intervention d'un expert de la programmation.

J2ME (Java 2 Micro Edition) (Piroumian, 2002) est l'API Java dédié aux terminaux mobiles. Elle résout le problème de non portabilité présent dans la solution Microsoft. J2ME est une version allégée de J2SE (Java 2 Standard Edition). Cette plate-forme est basée sur le concept de configurations et la notion de profils (Muchow, 2001). Les configurations J2ME constituent une classification des terminaux suivant leurs caractéristiques matérielles (mémoire disponible, type de connexion…) alors que les profils consistent à une classification orientée application ou utilisations (interfaces utilisateurs, applications réseaux…). La configuration et le profil les plus connus sont respectivement Connected Limited Device Configuration (CLDC) et Mobile Information Device Profile (MIDP) (Riggs *et al*., 2001).

2.4. Langages existants de description d'interfaces graphiques

Pour pouvoir présenter des interfaces graphiques sur des terminaux hétérogènes, il est rentable de partir d'un modèle générique des écrans à afficher. Plusieurs langages de description existent. Le modèle le plus connu est UIML (User Interface Markup Language) (Abrams *et al.*, 1999). Harmonia (Boshernitsan, 2001) est un environnement de génération d'interfaces graphiques en Java ou HTML, à partir de leurs descriptions en UIML. Ce langage est devenu un standard vu sa généricité et son adaptabilité à toutes les interfaces graphiques possibles. Cependant, ce langage présente un inconvénient majeur : la granularité trop fine des descriptions de ces interfaces rend l'implantation d'un tel modèle très complexe. La complexité de la description augmente exponentiellement avec la complexité de l'interface voulue. Avec un tel modèle, il n'y a pas une grande distance entre décrire l'interface ou la programmer.

Dans notre projet, l'approche que nous avons développée vise à utiliser un modèle de description plus convivial en utilisant un niveau de granularité plus gros. Ceci facilite principalement la tâche de l'utilisateur pour la description de nouvelles interfaces, mais aussi celle du générateur d'interfaces.

2.5. Les services WEB

Pour réaliser notre architecture, les traitements doivent être hébergés par des serveurs d'applications. Ces services représentent des produits pour les serveurs et des points d'entrée pour les clients. Les services WEB (W3C, 2002) constituent un standard indépendant de la plate-forme de programmation. Généralement, un service WEB est limité à une ou plusieurs procédures qui peuvent être invoquées à distance. Le langage de description des services WEB WSDL (W3C, 2001) est un moyen de connaître les paramètres nécessaires à l'exécution de ces procédures. Les services WEB utilisent le protocole SOAP (W3C, 2000) (Simple Object Access Protocol) pour invoquer des méthodes sur des objets distants. Plus généralement, ils utilisent XML-RPC (W3C, 1999) pour appeler des procédures distantes. Les services WEB présentent l'avantage de lier des bouts de code hétérogènes en parlant un même langage : XML (W3C, 1998). Le protocole de transport utilisé par les services WEB est HTTP, ce qui implique une disponibilité et une compatibilité très larges (passage facile à travers les firewalls…).

L'utilisation des services WEB dans une telle problématique nous facilite la tâche en créant un middleware standardisé entre les mobiles et le serveur. Les services qu'on peut offrir à ces terminaux peuvent être soit des services métier (liste des chirurgiens d'un hôpital) ou génériques (exécution d'une requête sur une base de données).

2.6. Conclusion

On remarque que dans toutes les solutions existantes, on a soit une adaptation statique très pénible à faire vu la diversité des terminaux soit une adaptation dynamique des pages WEB qui sont très limitées en interaction homme-machine. Nous avons conçu et développé un système qui permet le déploiement d'interfaces graphiques programmées plus riches en fonctionnalités que les pages WEB. Notre approche consiste à séparer les données, les traitements et les présentations des applications. Ceci nous permet d'établir une adaptation automatique des interfaces graphiques en utilisant un moteur de génération adaptatif et une adaptation dynamique des données en s'appuyant sur des services d'adaptation de données.

3. Notre proposition : Le projet SEFAGI

3.1. Présentation générale

Dans notre projet SEFAGI, nous nous plaçons dans le contexte de domaines d'applications où le nombre de profils d'utilisateurs est élevé, et où les besoins en interfaces graphiques personnalisées sont importants. Par exemple, dans le cadre du suivi de patients hospitalisés à domicile, les acteurs participant à la prise en charge du patient et de la maladie sont nombreux : le médecin hospitalier spécialiste, le médecin de ville, l'infirmière à domicile, l'assistante sociale, le kinésithérapeute, le pharmacien, et le patient lui-même : tous ces acteurs interviennent et ont donc besoin d'un accès au dossier du patient. Cependant, les besoins et droits sont très différents d'un acteur à l'autre, leurs modes d'intervention également. La pathologie suivie est un autre facteur à prendre en compte dans l'élaboration des interfaces : le suivi d'une dialyse à domicile n'a rien à voir avec le suivi d'un cancer en fin de vie. De plus chaque intervenant peut avoir besoin d'utiliser les mêmes interfaces de la même application sur des terminaux différents (PC au cabinet, téléphone mobile ou PDA en visite). C'est par exemple le cas du médecin généraliste qui consulte à son cabinet principalement, mais peut aussi rendre visite au malade chez lui. Si l'on veut satisfaire les besoins et droits de chacun, le nombre d'interfaces à développer est très important, et requiert un temps non négligeable.

Dans notre projet, nous proposons une toute autre façon de considérer le problème. Connaissant en premier lieu les données qui seront manipulées, et les services attendus par les acteurs, nous préconisons de commencer l'implantation par celle de la base de données et des services. Ensuite, nous fournissons aux utilisateurs (ou à un utilisateur de référence connaissant bien les besoins de chacun) les moyens d'intégrer à tout moment une nouvelle interface utilisateur pour la communication avec les services développés. Un assistant graphique permet à un utilisateur final de décrire une nouvelle interface graphique qu'il veut utiliser, et lie les composants de

cette interface aux services correspondants. L'assistant graphique se fonde sur un langage de description des interfaces de haut niveau, ce qui induit peu de travail pour l'utilisateur final. Un annuaire de services lui est associé, ce qui permet à l'utilisateur de parcourir l'ensemble des services de façon organisée et documentée.

Lorsque la description de l'interface est terminée, le code correspondant est entièrement et automatiquement construit par le générateur d'interfaces. Le code construit est en fait multiple : on trouve une interface pour les terminaux standards, et une interface pour les terminaux mobiles. Des règles de transformation décrites dans le générateur d'interfaces nous ont permis de définir des correspondances entre les concepts des interfaces sur terminaux standards et ceux des interfaces sur terminaux mobiles (affichage, navigation, boutons de commande…). Les interfaces ainsi générées peuvent ensuite être déployées sur les terminaux cibles.

L'utilisation des interfaces générées s'effectue au sein d'un environnement d'exécution. Nous avons construit un environnement d'exécution pour les terminaux standards et un pour les terminaux mobiles. Lors de l'exécution, les interfaces font appel aux services sous-jacents définis lors de leur conception. Des adaptations de contenu sont nécessaires à ce niveau, pour permettre la transmission et la gestion des données sur les terminaux mobiles.

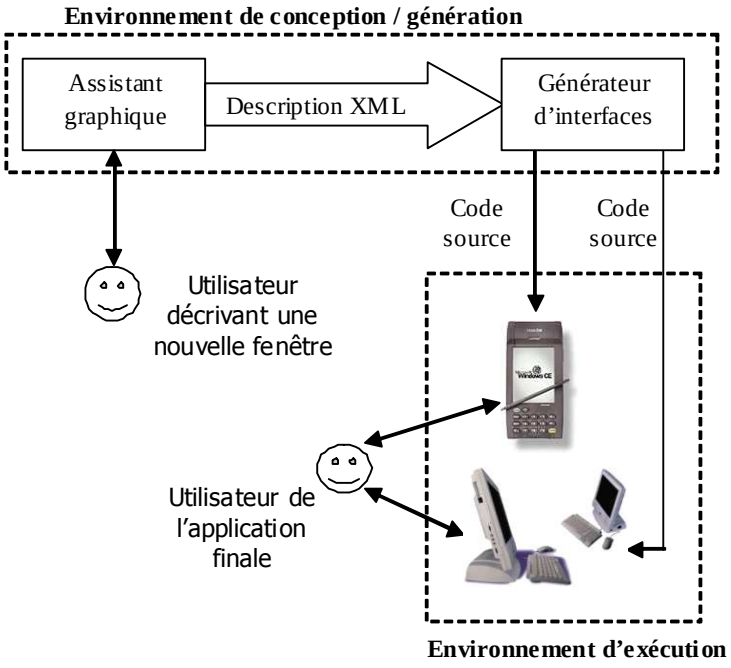


Figure 1. Présentation générale du projet SEFAGI

3.2. *Architecture*

Notre architecture se base sur trois entités principales : un serveur d'interfaces, un terminal et un serveur d'applications qui sont présentés ci-dessous. La figure 2 présente l'architecture de notre proposition.

3.1.1. Serveur d'interfaces

Le serveur d'interfaces est responsable de l'adaptation et de la génération des interfaces graphiques pour les différents terminaux. Ce serveur contient un environnement de génération qui prend en entrée la description générique de l'interface à générer. Pour chaque architecture cible, des règles de transformations permettent d'avoir un affichage adapté au terminal concerné.

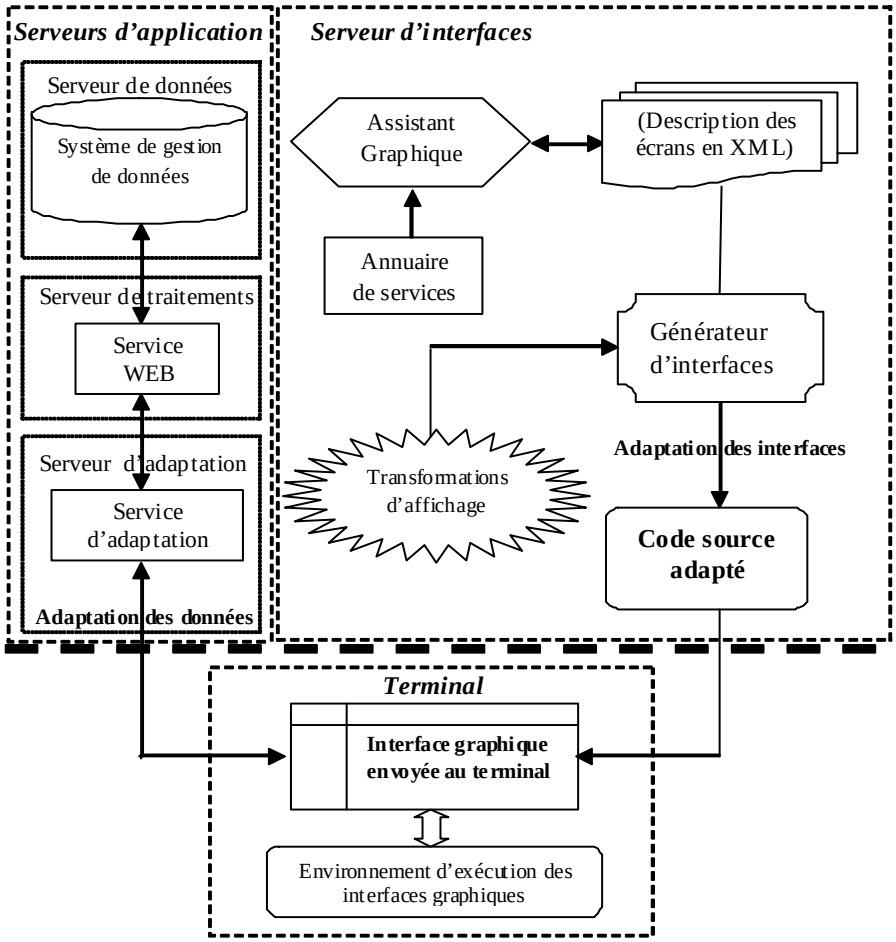


Figure 2. Architecture de SEFAGI

3.1.2. Terminal

Le terminal contient un environnement d'exécution qui héberge les interfaces générées et gère leur exécution. Cet environnement est composé des éléments suivants :

- Les interfaces générées qui sont téléchargées à partir du serveur d'interfaces.

- La bibliothèque de composants nécessaires à l'exécution des interfaces.

- Un gestionnaire de navigation assurant la navigation entre les différents écrans générés.

- Un gestionnaire de communication inter-écrans assurant l'interaction des écrans générés entre eux à travers une structure de données commune.

- Un gestionnaire d'invocation de services responsable de l'exécution des traitements distants et de la collecte de leurs paramètres d'entrée et de sortie.

3.1.3. Serveur d'applications

Notre serveur d'applications comporte trois types de serveurs :

- Serveur de données : ce serveur héberge des systèmes de gestion de bases de données et des systèmes de gestion de fichiers.

- Serveur de traitements : ce serveur héberge les services appelés par les interfaces générées en collectant des informations des serveurs de données et en effectuant des traitements éventuels sur ces informations pour les besoins de l'application. Par exemple un traitement effectue une analyse statistique sur des informations collectées à partir d'une base de données géographique.

- Serveur d'adaptation : ce serveur adapte les données issues des serveurs de traitements et des serveurs de données aux différents terminaux mobiles exécutant les interfaces graphiques générées. Il héberge des services d'adaptation qui prennent en entrée les caractéristiques du terminal connecté. Par exemple un serveur d'adaptation récupère des images à partir d'un serveur de données et adapte le format, la taille et la résolution au terminal demandant l'image.

3.2. Description générique des interfaces graphiques

Comme présenté dans l'état de l'art, les descriptions existantes des interfaces graphiques sont très complexes. On trouve presque tout le code de l'interface dans cette description. L'approche que nous avons développée vise à utiliser un modèle de description plus convivial en utilisant un niveau de granularité plus gros. Ceci facilite la tâche du concepteur d'interfaces puisqu'on donne à l'utilisateur un niveau d'abstraction assez loin du code qui va être généré.

En suivant l'objectif de modularité que nous avons fixé, on sépare l'affichage, les données et les traitements suivant le modèle MVC (Sanderson, 1999). Ainsi, le code d'une interface graphique $\mathbb{I}$ est composée d'une vue $\mathbb{V}$ (présentation graphique), un

modèle $\mathbb{M}$ (traitements et stockage de données) et un contrôleur $\mathbb{C}$ (gestionnaire d'évènements). Le contrôleur réalise la liaison entre le modèle et la vue lors du déclenchement d'un évènement dans l'interface.

$$\mathbb{I} = \mathbb{V} + \mathbb{M} + \mathbb{C}.$$

A chaque interface $\mathbb{I}$, on associe une description générique X qui va constituer l'entrée du générateur du serveur d'interfaces. $\mathbb{I}$ constitue la sortie du générateur. Cette description générique X d'une interface présente les composants constituant l'interface et leurs fonctionnalités attendues sans entrer dans les détails de leur disposition sur l'écran du terminal. En effet, cette tâche est laissée au moteur de génération qui va choisir la présentation de l'interface en fonction des caractéristiques des terminaux cibles.

Le granule de notre modèle est le *composant*. Un *composant* peut être défini comme l'entité graphique minimale qui peut engendrer une donnée utile pour le système d'information. Les composants peuvent être regroupés en panneaux. Un *panneau* traite un ensemble homogène de données pour réaliser une tâche spécifique (afficher le résultat d'une requête SQL, afficher des images…). Nous avons prédéfini un ensemble de types de panneaux génériques (tableau, image, graphique 2D, texte, vidéo, commande). Ainsi la description X d'une interface $\mathbb{I}$ est constituée de la description des panneaux qui la composent.

$X = \bigcup_{i=1}^{Np} Pi$, avec Np est le nombre de panneaux dans l'interface $\mathbb{I}$

Les composants peuvent interagir entre eux (passage de valeurs entre les composants), avec un service distant ou avec des variables internes à l'application. Une association entre les services et les composants doit être établie pour garantir l'interaction de l'interface graphique avec les différents traitements distants. Cette association est d'ordre n-n puisqu'une interface peut être liée à plusieurs services et inversement. A chaque panneau, nous associons un ou plusieurs services qui ont des identifiants connus par le serveur. Par convention, un composant ne peut engendrer qu'une seule donnée (qui peut être multi – valuée).

Le granule de description d'un service est un *paramètre* qui représente une donnée d'entrée ou de sortie du service. Un service peut être modélisé par une fonction $\mathcal{S}: De \rightarrow Ds$ qui prend en entrée un ensemble de données $De = \bigcup_{i=1}^{Nde} de_i$ et fournit en sortie un ensemble de données $Ds = \bigcup_{i=1}^{Nds} ds_i$, avec Nde le nombre des données d'entrée du service S et Nds le nombre des données de sortie du service S.

On distingue deux types d'association entre un service et un ensemble de composants $\mathcal{A} = \{C_{i,j}\}$ où $C_{i,j}$ est le jième composant du ième panneau de l'interface :

- Association contextuelle :

On parle d'une association contextuelle entre un service S et un ensemble de composants $A = \{C_{i,j}\}$ si tous ces composants constituent un même panneau Pi de l'interface :

$A = \bigcup_{j=1}^{Nc} Ci,j$, avec Nc le nombre de composants dans le panneau Pi.

Dans ce cas, l'ensemble des données de sortie Ds du service S sera affecté à l'ensemble des composants $C_{i,j}$ constituant l'affichage du panneau Pi. Ainsi Ds forme un ensemble homogène de données appartenant à un contexte bien déterminé (informations concernant un même patient, sorties d'un même algorithme de calcul). Du point de vue du concepteur de l'interface, le panneau va prendre l'ensemble de données Ds et va s'occuper de façon automatique (par une procédure interne) de l'affectation des différentes données. Ainsi on a une affectation globale $Ds \rightarrow Pi$ et le nombre de données de sortie du service est égal au nombre de composants constituant le panneau associé (Nds = Nc).

- Association non contextuelle :

On parle d'une association non contextuelle lorsqu'on n'a pas de conditions particulières sur l'ensemble des composants $A = \{C_{i,j}\}$. Les données de sortie du service considéré sont contextuellement indépendantes et les composants qui vont les recevoir peuvent appartenir à des panneaux différents. Ainsi on a une association individuelle $ds_k \rightarrow Ci,j$.

Une autre application établit un lien entre les données d'entrée De et quelques composants ou variables.

En conclusion, pour la description X d'une interface graphique 𝕀, un panneau est un ensemble de composants et un ensemble de services.

$X = \bigcup_{i=1}^{Np} Pi$ Avec $Pi = \bigcup_{j=1}^{Nc} Ci,j + \bigcup_{k=1}^{Ns} Si,k$, avec Ci,j le jième composant du panneau Pi et Si,k le kième service du panneau Pi. De ce qui précède découle le diagramme de classes UML donné en figure 3.

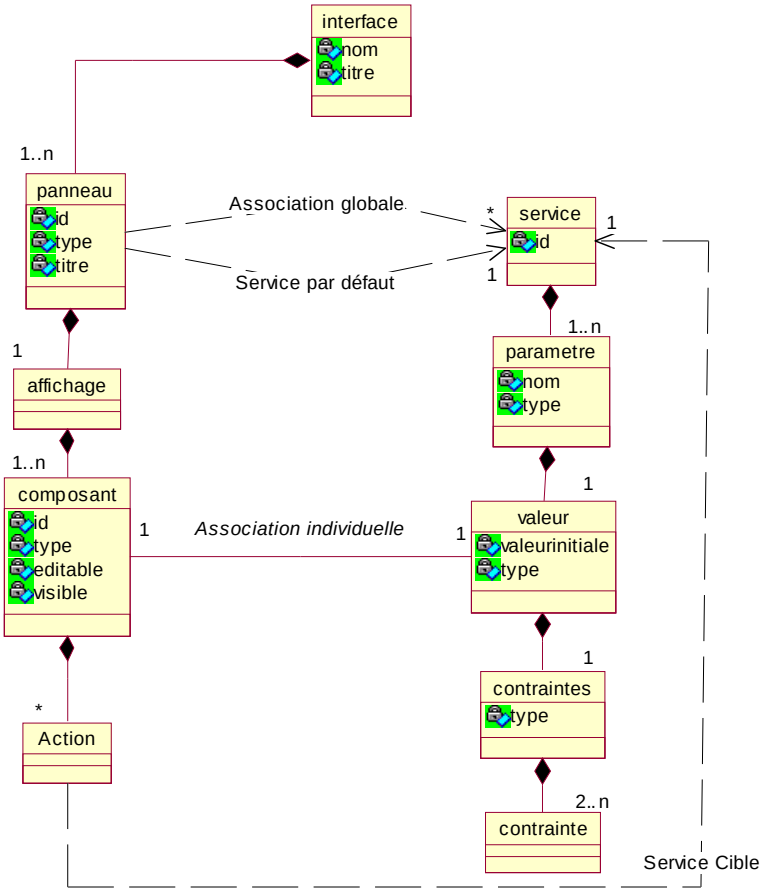


Figure 3. Diagramme de classes de la description générique d'une interface graphique

3.3. Génération adaptative des interfaces graphiques

La différence de taille des écrans entre terminaux standards et terminaux mobiles induit des transformations d'affichage et de disposition des composants différentes selon le contexte d'utilisation de l'interface. On distingue deux types de transformations : les transformations au cours de la génération et les transformations au cours de l'exécution des interfaces graphiques.

3.3.1. Transformations au cours de la génération

La génération des interfaces graphiques à partir du modèle générique peut être vue comme une fonction :

$\mathcal{G} : (X, L) \rightarrow I$ où :

- L est la librairie des composants graphiques du type du terminal cible. (A chaque type de terminaux correspond une librairie L).

- $\mathbb{I}$ est l'interface graphique résultante.

- X est la description de l'interface voulue.

Puisque les panneaux sont graphiquement indépendants,

$$\mathcal{G}(X,L) = \bigcup_{i=1}^{Np} \mathcal{G}(\mathcal{P}i, \mathcal{L}) = \bigcup_{i=1}^{Np} Pi_L$$

où Pi_L est l'implémentation du panneau Pi suivant la librairie des composants L.

$Pi_L = \mathcal{G}\left(\bigcup_{j=1}^{Nc} Ci, j, L\right) + \mathcal{G}\left(\bigcup_{k=1}^{Ns} Si, k\right)$ = Ai + Ti. Où :

- Ai est le code source correspondant à l'affichage du panneau Pi.

- Ti est le code correspondant aux services à invoquer dans le panneau Pi.

Ainsi, la vue de l'interface $\mathbb{I}$ s'écrit $\mathbb{V} = \bigcup_{i=1}^{Np} Ai$, et le modèle de l'interface $\mathbb{I}$ s'écrit $\mathbb{M} = \bigcup_{i=1}^{Np} Ti$.

Au cours de la génération, la fonction $\mathcal{G}$ effectue un certain nombre de transformations pour adapter les interfaces graphiques aux différents types de terminaux : transformations d'affichage, de disposition et de navigation.

3.3.1.1. Transformation d'affichage

La transformation d'affichage consiste à affecter à chaque composant $C_{i,j}$ de la description XML de l'interface son équivalent $C_{i,jL}$ dans la librairie L. Ce type de transformation est basé sur une table de correspondance où nous associons à chaque composant graphique du modèle générique son implémentation pour chaque type de terminaux. Les composants de deux librairies différentes peuvent présenter la même information différemment mais tout en gardant la fonctionnalité de base de chaque composant.

3.3.1.2. Transformation de disposition

La disposition ou la taille d'un composant est ajustée en fonction de la limitation des dimensions de l'écran. Par exemple, il est parfois nécessaire d'effectuer la division d'une fenêtre graphique sur plusieurs écrans logiques. Ce type de transformation est basé sur une convention d'affichage garantissant le confort d'utilisation et la valeur ergonomique de l'interface :

- Pour les écrans standard nous présentons les composants horizontalement.

- Pour les petits écrans, nous présentons les composants verticalement, ce qui permet une meilleure lisibilité des éléments de l'interface (et ce qui est d'ailleurs conforme aux « standards » utilisés).

On désigne par P_{LS} l'implémentation d'un panneau pour un terminal standard et P_{LA} l'implémentation d'un panneau pour un terminal ambulatoire.

$\mathcal{G}$ (P,L) = P_{LS} pour les écrans standard

= P_{LA} = $\bigcup_i Ei$ où E_i représente un écran logique sur un terminal ambulatoire

En tenant compte de la convention citée, on aura les **transformations de disposition** suivantes :

- Si le panneau contient un seul composant, la même disposition sera prise en compte pour tous les terminaux et le panneau occupera tout l'écran des petits appareils.

- Si le panneau contient plusieurs composants :

* Pour les écrans standards, les composants sont affichés horizontalement.

* Pour les petits écrans, les composants sont affichés verticalement dans un même écran ou sur plusieurs écrans successifs si les données engendrées par ces composants sont multivaluées.

- Si le panneau contient plusieurs instances d'un même composant (requête SQL renvoyant un résultat multilignes), le résultat sera affiché dans une table pour les écrans standards. Pour les terminaux mobiles, on commence par afficher la colonne la plus significative (on peut prendre la première par défaut) et ensuite l'affichage se fera ligne par ligne suivant la sélection de l'utilisateur.

Ainsi, pour les terminaux mobiles :

- P_{LA} = E si P = C (si le panneau est constitué d'un seul composant, on aura un seul écran)

- P_{LA} = E si les données associées aux composants de P sont monovaluées

- P_{LA} = {E_1, E_2,…} card($\mathcal{G}$ (P,L))>=2 (on aura au moins deux écrans) si les données associées aux composants de P sont multivaluées.

3.3.1.3. Transformation de navigation

Pour garantir une interaction complète de l'utilisateur avec les différents écrans constituant l'interface, une arborescence de navigation doit être implantée. Ceci est une conséquence de la division d'une interface graphique sur plusieurs écrans logiques pour les terminaux mobiles.

Quand l'utilisateur se connecte à l'environnement d'exécution, il dispose d'un menu principal permettant d'accéder aux différentes interfaces générées. Ensuite, il entre dans le menu secondaire qui présente les fonctionnalités offertes par l'interface choisie. Les éléments de ce menu correspondent généralement aux titres des panneaux. Un quatrième niveau peut exister si un panneau engendre plus qu'une fonctionnalité de base (le cas de l'affichage d'un panneau sur plusieurs écrans).

Après l'implémentation, cette arborescence est constituée d'une racine qui correspond à l'entrée de l'environnement d'exécution, ensuite au niveau 1, on trouve les interfaces graphiques générées $\mathcal{G}(\mathcal{X},\mathcal{L})$ enfin dans les deux derniers niveaux on trouve les implémentation des panneaux $\mathcal{G}(\mathcal{P},\mathcal{L})$. Pour les terminaux mobiles un panneau Pi,j de la description de l'interface peut correspondre à plusieurs écrans logiques lors de son implémentation. L'arborescence a donc la structure donnée en figure 4, selon un diagramme de classes UML.

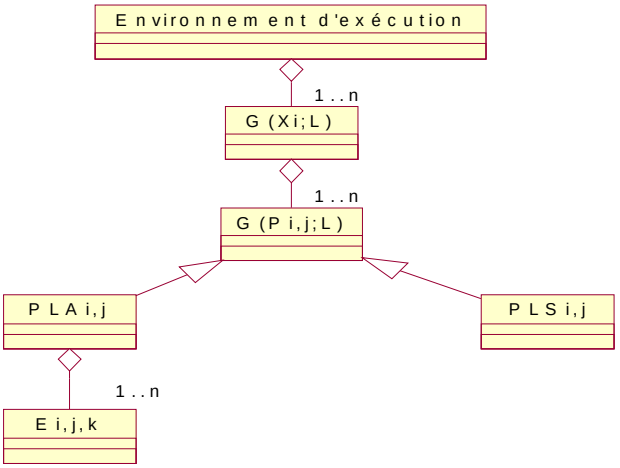


Figure 4. Arborescence de navigation

3.3.2. Transformations au cours de l'exécution

Après leur génération, les interfaces graphiques doivent interagir avec l'environnement d'exécution implanté au niveau du terminal pour former une application complète. Ces interfaces doivent recevoir et envoyer des données aux services. Cet échange induit une adaptation des données aux différents terminaux. On distingue trois types de transformations au cours de l'exécution : transformations de cohérence, de contenu et de transmission.

3.3.2.1. Transformation de cohérence

Lors de l'échange de données entre le terminal et les services WEB, ces derniers reçoivent un bloc de données *De* et renvoient un bloc de données *Ds*. Ces blocs de données sont atomiques lors de la transmission. Or, ces informations peuvent être présentées ou récupérées de plusieurs écrans logiques différents. Ainsi, des procédures de groupage et de séparations de données s'imposent. De plus, vu que les connexions des terminaux mobiles sont intermittentes, une gestion des points de reprise s'avère utile pour la sûreté de fonctionnement des interfaces générées.

- *Groupage* : Lors de la saisie, l'utilisateur peut introduire des informations sur plusieurs étapes. En outre, une donnée *de* du bloc *De* peut être récupérée d'une variable. Un gestionnaire de récupération de données avant l'invocation des traitements a été mis en place pour assurer la cohérence des entrées de chaque service.

- *Dissociation* : Lors de la récupération du bloc de sortie *Ds* d'un service distant, les données *ds* constituant ce bloc peuvent être présentées sur plusieurs écrans ou elles peuvent être stockées dans des variables internes à l'application. Le gestionnaire d'affectation de données après l'invocation des services est responsable de ce type de transformations.

- *Reprise* : Pour éviter les problèmes de pertes de données suite à des coupures de transmissions, une gestion des points de reprises s'avère indispensable. Un mécanisme de sessions sauvegarde les points de reprises de chaque client au niveau des serveurs d'applications.

3.3.2.2. Transformation de contenu

Les données envoyées aux interfaces générées doivent être compatibles avec les limites matérielles et logicielles du terminal cible. Le serveur d'adaptation effectue un certain nombre de transformations pour assurer la présentation adéquate de ces données pour le client :

- *Modification* : Cette transformation consiste à modifier la taille, le format ou la structure d'une donnée pour qu'elle puisse être présentée au niveau du terminal. Par exemple, les terminaux mobiles ne prennent pas en charge les formats connus d'images (JPEG, GIF, TIFF, BMP…). Une adaptation en format PNG (Portable Network Graphics) compréhensible par ces terminaux est donc nécessaire

- *Substitution* : Cette transformation consiste à remplacer une donnée par une autre qui porte la même sémantique que la première. Ce type d'adaptation est dû à l'impossibilité de présentation d'une donnée sur le terminal. Par exemple, pour un terminal qui ne peut pas afficher de vidéos, on peut envoyer des descriptions textuelles des informations pertinentes de la séquence.

Ces transformations doivent conserver la valeur sémantique initiale de la donnée. Si la modification engendre une dégradation considérable de la ressource, on préfèrera utiliser une donnée de substitution. Par exemple si une modification de la résolution d'une image médicale engendre une perte considérable de lisibilité, il vaut mieux la remplacer par une description textuelle de l'interprétation de l'image. L'adaptation du contenu peut être vue comme une relation entre une ressource initiale $\mathcal{R}$ et son équivalent $\mathcal{R}'$ qu'on veut envoyer vers un terminal cible.

On a vu dans ce paragraphe deux relations principales : « Modifiée en » et « Substituée par ». Il est important de trouver une ressource équivalente $\mathcal{R}'$ qui garde la sémantique de la première ressource. On doit toujours trouver une alternative pour ne pas omettre une information qui peut être pertinente à l'utilisateur.

3.3.2.3. Transformation de transmission

Nous avons opté pour XMLRPC comme protocole de transmission entre les terminaux et les services WEB vu sa légèreté et son adéquation avec les appareils mobiles. Le format d'échange d'informations est donc XML. Ceci implique une première transformation consistant à convertir les données dans leur format d'origine au niveau du terminal en XML et la deuxième transforme les données XML reçues par le serveur en langage compréhensible par le service et réciproquement.

3.4. Synthèse

Après avoir défini un langage de description d'interfaces graphiques de haut niveau, simple et convivial, nous avons caractérisé les transformations nécessaires lors de la génération et lors de l'exécution de ces interfaces. Ces transformations touchent les données, les vues (présentations) et les traitements (services). La figure 6 donne un tableau récapitulatif de ces transformations.

Transformations au cours de la génération

Transformations de l'affichage (présentation)	- Correspondance entre les composants du modèle et ceux des langages cibles
Transformations de disposition (présentation)	- Disposition des composants dans les panneaux - Disposition des panneaux sur les écrans
Transformations de navigation (présentation)	- Facilités de navigation pour l'utilisateur

Transformations au cours de l'exécution

Transformations de cohérence (données)	- Groupage ou dissociation de données. - Gestion des points de reprise
Transformations de contenu (données)	- Modification ou substitution d'une ressource par une autre plus adaptée
Transformations de transmission (traitements)	- Protocole de transmission standard - Typage standard des données

Figure 5. Transformations pour l'adaptation des interfaces graphiques

Ces transformations nous permettent de fournir des interfaces graphiques complètes présentant une haute intéractivité avec l'utilisateur et avec les traitements distants. Ainsi nous avons opté pour une adaptation statique automatique des interfaces graphiques (génération adaptative automatique) et une adaptation dynamique des données (application des transformations de données au cours de l'exécution). Toutes ces considérations ont contribué à la réalisation d'un prototype.

3.5. Implantation

Nous avons mis en œuvre un prototype pour valider notre approche. Il comporte un assistant graphique qui aide l'utilisateur à créer la description XML de l'interface qu'il veut générer. Cet outil procède par étapes en posant un certain nombre de questions à l'utilisateur et en automatisant le plus grand nombre de tâches possibles. L'assistant interagit avec un annuaire de services pour que l'utilisateur puisse effectuer l'association entre les entrées / sorties des services et les composants de l'interface graphique qui est en cours de création.

Pour résoudre le problème de diversité des terminaux mobiles, nous avons utilisé une API commune à tous les appareils. L'API retenue est J2ME vu sa popularité et sa disponibilité sur la plupart des mobiles du marché. Cette API présente des composants qui tiennent compte de la limitation de la puissance de calcul et de la quantité de mémoire disponible. Le prototype que nous avons mis en œuvre permet de générer à partir d'une même description XML d'une interface graphique deux versions adaptées de cette interface : une version J2SE pour les terminaux standards et une version J2ME pour les terminaux mobiles.

L'environnement de génération est un outil graphique permettant de générer le code source de la vue, du modèle et du contrôleur d'une interface à partir de sa description XML, conformément au modèle donné en figure 3. Cet environnement

effectue toutes les transformations spécifiées dans le paragraphe 3.3.1 pour adapter l'affichage au terminal cible.

Avant de pouvoir exécuter les interfaces générées, le terminal doit télécharger un environnement d'exécution comportant la bibliothèque de composants graphiques nécessaires à l'exécution des interfaces, un programme principal fournissant des facilités de navigation entre les différentes interfaces, un gestionnaire de communication inter-écrans, un adaptateur de transmission qui assure le formatage des données échangées en XML et un invocateur de services pour assurer l'interaction de l'interface avec les services WEB existants. Les invocations de service incluent toutes un paramètre indiquant le type du terminal qui a fait la demande. C'est avec ce paramètre que le serveur d'adaptation décide des adaptations dynamiques à effectuer.

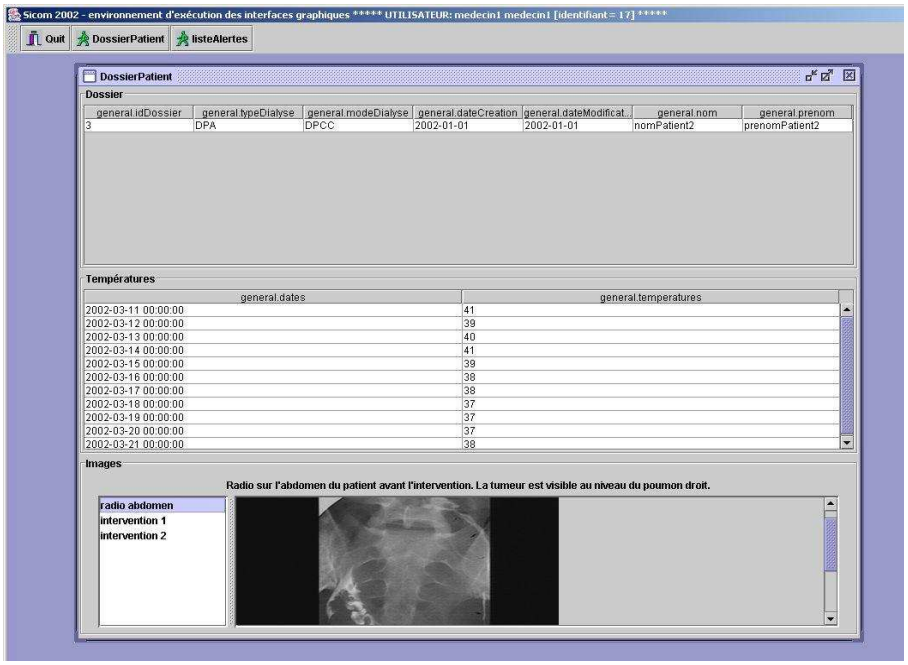


Figure 6. *Interface graphique générée pour les terminaux standards*

La figure 6 présente une fenêtre générée pour les terminaux standards dans l'environnement d'exécution, c'est-à-dire comme elle est vue et manipulée par un utilisateur final. Une barre de menu permet la navigation entre les différentes interfaces générées (Dossier Patient et listeAlertes). La fenêtre DossierPatient est affichée. Elle comporte 3 panneaux. Les deux premiers sont de type tableau, le troisième est de type image. Les panneaux de type tableau contiennent une ligne par n-uplet de données à afficher/saisir. Chaque élément de la ligne est ici une zone de

texte simple, mais on peut trouver des listes de choix, des cases à cocher… Le panneau de type image contient trois zones : la première fournit la liste des noms des images, la seconde présente l'image sélectionnée, et la troisième (au-dessus) affiche le texte descriptif associé à l'image.

La figure 7 donne l'équivalent pour les terminaux mobiles. La barre de menu est remplacée par un premier écran fournissant la liste des fenêtres disponibles. Le système présente ensuite le premier panneau de l'interface choisie par l'utilisateur. Le menu contextuel permet alors de passer à un autre panneau de cette fenêtre, ou de retourner au menu principal.

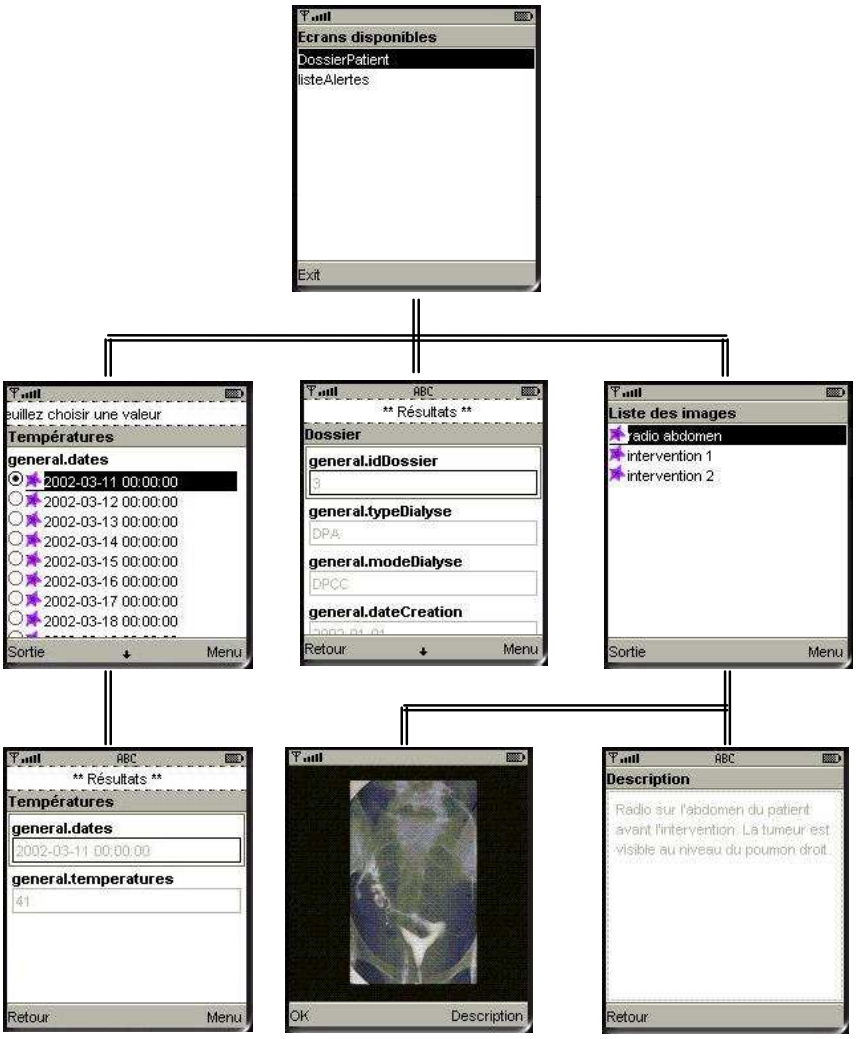


Figure 7. Interface graphique générée pour les terminaux mobiles

Le premier panneau (résultat monoligne dans un panneau tableau) est constitué d'un seul écran sur le mobile, les éléments sont affichés de façon verticale (ils sont affichés horizontalement sur un écran standard). Pour le panneau tableau multi-lignes, on accède premièrement à une liste correspondant aux valeurs de la première colonne, qui permet de sélectionner l'affichage de la ligne correspondante dans un autre écran (affichage vertical des informations). Pour le panneau de type image, un premier écran est proposé avec la liste des images. La sélection d'une image permet de passer à un écran où l'image elle-même est présentée (elle est entre temps passée par une adaptation de contenu pour être transformée au format PNG). Le menu « description » permet ensuite de passer à la description textuelle de l'image. Dans le cas où le terminal utilisé ne supporte pas l'affichage d'images, la sélection d'une image amène directement à la description textuelle correspondante.

Pour la création de cette interface, nous avons utilisé l'assistant graphique de SEFAGI. Nous avons d'abord donné le nom et le titre de l'interface (figure 8).

Figure 8. Environnement de génération - création d'une nouvelle interface

Ensuite, pour l'ajout de chaque panneau, il suffit de spécifier son type parmi les types déjà prédéfinis (figure 9) puis de lui affecter un service parmi la liste des services existants (figure 10).

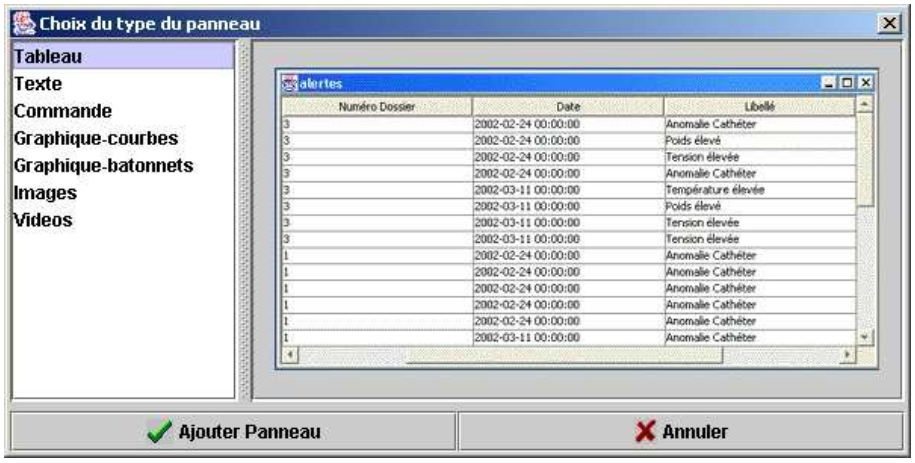


Figure 9. Choix du type de panneaux

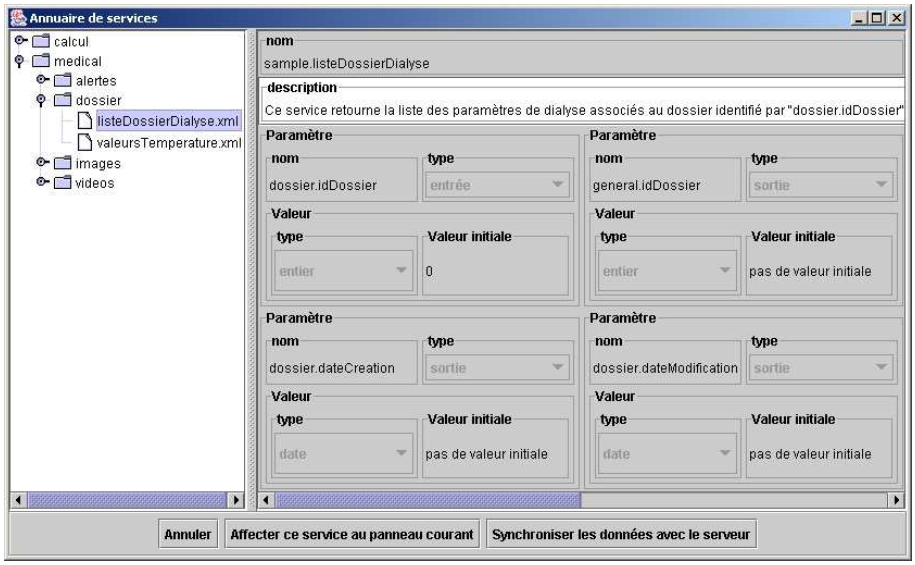


Figure 10. Choix du service à invoquer

L’assistant détecte automatiquement les composants graphiques nécessaires pour l’interaction avec le service (affectation des paramètres d’entrée et de sortie du service aux composants graphiques de l’interface). Cependant, l’utilisateur peut modifier le choix des composants et le mode d’interaction avec le service (d’où le service prend ses entrées : variables internes à l’application ou directement à partir d’un composant) (figure 11).

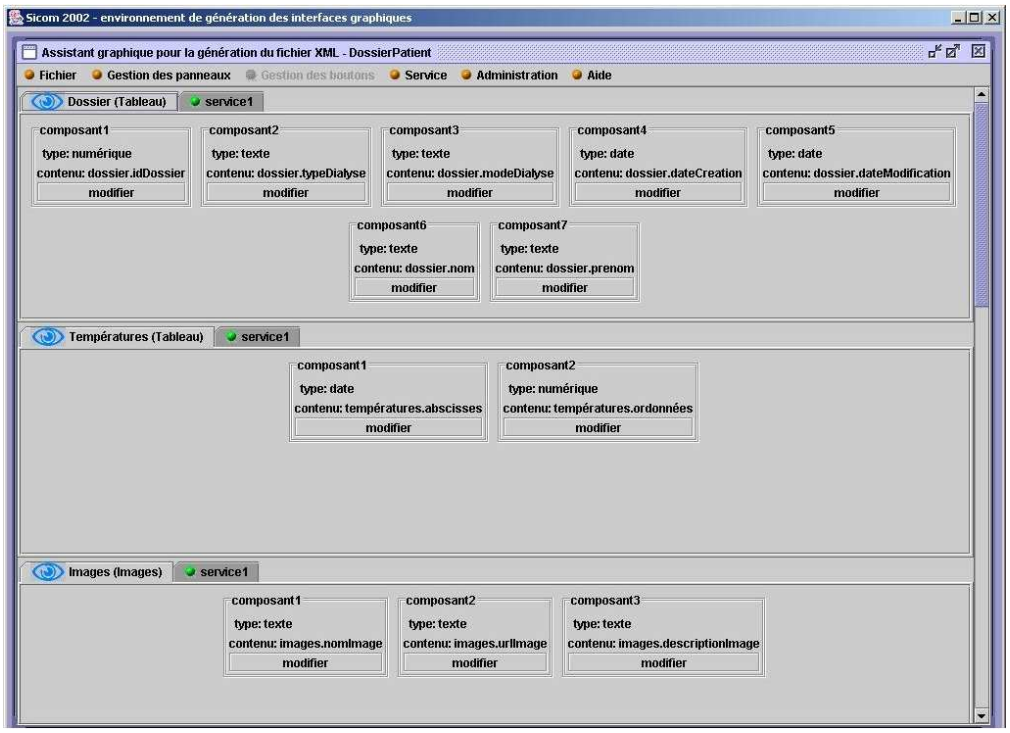


Figure 11. Création de l'interface "DossierPatient" par l'assistant de SEFAGI

L’assistant de SEFAGI sauvegarde les actions effectuées par l’utilisateur sous format XML. Ainsi, l’utilisateur peut éditer ou continuer la création d’une interface enregistrée sur le disque. Ensuite, à partir du fichier XML résultant, l’assistant génère le code source correspondant pour la catégorie des terminaux voulue. Dans notre prototype nous avons défini deux catégories : terminaux J2ME (terminaux mobiles supportant java) et terminaux J2SE (terminaux standards supportant java) (figure 8).

4. Conclusion et perspectives

Le projet SEFAGI (Simple Environment For Adaptable Graphical Interfaces) donne une solution simple et conviviale pour la génération automatique et l'adaptation des interfaces graphiques multi-terminaux. Il adresse les problèmes d'hétérogénéité et de limitation matérielle des terminaux mobiles. Nous avons opté pour la *modularité* en séparant les données, les traitements et les présentations. Notre deuxième facteur clé est l'*adaptation* qui touche les trois facettes d'une application. Ceci augmente le degré d'adaptabilité des applications à plusieurs terminaux en permettant d'exploiter les mêmes données pour plusieurs interfaces graphiques et vice versa. Notre proposition présente aussi l'avantage de partir d'un langage de description générique simple et suffisant pour décrire la majorité des fonctionnalités attendues d'une interface graphique.

Cette étude nous a conduit à mettre en œuvre un prototype validant toute l'architecture du projet SEFAGI (figure 2). Ce prototype est entièrement fonctionnel. Il génère complètement les interfaces graphiques ; ces interfaces interagissent avec des traitements distants divers (calculs mathématiques, mise à jour d'une base de données, téléchargement d'une image…). A partir d'un assistant graphique, l'utilisateur choisit les services et les composants graphiques qu'il veut utiliser. Ensuite le code source de ces interfaces est généré d'une façon automatique et adaptée à l'appareil cible. Finalement, sur le terminal, ces interfaces interagissent avec les services distants spécifiés. Des services adaptent les données à envoyer au terminal. Ainsi, le client dispose d'une application complète et fonctionnelle sur son appareil.

La sécurité et la confidentialité des données sont des facteurs importants dans les systèmes d'information. Les fonctionnalités offertes par une interface graphique se limitent à saisir et présenter des données. Pour intégrer la sécurité dans notre architecture, on pourra dédier des services WEB au filtrage des données et à la vérification des droits d'accès.

Ce projet présente une architecture complète d'adaptation des applications pour les terminaux mobiles. Parmi les perspectives tracées pour ce projet, nous comptons établir une architecture qui aide le concepteur à concevoir et générer l'ensemble des services élémentaires qui vont constituer le modèle de traitements de son application. Ces traitements peuvent être vus comme une composition d'un ensemble de services qui interagissent entre eux. Une architecture à base de composants peut être utile à cette problématique. Dans cette perspective, nous avons développé des services génériques (génération automatique de requêtes SQL à partir de l'assistant, service de cryptage et de sécurité, service d'adaptation d'images…) pour couvrir la majorité des typologies de traitements possibles.

Bibliographie

Abrams et al., 1999 M. Abrams, C. Phanouriou, “UIML: an XML language for building device independent user interfaces”, *XML’99*, Philadelphia, December 1999.

Benjaminsen *et al*., 2002 S. Benjaminsen, A. Djora « JetRek: How organisational identities slowed down speedy requisitions », *BSA medical sociology conference*, september 2002, York. http:// www.sv.ntnu.no/iss/Aksel.Tjora/publications/ Siri-york02-09.pdf

Birnbaum, 1997 J. Birnbaum, Pervasive Information Systems, *Communications of the ACM 40 no 2*, february 1997.

Boshernitsan,2001 M. Boshernitsan, “Harmonia: A flexible framework for constructing interactive language-based programming tools”, Technical Report, University of California, Berkeley, June 2001

Calvary *et al*., 2001 G. Calvary, J. Coutaz and D. Thevenin “Supporting context changes for plastic user interfaces: a process and a mechanism”, *Proceedings of HCI-IHM 2001, BCS conference series*, Springer Publ., pp. 349-363

Calvary *et al*., 2002 G. Calvary et J. Coutaz “Plasticité des interfaces : une nécessité !”, information-interaction-intelligence, *Actes des deuxièmes Assises nationales du GDR I3*, Cépaduès Editions, Nancy pp 247-261

Kamada, 1998 T. Kamada, “Compact HTML for small appliances”, *World Wide Web Consortium*, february 1998.

Larson, 2002 Eric D. Larson, Wireless Java Application Saves Women's Cancer Center $500,000 per Year, J2ME case studies, september 2002. http://wireless.java.sun.com/midp/casestudies/wcc/.

Layaïda, 1999 N. Layaïda, Adaptabilité : pistes d'étude pour la définition d'une infrastructure d'accès au contenu multimédia pour des machines hétérogènes, Rapport Technique, INRIA Rhône-Alpes, octobre 1999.

Lemlouma *et al*., 2001 T. Lemlouma, N. Layaïda, « A Framework for Media Resource Manipulation in an Adaptation and Negotiation Architecture », OPERA project, INRIA Rhône-Alpes, august 2001

Massé, 2002 V. Massé, « La société MobilePlanet Europe fournit des terminaux mobiles à la Brigade des Sapeurs Pompiers de Paris (BSPP) pour l’équipement de ses véhicules d’intervention », Mobile Planet, juin 2002. http://www.mobileplanet.fr/m_includes/press_release/2002_brigade.asp

Menkhaus, 2002 G. Menkhaus « Adaptive User Interface Generation in a Mobile Computing Environment », PhD Thesis, Salzburg University, 2002.

Mori, 2000 Y. Mori, “IMODE phase II”, *ITU Telecom, ASIA 2000 special report*, November 2000.

Muchow, 2001 J. W. Muchow “*the basics of J2ME*” In “*Core J2ME Technology & MIDP*”, Pearson Education, december 2001.

Narayanaswamy, 2002 A. Narayanaswamy, “Introducing Microsoft .net” , *Microsoft Press*. 2002.

Nigay, 1995 L. Nigay "MATIS : un système multimodal d'information sur les transports aériens", *IHM'95, 7èmes journées de l'ingénierie de l'interaction Homme Machine*, CEPAD Publ., pp. 131-132

Piroumian, 2002 V. Piroumian "*Introduction to the Java 2 Micro Edition Platform*" In "*Wireless Java 2 ME Platform Programming*", Pearson Education, march 2002.

Riggs *et al*., 2001 R. Riggs, A. Taivalssari, M. Vandenbrink, *"MIDP Application Model"* In *"Programming Wireless Devices with the Java 2 Platform*", Addison-Wesley, June 2001. http://wireless.java.sun.com/midp/chapters/pwdevices/ch08.pdf

Sanderson, 1999 R. Sanderson , "MVC-Client: Putting Model-View-Controller to work", White Paper, *Fourbit Group*, 1999.

Satyanarayanan, 2001 M. Satyanarayanan, Pervasive Computing: Vision and challenge, *IEEE Communications*, 2001.

TeraCom, 2002 TeraCom "Soluphone santé" 2002. http://www.soluphone.com/SoluSante.pdf

Thevenin *et al*., 2000 D. Thevenin G. Calvary et J. Coutaz "La multimodalité en plasticité", Actes du c*olloque sur la multimodalité*, Grenoble, pp. 55-58

W3C, 1998 World Wide Web Consortium, « extensible Markup Language 1.0 », february 1998 http://www.w3.org/TR/1998/REC-xml-19980210

W3C, 1999 World Wide Web Consortium, « XMLRPC Specifications », june 1999 http://www.xmlrpc.com/spec

W3C, 2000 World Wide Web Consortium, « Simple Object Access Protocol (SOAP) 1.1 », may 2001 http://www.w3.org/TR/soap

W3C, 2001 World Wide Web Consortium, « Web Services Description Language (WSDL) 1.1 », march 2001 http://www.w3.org/TR/wsdl

W3C, 2002 World Wide Web Consortium, « Web services architecture requirements », october 2002. http://www.w3.org/TR/2002/WD-wsa-reqs-20021011

WAP Forum, 1999a Wireless Application Protocol Forum, "WAP, WML: Wireless Application Protocol, Wireless Markup Language Specification", version 1.2, november 1999.

WAP Forum, 1999b Wireless Application Protocol Forum, "WMLScript specification: WMLScript language specification", version 1.1, november 1999.

Yao, 2002 P. Yao, « Microsoft .net Compact Framework for windows CE » *Microsoft Press*. 2002.

[3]
Tarak Chaari, Dejene Ejigu, Frédérique Laforest, and Vasile-Marian Scuturici. A comprehensive approach to model and use context for adapting applications in pervasive environments. *Int. Journal on Systems and Software*, 3, 2007.

A Comprehensive Approach to Model and Use Context for Adapting Applications in Pervasive Environments

Tarak Chaari, Dejene Ejigu, Frédérique Laforest and Vasile-Marian Scuturici
LIRIS-CNRS-UMR 5205, INSA de Lyon, France
{tarak.chaari, dejene.ejigu, frederique.laforest, marian.scuturici}@insa-lyon.fr

Abstract

With an increasing diversity of pervasive computing devices integrated in our surroundings and an increasing mobility of users, it will be important for computer systems and applications to be context-aware. Lots of works have already been done in this direction on how to capture context data and how to carry it to the application. Among the remaining challenges are to create the intelligence to analyze the context information and deduce the meaning out of it, and to integrate it into adaptable applications. Our work focuses on these challenges by defining generic context storage and processing model and by studying its impact on the application core. We propose a reusable context ontology model that is based on two levels: a generic level and a domain specific level. We propose a generic adaptation framework to guarantee adaptation of applications to the context in a pervasive computing environment. We also introduce a comprehensive adaptation approach that involves content adaptation and presentation adaptation inline with the adaptation of the core services of applications. Our case study shows that the context model and the application adaptation strategies provide promising service architecture.

Keywords: Context-aware computing, context modeling, adaptation, web services, pervasive computing.

1. INTRODUCTION

New challenges have appeared in the current information systems. Users want to have the needed information at any time and everywhere. These challenges have incited developers to integrate mobile devices in their applications creating a new research domain called pervasive (or ubiquitous) computing [1]. Such applications have to adapt to a set of new parameters including the type of the device, the connection state, the user environment, etc. These parameters characterize a contextual situation.

In different context situations, users may access different data and exploit different aspects of an application. For example, in one context a doctor accesses a medical database for screening patients for prevention care, while in a different context the same doctor accesses the same database for post-treatment analysis of cases. While data are the same, the way they are returned may vary according to the doctor's goal. Often, in different contexts, users access almost the same data and the same services but receive answers shaped differently, with different presentation and possibly different content detail. For example, a doctor examines a patient record at the hospital using a desktop computer connected to the hospital database, or consults the same record stored on a PDA while visiting the patient at home, or receives an audio description of the patient record during a surgical operation.

Context-aware applications have to perceive the situation of the user and his environment and consequently adapt all their behavior to that situation without an explicit demand from the user. A clear architecture and a well founded explicit relationship between the context as a key element and the adaptation strategy in context-aware pervasive computing is critical [2].

The realization of context-aware adaptation of applications requires studying many issues: context capturing, context modeling and interpreting, context dissemination and application adaptation to context. Many research efforts were made to address these issues but an extensible and comprehensive model is yet missing. Moreover, the question "how to adapt the application to the captured context?" remains without a precise and complete answer. In this paper we try to address these two problems by proposing a reusable and flexible context management model and an adaptation strategy of applications to different context situations. Our context management model is based on the semantic representation of context data, rules and concept ontology. It aims to support concept reasoning so as to provide context-aware services. Our adaptation approach concerns the available services of the application, the user interface that ensures the interaction with them and the content they provide to the user.

The paper is organized as follows: after reviewing the state of the art in context modeling and context-aware adaptation in section 2, we present our context model in section 3. Our context-aware architecture is presented in section 4. Then, in section 5, we detail our adaptation strategy to context. In section 6, we present how we have implemented our adaptation principles and how we have validated them using a case study in the medical field. Conclusions and highlights of future work are given in section 7.

2. STATE OF THE ART

Various areas of computer science have been investigating the concept of context over the last decades. It has been a key concept in the fields of human-computer interaction, expert systems, artificial intelligence, etc. Pervasive computing is a new domain in which context is receiving growing attention. Currently, pervasive computing devices are characterized by a shift from

general-purpose computing to task-specific support. Among these devices are PDAs, cell phones, wearable computing devices, etc. For long time, systems like Geographic Positioning System (GPS) and Geographic Information System (GIS) remained the sole source of context for the development of location-aware systems. However, location is only one aspect of a more complex physical and logical context. With the current advances in sensor technology and computer applications, context awareness can be attained through integrated effort in the development of an open and global framework.

T. Strang et al. [3] present a survey of six context modeling approaches: key-value modeling, markup scheme modeling, graphical modeling (UML, ER, etc), object oriented modeling, logic based modeling and ontology based modeling. According to their analysis that is based on the appropriateness requirements presented in the same article, they found that ontology based modeling is the most promising approach for context modeling in pervasive computing environments.

K. Henricksen et al. [4] investigate the UML and ER modeling approach and come up with their shortcomings for robust representation of context and all their properties. They introduced a new object based graphical model for their particular use.

X. H. Wang [5] present CONON, a context Ontology that is based on OWL for reasoning and representation of contexts in pervasive environments. Their approach is based on the treatment of high-level implicit contexts that are derived from low-level explicit contexts.

A. Held et al. [6] make an investigation of existing systems for comprehensive context representations. They proposed a comprehensive structured context profiles (CSCP) that is based on resource description framework (RDF). It lacks standardization and is developed as a proprietary model for specific domains.

H. Chen et al. [7] present Context Broker Architecture (CoBrA) to support context aware systems in smart space. They use their COBRA-ONT ontology for representations and OWL for describing places, agents, events and their associated properties.

P. Dockhorn et al. [8] identify that works on the context management in the literature are classified into four approaches: conceptual framework, service platforms, appliance environments and computing environments.

Conceptual frameworks focus on the architectural aspect of context-aware systems and provide means to facilitate capturing, interpreting and carrying context data to the interested parties. The Context Toolkit [9] and the Cooltown [10] projects are examples of this approach. *Service platforms* aim at providing the pertinent services to the user depending on context. This includes dynamic service discovery, dynamic deployment of adaptive services addressing issues of scalability, security and privacy. M3 [11] and Platform for Adaptive Applications [12] are examples of contributions to this approach. *Appliance environments* try giving solutions to the heterogeneity problem by providing interoperability techniques and frameworks. Ektara [13] and Universal Information Appliance [14] are projects which use this approach. *Computing environments* for pervasive applications focus on designing the physical and logical infrastructure to hold ubiquitous systems. The PIMA [15] project is an example of this approach.

Despite such a rich landscape in context related research, a complete and comprehensive model is still missing. Among the reasons is the lack of a standard and reusable reference model that can be used to handle context in multiple domains of application. Moreover, defining how the

application can adapt to the context is still a question that has no definite answer. In the following sections we shall present and discuss our solutions.

3. MODELING CONTEXT

Context and context-awareness are some of the core components of reasoning systems in the upcoming pervasive computing world so as to intelligently support user tasks by acting autonomously on behalf of users. The behavior of context-aware applications depends not only on their internal state and user interactions but also on the context sensed during their execution. Some early models of context information already exist, however basic issues related to context information modeling are still not fully addressed as existing context models vary in the types of context information they can represent. While some models take the user's current situation, others model the physical environment. The challenge is to put in place a more generic approach to context modeling in order to capture various features of context information including a variety of types of context information, dependencies among context information, quality of context information and context histories.

3.1 Basics of Our Context Modeling Principles

Definitions given by earlier works and standard dictionaries agree on the key idea that contexts describe situations. Anind K. Dey et al [16] have also confirmed this by defining context as: "*Any information that can be used to characterize the situation of an entity. An entity is a person, a place, or a physical or computational object that is considered relevant to the interaction between a user and an application, including the user and application themselves."*

T. Winograd [17] indicated that definitions like this are intended to be adequately general to cover the work that has been done on context-based interactions. He however argues that in using open-ended phrases such as "any information" and "characterize", it becomes so broad that context covers everything without boundary. According to Winograd, "*something is context because of the way it is used in interpretation, not due to its inherent properties. The voltage on the power lines is a context if there is some action by the user and/or computer whose interpretation is dependant on it, but otherwise is just part of the environment."*

We will focus on Winogard's approach whereby context is considered as an operational term whose definition depends on the interpretation of the operations involved on an entity at a particular time and space rather than the inherent characteristics of the entity itself. We classify basic context descriptors in a computing environment as: user context, device context, network context, activity context, service context, location context, and resource context. Figure 1 shows basic context descriptors and few examples of potential subclasses.

Such a listing can by no means be a complete list and therefore we need to have a scalable model to accommodate additions of new components. Using these descriptors and the pervasive environment paradigm, we redefine context as: *"Situations in each class of descriptors and their interaction with one another that affect actions taken by or actions accepted by computing entities, where classes of descriptors are users, devices, networks, activities, services, locations, resources …"*

The primary characteristic of a context is that it possesses an owner (person, device, place, etc.) For our subsequent discussion, we name this characteristic of a context as a subject. The situation of a subject, or the universe that surrounds it, is expressed in terms of multiple properties and their values. In our subsequent discussion, we use the terms predicate and value to represent the situation

of the subject with respect to a specific property. We prefer to use the term predicate in place of property because predicate has an important significance in describing situations in the predicate logic and language.

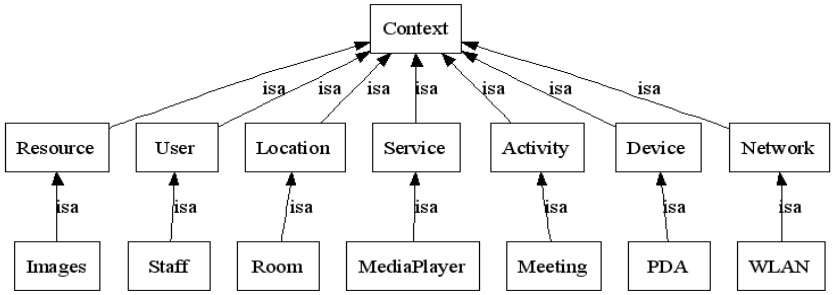


Figure 1- Basic context descriptors

Another important characteristic of context is that it is temporal. Just like humans learn their environment from experience, context-aware systems should keep track of past activities as history so as to learn from them. We can also consider quality of context information as one parameter for modeling context. Quality can be expressed as the *certainty* of occurrence of the context represented by a numeric value between 0 and 1.

Putting all our context parameters together, we get the general representation of context as *(Subject, Predicate, Value, Time, Certainty).*

When context timing is necessary to keep, we can have a time stamp associated and stored with the context data. Similarly, accuracy (certainty) as a measure of quality, if exists, can also be attached to each context data element. Predicate based expressions like hasTime() and hasAccuracy() can be used for association of these parameters with the core context value. Therefore, for most part of our context representation in this work, we use the three basic parameters *(Subject, Predicate, Value)* which can equivalently be represented in the form of *predicate(Subject, Value).*

3.2 The Need for a Semantically Rich Context Model

We will now study the characteristics of context data using a scenario where devices pda-01, pda-02 and laptop-01 are used to run different application services. This example demonstrates the need for a semantically rich descriptive context model to provide shared understanding.

In the following tabular representation, pda-01 exists as a subject in one entity and as a value in another entity suggesting the need for further treatment of such cross-attribute relationships and the need for the semantic description of attributes, entities and their relationships.

Subject	Predicate	Value
VideoService	runsOn	PDA-01
PDA-02	runs	VideoService
RealPlayer	runsOn	Laptop-01
ImageService	runsOn	Laptop-02
…	…	…

Handling user query is another area where semantic knowledge is necessary. For example, retrieving devices that are currently running *VideoService* from this table using standard database query is not straight forward. The query *[select value from context.device where subject = "VideoService"]*, gives only "pda-01" as an output. But in reality, if this information is given to a human assistant who knows, by common sense, that *RealPlayer*s give an Audio/Video Service and also knows that *runs* is an inverse function of *runsOn*, he will respond pda-01, pda-02 and laptop-01 to the query. So for processing context data, we need systems that are more intelligent and that are capable of processing not only contents but also the meanings (semantics) of data.

This, therefore, calls for the use of a context representation model that describes concepts, concept hierarchies and their relationships. Ontology is a structure that represents relevant entities, their relationships and related rules. OWL [18], a web ontology language, is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content is presented to humans as demonstrated in the above query.

In addition to the basic subject, predicate, value triples; context data possesses other information like time, accuracy, source, etc. This can be represented by making statement about a statement that gives meta-information about the context. This information can be embedded into the context ontology using the reification process. The following is an example that shows how reification is used to represent context meta-information. Assume that initial context sensed is: (hasSpeed, NetworkConnection, PP). Its meta-information could be: it is sensed at time tt, it is reported with accuracy aa and it is reported by sensor ss. Reified form of this context data is then given as follows:

```
<ns:xx rdf:type resource=rdf:statement/>
    <ns:xx rdf:subject resource=ns:NetworkConnection/>
    <ns:xx rdf:predicate resource=ns:hasSpeed/>
    <ns:xx rdf:object resource=ns:pp/>
<ns:xx ns:sensedAtTime  resource=ns:tt/>
<ns:xx ns:withAccuracy resource=ns:aa/>
<ns:xx ns:sensedBy resource=ns:ss/>
```

3.3 Using Ontology for Context Modeling

In this section, we present our ontological approach for context modeling so as to enrich our proposed *(Subject, Predicate, Value, Time, Certainty)* representation of context data. As shown in Figure 2, the captured low level *static and dynamic contexts* are aggregated by a *context interpreter* to give another set of deduced high level contexts (for example, coordinate based location values are converted to street names). Instances of contexts about a subject and its environment are mapped to context ontology structure by the *context manager*. This mapping helps to make the context ready for reasoning and decisions using available relations, concept ontology and rules. The *Domain Specific Rule Capture* module is responsible to integrate domain specific rules into the ontology structure. It also supplies rules to the *context-aware service module* and stores and retrieves rules to/from the *domain specific rule database*. The three sources of data context, ontology and rules are used by the Context-Aware Service module to perform reasoning and decisions. Decisions by the *Context-Aware service module* may be used to trigger actions like delivering contents, adapting applications, running applications, etc.

The use of ontology will make our model independent of programming and application environment. In addition to the standardization of the structure of the context representation, it will help us provide semantic descriptions and relationships of entities. Using ontology, we can also perform deeper knowledge analysis by defining domain specific rules.

Our generic context ontology (GCO) model consists of the following two basic components; the base ontology and the domain ontology. The base ontology part is defined based on our context descriptors while the domain ontology part is dependent on domain specific sub descriptors. Given:

B (basic context descriptors) = {User, Device, Location …}

D (domain specific descriptors) = {Staff, PDA, Room …}

R (binary relations) = {isa, runsOn, isEngagedIn …}

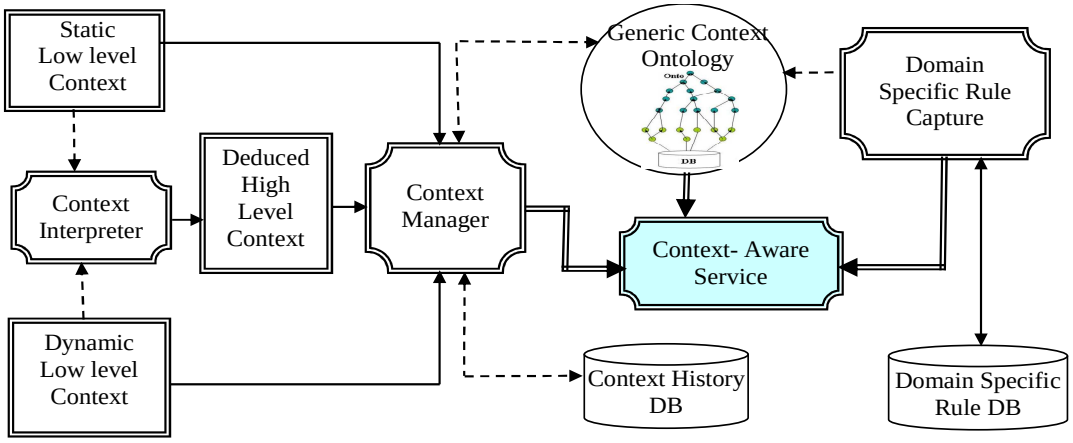


Figure 2 - Context flow and representation model

We use a directed graph G = (V, E) that consists a finite set V = {v_1,… v_n} of nodes (or vertices) and a finite set E = {e_1, … e_n} of edges (or arcs). In our case V = {Context}∪B ∪ D and each e ∈ E is an ordered triple (v_j, r, v_k) of v_j∈V, r∈R, and v_k∈V where V_j and V_k are called the *tail* and the *head* of the edge e respectively. We also have a restriction that a node v_0 = "Context" should exist at least ones only as a tail of an edge in E and it should never exist as a head.

Then more formally:

$$G = (V, E) \quad where$$

$$V = \{v_i : v_i \in \{Context\} \cup B \cup D\} \wedge$$

$$E = \{e_i : e_i = (v_j, r, v_k) \wedge v_j \in V \wedge r \in R \wedge v_k \in V\} \wedge$$

$$\exists e \in E : e = ("Context", r, v) \wedge \forall e \in E, e \neq (v, r, "Context")$$

Our context ontology G_{co} is then a transpose G^T=(V,E^T) of the graph G where G^T is the same graph G with all the arrows reversed:

$$G_{co} = G^T = (V, E^T)$$

All tails of the edges in E^T whose heads are the term "Context" are basic context descriptors (B):

$$\forall e \in E^T, \quad e = (v, r, "Context") \Leftrightarrow v \in B$$

All tails of the edges in E^T whose heads are not the term "Context" are domain context descriptors (D):

$$\forall e \in E^T, \quad e = (v, r, v_c) \wedge (v_c \neq "Context") \Leftrightarrow v \in D$$

Figure 3 shows an example of a simplified graph representation of the GCO model for two base ontology classes (*device* and *service*).

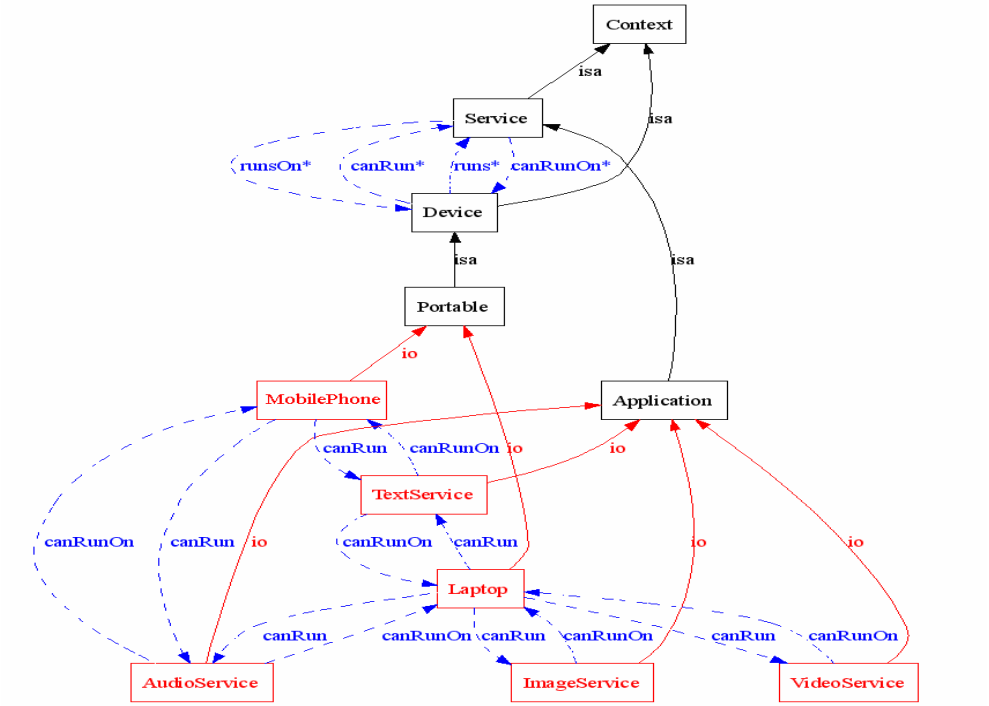


Figure 3 – The GCO model showing sample base and domain specific classes and instances.

One domain of application might involve adaptation of application services to devices using context of the device and the application itself. Detailed and expanded scenario of this example can be seen in the case study (section 5). As shown in the graph, relationships in this adaptation process can be expressed using predicates like: canRunOn(Service, Device), runs(Device, Service), etc.

In addition to the inherited properties from the base class, we can also define domain specific properties for the subclasses. Subclass hierarchy can also be extended further down to accommodate the structure of the problem in the specific domain.

SWRL [19], a Semantic Web Rule Language that combines the rule markup language (RuleML) and OWL can be used to express our decision rules in building domain specific knowledge for the context-aware services. For our experiment, we use the SWRL tool that exists as a plug in service in the OWL development environment in Protégé [20].

Rules combined with timely contextual facts are used to decide on the context-aware actions to be triggered. An example of the SWRL representation of rules and contexts for a simple rule that says "If an application is set to run on a device and if the device does not support a particular type of service (for example image display service) then lock this service in the application and take some other action like substituting the image with its description or look for some other content adaptation service" can be given as follows:

```
Sample Rule: //device does not support a service then lock or adapt it.
runs(?device,?service) ∧
type(?service, Application) ∧
not (canRunOn(?service, ?device) →
lockOrAdapt( service, device). // External function call
Context: //from ontology data or sensed on the fly.
runs(MobilePhone, ImageService)∧ //from sensed context
```

```
type(ImageService, Application) ∧ //form ontology
canRunOn(TextService, MobilePhone) ∧ //form data
canRunOn(AudioService, MobilePhone) ∧ //from data
canRunOn(VideoService, LapTop) ∧ //from data
canRunOn(ImageService, LapTop)∧ //from data
.. ..
```

Such rules can be incorporated into the ontology itself using the SWRL support in the Protégé editor or we can rewrite the rules using generic reasoners like, for example, the Jena[21] generic rule reasoner API in Java. This helps an implementation of the external functional calls like the lockOrAdapt() function above.

A small portion of a reasoning code that shows the core components for combining the context (app.ctxt), the rules (app.rules) and the ontology (app.owl) for making decisions and taking actions is given below.

```
//imports …
.. ..
public class OwlReasoner {
 public static void main(String[] args) {
  //Reasoning setup
  List rules = Rule.rulesFromURL("file:app.rules");
  OntModelSpec customInfSpec = new OntModelSpec(OntModelSpec.OWL_MEM);
GenericRuleReasoner reasoner = new GenericRuleReasoner(rules,
customInfSpec.getReasonerFactory());
   List context = Rule.rulesFromURL("file:app.ctxt ");
   reasoner.addRules(context);
   customInfSpec.setReasoner(reasoner);
   OntModel model= ModelFactory.createOntologyModel(customInfSpec, null);
   model.read("file:app.owl");
.. ..
//Call to usage functions
```

This context model is used in subsequent sections to represent and process contexts, rules and ontology and related knowledge to provide context-aware adaptation service.

4. ARCHITECTURE FOR ADAPTATION OF APPLICATION TO CONTEXT

In this section, we propose a generic platform to adapt applications to different context situations. The architecture of this platform is illustrated in Figure 4. It is based on four subsystems: the application core, the adaptation system, the context management system, and the user interface system. A set of components from each subsystem manage the operations needed for sensing and interpreting the context, and adapting consequently the application core and the user interface. The context management system uses the model defined in section 3 and the data coming from the domain definition interface (DDI). DDI is an interface that provides tools to define domain specific sub ontology and necessary rules for the knowledge management.

Different technologies can be used to build applications, and no one appears to dominate the current scenarios. The adoption of Web services, however, is widespread and is considered today a viable architecture for evolving applications, mainly due to its “loosely coupling” approach for the integration of application functions [22]. We therefore adopt a Web services paradigm for discussing about context-aware application design. Thus, we use web services to ensure the adaptation of applications to context.

Our adaptation system is composed from three modules: service adaptation module, content adaptation module and presentation adaptation module. The service adaptation module modifies the behavior of the application services by offering new adapted services that can be supported in different context situations. The content adaptation module provides the necessary content

transformations to adapt the multimedia content delivered to the user. Finally, the presentation adaptation module automatically generates a complete and adapted user interface to interact with the different adapted services of the application. The elementary entities of these three adaptation modules are modeled and implemented by the web services paradigm. We detail all the main functions of these modules in section 5.

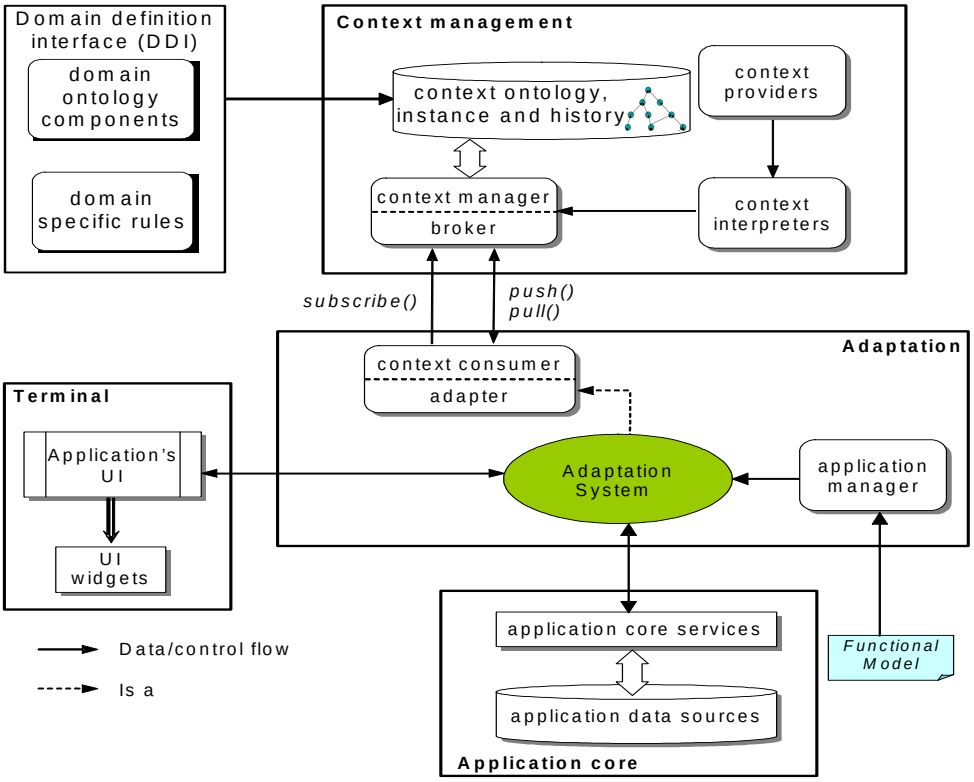


Figure 4 – Architecture for adaptation of applications to context

4.1 Context Management

Context management is highly dependent on the environment, i.e., on the physical context properties and on the sensors which capture and transmit raw data. We encapsulate such dependences in generic context providers, which are also responsible for managing the aspects of context related to the user, such as profile and history. Since the captured context may not be meaningful to the application, context interpreter modules translate the low level context into a high level representation (e.g., it maps geographical coordinates to a street address).

Context management incorporates the domain specific sub ontology and rules from the DDI into the context ontology hierarchy. This will be used by the context manager to generate the necessary knowledge for decision on the pull/push actions to provide a context-aware support.

Context entities, parameters, relationships and rules that govern these relationships are represented using the OWL language within the framework of our context ontology model. A part of context data is dynamic and volatile, and is consumed as it is acquired (e.g., location) while the other part does not change frequently, and its value may survive different execution sessions (e.g., user profile). Context ontology with all its related hierarchy of concepts, entities and relationships are linked to the database of both volatile and non volatile context instances. Some part of context

data may also be kept as part of the context history for pattern analysis and learning of user intensions. This will be used for decisions with respect to the push type consumption in a context–aware pervasive world.

In order to get the context, the adaptation system must subscribe to a *context broker*. In our case, due to the clear decoupling between the application services and the adaptation system, the context broker constitutes an interface between the context management system and the adaptation system. The broker carries the pertinent context parameter to the adaptation system. While subscribing, each adaptation service tells the broker which part of the context is relevant to it. The broker can therefore provide a specialized context view for each service.

Pull context consumers define logic rules while subscribing to the context broker: when the expression of the rule turns to "true", the context is pushed to the consumer. Push consumers receive the pertinent parameters of the context defined without explicit subscription. This view can dynamically evolve during execution, requiring some intelligence in the broker that is therefore tightly coupled with the context manager: it detects the changes of the context parameters and makes the necessary operations to refresh the instances of the classes defined in the context ontology and to activate push actions.

4.2 Preparing the Application for Adaptation

Before performing the adaptation of the behavior of an application to different context situations, we have to describe its offered functionalities to the user. In our platform, an application manager holds a session object for each client, containing the service references and their dependences. It is responsible for adapting the application by calling and properly linking the adaptation services.

Such adaptation must be established independently from the design and even from the implementation of the application. In fact, we have to add context-awareness after designing and implementing all the basic, non-adapted services that the application offers to the users. We call these services *"core services"*.

We model a service by a function R = f(X) getting input X and computing some output value R, where X = (x_1, x_2… x_m) and R = (r_1, r_2…r_n) are vectors of typed values. Each output value r_i (that we denote f.R[i]) may have a complex structure and a multiple cardinality (e.g., a table of records) r_i = (r_i^a, r_i^b…). This generic representation offers a standard exchange format between functions, which facilitates the composition and the adaptation of services. It is a simplified representation of existing complex models using other properties of services like states, effects, pre-conditions and post-conditions. We have made this choice to manage legacy applications in our context-aware architecture. In this type of applications we have very limited technical knowledge about the services offered to users. Our architecture aims to guarantee the adaptation of applications using the minimum knowledge on how they have been developed.

In the same application, services have functional dependences between them, since the input of a service may depend on the output of other services.

We use the notation (f_1, f_2… f_n) → f to model the dependence "f depends on (f_1, f_2… f_n)". This dependence can be illustrated by a Petri net representation where the places are the services and the transitions are the conditions that allow the application to invoke the next service. The main condition is the correct execution of the previous services. In the case depicted in Figure 5, the service f cannot be offered to the user (i.e., the transition cannot be fired) before completing the services f_1, f_2… f_n.

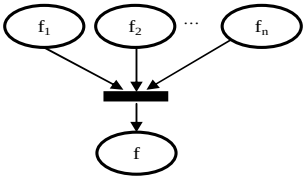


Figure 5 - A Petri net representation of service dependences

We define the *functional model* of the application by a non-autonomous Petri net [23] that describe the dependences between the services of the application.

Formally, a functional model of an application is a tuple N = (S, T) satisfying the following requirements:

(1) S is a finite set of services $\{f_1, f_2 \ldots f_n\}$

(2) T is a finite set of transitions $\{t_1, t_2 \ldots t_k\}$

(3) each t_i is a tuple (d, GC, A)

(4) d is the maximum delay for passing the transition

(5) GC is the general condition of the transition

(6) A is a finite set of associations $\{a_1, a_2 \ldots a_l\}$ between the services

(7) a_i is a finite set of pairs (sourceParameter, destinationParameter) specifying an association between two services by linking the outputs of the source service to the inputs of the destination service. SourceParameter can be a complex expression involving some basic operators (like number summation, string concatenation, Boolean evaluation, etc.) on the elementary output parameters of the source service.

Every transition of the Petri net is fired when the execution of all its entry services is completed successfully and the logical conditions on the output parameters are satisfied (e.g., “userID is not null”). An extra general logical condition that involves external events can be defined for every transition. By default, when the exit services of the transition are not context-aware, the value of the general condition is the “true” constant.

The root place of the net corresponds to the initial service offered to the user (e.g., user authentication). Every transition is time-controlled by a predefined expiration delay to avoid execution time deadlocks at the client side. The service can generate an error when it is not suitably invoked (e.g., due to a bad input vector). This type of error is controlled by logical expressions specifying conditions on service inputs.

At any time during execution, the user can go back to the previous place/service through an implicit transition which is fired when the user makes an explicit action on the client application (e.g., a mouse click on navigation buttons) or when the expiration delay of the transition is elapsed.

The deployment of the functional model is ensured by an XML descriptor which is a tagged representation of the application functional model. We have developed an extension of the Petri Net Markup Language [24] to describe services and their dependences. Figure 11 gives an example of a medical application.

The Petri net representation constitutes a formal way of modeling service dependences and interoperability that guarantees their correct composition and adaptation. The goal of this representation is not to build a specific application using generic services. It is a mean to describe the services of the existing application independently from context. The main goal is to adapt these

services and functionalities to new context situations. In the remainder of this section we detail our adaptation approach.

5. ADAPTING APPLICATIONS TO CONTEXT

Our adaptation system is based on three adaptation modules. The first takes the initial functional model of the application and performs service adaptation by applying transformations on their functional behavior. This module implies modifications on the entire structure of the functional modal. The second one ensures the adaptation of the provided content of the adapted application services. It applies all the necessary modifications on the multimedia data exchanged with the user and updates the description of the output and input parameters of the services in the functional model. Finally, the third module guarantees the automatic generation of an adapted user interface to the end-user device and to the user preferences. This module, called "the presentation adaptation module", takes as an input, the functional model provided by the content adaptation process and generates the necessary user interfaces to interact with its services.

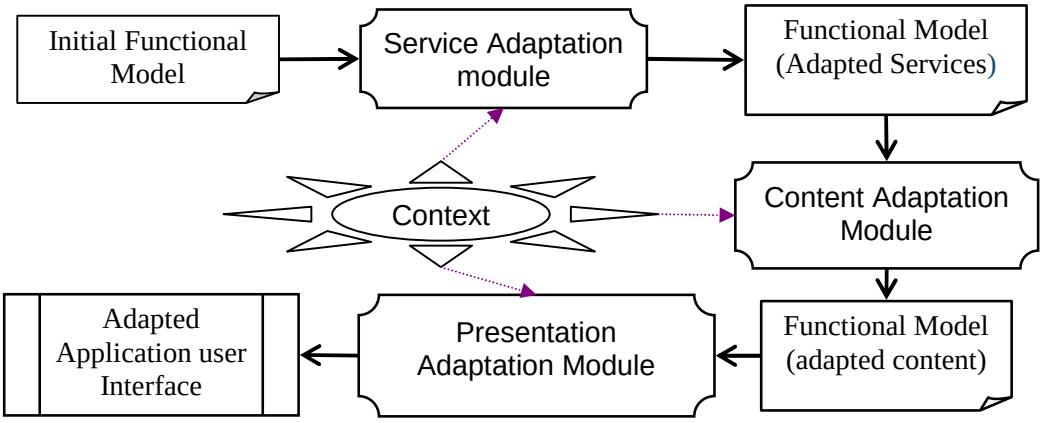


Figure 6 - application adaptation to context

5. 1 Service adaptation

Service adaptation performs the transformation of the Petri net of the services dependences into another adapted non-autonomous Petri net where the transitions can be controlled by external events coming from context providers. To guarantee this adaptation we add an adaptation layer on top of the application core. The adapted services must have many versions or instantiations, corresponding to different context situations. Service adaptation is guaranteed by tiers that we call adapters. Each adapter is associated with one original service of the application. The adapter associated with its original service and all its new different versions, constitute an adaptation entity.

In general, while the application services do not change, different adaptations may not be equivalent at the outer level, e.g., due to different data types handled in different contexts. The adapter can be written as *ad(X, cad(c))*/f_a, f_b, ..., where *ad* is the adapter service that chooses among the f_i according to the current context situation, X is the application data initially provided for the non-adapted service f, and *cad(c)* is the necessary view of the context c for the adapter service ad to perform the adaptation. *ad* knows the list f_a, f_b, ... of the available versions for a given service f.

The selection of the service instances is not the only task of the adaptor. It intercepts service calls and may apply adaptation actions on them. We distinguish two types of adaptation operators: the

first one specifies all the possible operators on the outputs and instances of the services; the second one collects all the adaptation transformations on the functional model (i.e., services dependences). The first type of adaptation operators is managed by the "adapter" (Figure 7). The second type is guaranteed by the service adaptation module (Figure 6).

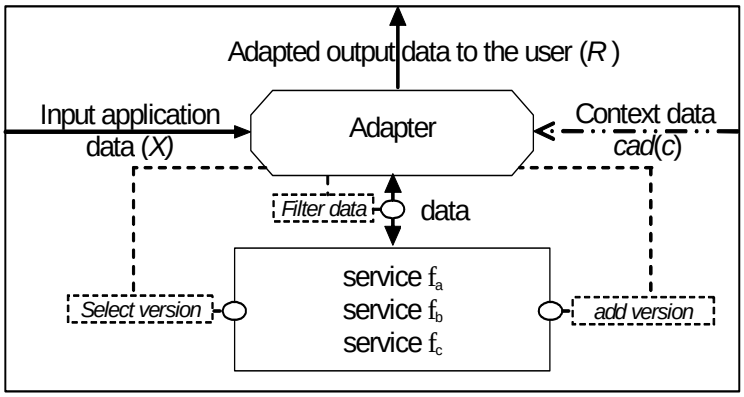


Figure 7 - Adaptation entity of a business service f

5.1.1 Operators on the services

• projectOutput: this operator is applied on the output vector R of the service and projects it on a subset of its components: e.g., the projection of R(r_1, r_2 …, r_n) on (r_2, …, r_n) is the vector (r_2, …, r_n). In this example, the elementary output information r_1 of the service is not returned to the user. This operator is useful when we need to hide some output values, or to prepare an entry data set for another service that uses only a part of the output of the previous service.

• selectOutput: this operator is applied on the output vector R of a service to remove some instances of the output data which are not relevant in the current context. They are identified through by some intentional specification (e.g., like in a where clause in SQL). For example, selectOutput(r>"17") means that we select all the instances where the value of the third component of the service output is greater than 17.

• joinOutput: this operator joins the output of a service with extra parameters, generally selected from the output of another service.

• selectVersion: this operator is used to select a specific version of a service among the available versions, according to the context situation.

• addVersion: this union operator is used to add a new version of a service to adapt its behavior to a new context situation.

5.1.2 Operators on the application's functional model

• lockService: this operator locks a transition in the functional model of the application by setting the general condition of the transition to the "false" constant. The services that depend on the locked transition become inaccessible. Using this operator we obtain a sub-net of the original functional model.

• addService: this operator inserts a new service in the functional model as a leaf of the Petri net, so helping in creating a context-aware application incrementally. All the dependences between the

newly added service and the existing services must be specified to guarantee the application consistency.

• insertService: this operator replaces a service f_i of the functional model of the application by an adapted service f_j that uses f_i (e.g., it splits the output of a service in chunks to avoid memory overload in devices of limited capabilities). The new service f_j must have the same dependences as the service f_i and must provide the same output structure R. This condition is necessary to maintain the application consistency.

5.1.3 Adaptation rules

The operators defined in this section are used by adaptation rules that associate contextual situations to adaptation actions. Each rule is modeled by a pair (Context Expression, Adaptation action). Context Expression is a logical expression defining a pertinent contextual situation for the adapter. In general, these logical expressions are formulated in terms of context statements and the properties of the application services. Adaptation action defines the operators that must be applied to adapt the service to the considered contextual situation. These rules are executed until the logical expression turns to false. For example, the following rule defines a contextual situation in which the terminal doesn't support images and we have services that provide image output in the actual application.

```
¬(actual_device acceptContentType image)∧ (actual_application hasService ?S) ∧ (?S
hasOutputVector ?Ov) ∧  (?Ov hasParameter ?P) ∧ (?P sys:Type image)→
 (actual_application hasAdaptation ?A) ∧ (?A hasOperator lockService) ∧ (?A
hasAdaptationParemeter ?S)
```

The adaptation action that is taken according to this rule is locking the access to these services in the functional model using the operator lockService(). The application manager holds a rule database stored in an ordered list. This order is very important since the execution of rules isn't commutative. We have let all the task of defining these priorities to the administrator of the application to supervise the results of their application and to maintain the semantic coherence of the application.

5.2 Content adaptation

The adapted functional model provides services that offer behaviors that are adapted to context. These services exchange multimedia content with the user. The content may not be supported by the used device or may not satisfy the user preferences. For example, a service can provide an image in a non supported format. In this case, a content adapter is needed to transform the format of this image to an adapted one. The content adapter may have more than on transformation to apply. For example, if we consider a service that provides a French text and the user is busy and her/his preferred language is English. In this case, we have to perform content adaptations. The first step may be translating the text from French to English and then generating an audio speech from the translated text or vice versa. Each content is described by statements $m(p_1, p_2 \ldots, p_n)$ where m represents a media and $p_1, p_2 \ldots p_n$ are its properties or metadata. A property can be the size, the format, the color depth… of the media. We associate a specific type to each combination of properties or metadata. This type will be verified for every parameter of every provided service to the user.

We have designed and implemented a content adaptation module that guarantees the adaptation of all the media types that will be delivered to the user. This module contains a content adaptation manager that identifies the different services that provide non adapted data. For each identified service, the manager instantiates a content adapter which uses a content adaptation engine to adapt

every non supported media in the current context situation. We have used an existing adaptation engine that has been developed in our research team by G. Berhe [25]. We have chosen this adaptation engine because it provides an optimized adaptation of each media by choosing the best composition of the suitable content adaptation services to adapt each media to context. This engine uses on a proxy based adaptation method that offers a distributed execution of the adaptation process. It generates a graph of every possible combination of adaptation services and looks for the optimum path in the graph to achieve an adapted format to actual context situation. Figure 8 illustrates the general architecture of the content adaptation module in our platform. The content adapter delegates the adaptation process of every non adapted parameter to the local adaptation proxy as specified in [25]. The proxy uses an adaptation planner that generates a graph of all the possible compositions of adaptation services that can lead to supported content. The planner selects these services from a Content Adaptation Registry where we reference elementary content adaptation services. Each service is a possible instance of a content adaptation operator modeled by the expression $t(a_1, a_2 \ldots a_n)$ where t is a transformation and $a_1, a_2 \ldots a_n$ are parameters. For example, *jpegToGif(jpeg:dynamicSize, gif:dynamicSize)* is a transformation from the jpeg to gif image format. The adaptation planner identifies the necessary adaptation operators to achieve an adapted content to the actual context situation. These adaptation operators can have many possible executing services. The adaptation Planner generates the graph of all the possible combinations of the available services in the content adaptation registry of the module. This graph is produced by an algorithm called Multimedia Adaptation Graph Generator (MAGG) [25]. Then, the adaptation planner selects the optimum path in the resulting graph regarding the cost of each node. An aggregate cost function has been defined in the same work to identify the optimum adaptation path in the graph for each non adapted parameter. The content adapter updates the description of the attached service by setting the parameters types according to the results of the predicted content adaptation process. This generates a new functional model producing adapted content to the contextual situation. This new model will constitute the entry of the presentation adaptation module that we will present in the next paragraph.

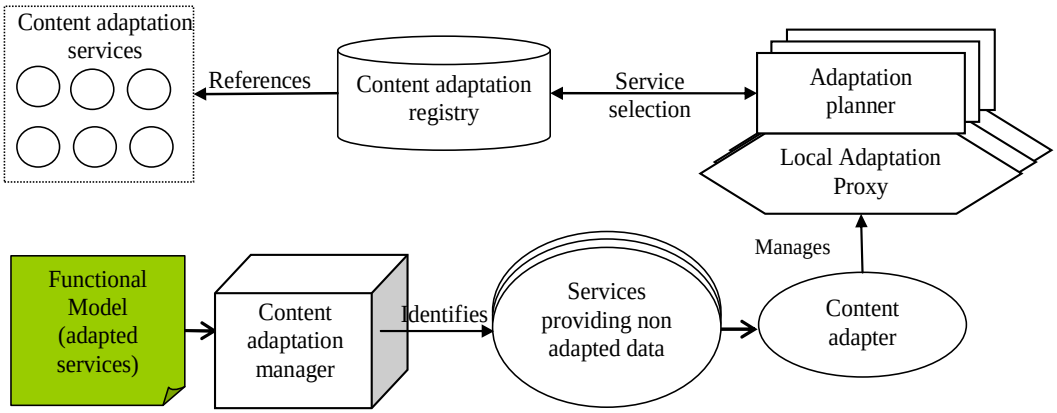


Figure 8 - The architecture of the content adaptation module in our platform

5.3 Presentation adaptation

In our architecture, the presentation adaptation is based on the automatic generation of the complete code of the necessary user interfaces that ensures the user interaction with the different adapted services of the functional model. For each service of the application, the presentation

adaptation module generates an input interface and an output interface that guarantees the interaction between the user and the application services. These interfaces provide navigation facilities to navigate between the services in the functional model of the application. The presentation adaptation process follows the content adaptation planning in order to prepare the adequate human-computer interface to the user. The presentation adaptation manager takes the adapted functional model and generates an abstract window description for each service of this model. This description is based on a set of abstract interaction components that we have defined according to the data types returned by the service. A user interface generator produces the complete code of the necessary interaction screens for each service. This generator uses a vocabulary describing the available interaction components on the used terminal. The generation process takes also in account the user preferences regarding presentation modalities from his context profile. Figure 9 presents the architecture of the presentation adaptation module in our platform.

We have defined a vocabulary that describes the different interaction APIs on object oriented target platforms. This vocabulary was the result of our previous study [26] on the available interaction APIs of the market. We have designed this abstract vocabulary to address the problems of diversity and standardization of the programming languages on the user terminals of the market. Our vocabulary is based on high level object oriented principles to cover the majority of the available interaction APIs. We have developed this vocabulary in an XML format containing three general structures [26]. A user interface has: (i) a logical structure describing its composition in terms of elementary interaction entities; (ii) a behavioral structure defining how to interact with each defined entity in the logic structure and (iii) a physical structure defining their layout on the physical screen of the end user device. The separation between these three structures simplifies the description of an interaction software platform and helps reusing the same components with several layouts and vice versa. For each device used in the application, we provide an instance of this vocabulary. We attach this instance to the device class in the context ontology.

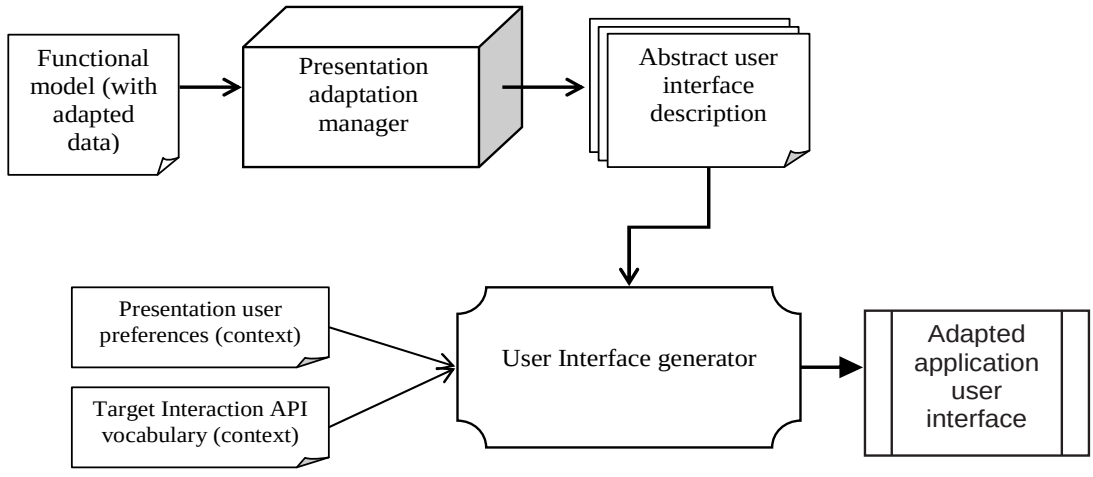


Figure 9 - the architecture of the content adaptation module in our platform

To generate adapted user interfaces to their execution environment, we have defined a generic user interface abstract model. This model is based on the abstract interaction entities defined in the generic interaction vocabulary. These entities are instantiated in the concrete context targets to provide the code of the interface. We have chosen the MVC model [27] to organize user interface

code. This model is the most adequate for our problem since it ensures a complete separation between display, data and the tasks of user interfaces. Thus, the code of a user interface is composed of a view **V** (graphical display), a model **M** (service calls and data management) and a controller **C** (event management). The controller ensures the connection between the model and the view when an interaction event occurs on the interface.

We associate a generic description AWD (abstract window description) for each user interface I. AWD will be given to the user interface generator to produce the user interface code I. In the AWD, we describe the necessary abstract components to the interaction with the associated service. The granule atom of our model is the component. A component is an interaction entity that will be associated to an input or an output parameter of a service. We organize components in panels. A panel treats a homogeneous set of data of a specific task corresponding to an interaction with a service of the application. The description AWD of an interface I is composed by the description of a set of panels. In our platform, we associate two panels for each service of the functional model. The user interface I, is composed of two panels: an input panel and an output panel. The input panel guarantees the collection of the input parameters of the associated service whereas the output panel transmits the result of the service invocation to the user. The components of a panel can interact among them (value transmission between component), with a distant service or with internal application variables. These internal variables constitute a means of information exchange between the services of the application's functional model. The associated value with this component can be multi-valued in accordance with the description of a service provided in the functional model. In our platform, the description "AWD" of a user interface is composed of the description of two panels P^i and P^O associated with a service f. P^i is the input panel that collects the various input parameters of a service f. P^O is the panel that gathers the output parameters of the service f. Each interface has a control (button, menu…) that triggers the execution of the service. We call this control "trigger" (Tr). Each panel is composed from interaction components. They are identified by our generator using the parameter types of the associated service. For example, if we have a service that computes the sum of two integers than the input panel will hold two components that can get integers from the user and the output panel will have one component to render the result of the sum.

The user interface generator uses the vocabulary describing the available interaction API in the current contextual situation to associate concrete interactions entities to the abstract window components. The layout of the components on the end user screen is also defined by layout rules in the target vocabulary. They can be also specified in the user preferences. Our user interface generator uses the default layout manager of each component when user presentation preferences can not be satisfied. The code corresponding to the instantiation and the layout of these components constitutes the code of the view **V** of the user interface.

Our generator produces the model **M** of the interface using the description of the service in the functional model. This model produces the necessary functions to the collection of the input parameters of the service f, its invocation and the extraction of the results. The controller **C** of the interface is generated from the description of the "trigger" (Tr) corresponding to the invocation of the associated service to the interface. The controller contains all the necessary code to manage the different interaction events on the user interface (mouse click, button click, menu selection…).

Finally, we obtain the code of the user interface by aggregating the model M, the view V and the controller **C**. This code is stored in a local cache and sent to the end user to interact with the adapted services that provide adapted content.

6. IMPLEMENTATION AND CASE STUDY

6.1 Implementation of our adaptation approach

According to [28], [29] and [30], efficient use of context models needs careful planning. In our case ontology as a structure for context hierarchy representation and rule management keeps the hierarchy of the necessary adaptation operators and their parameters in our adaptation process. To use these operators and adapt applications to new context situations, we have added a new domain class named *Application* in the context ontology model under the *Service* base ontology class.

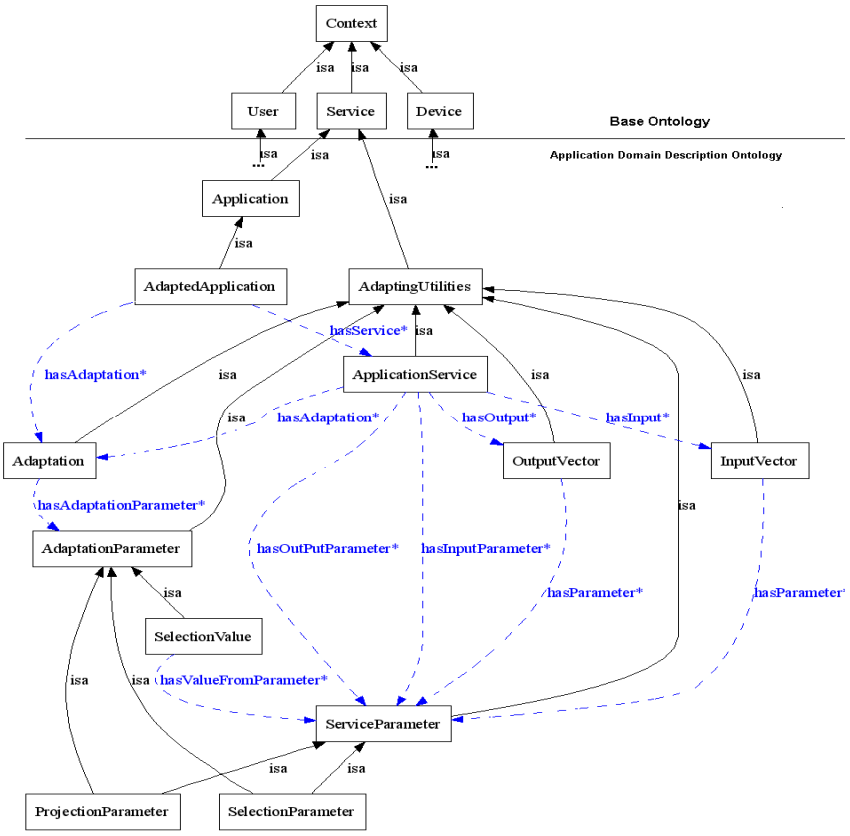


Figure 10 – Example of context ontology for the adaptation of application to contexts

Figure 10 shows the relationship between the general *context base ontology* part and *the application domain description* part. This relation keeps hierarchy of context entities in our adaptation process to describe the services of the application and the different adaptation operators that we have defined in previous sections. In addition to the components indicated in this figure, we have also defined other subclasses under the base ontology classes and their relationship with one another. Under *Device* class, for example, we have a sub class *Terminal* which is associated, to a *hasMemory* property.

In general, it is up to the application designer or the user to add or remove the domain specific classes, sub classes, domain based class properties and relationships using the DDI interface. Adaptation rules based on changing context can also be defined by designers or users using this interface.

The context manager module is then responsible to use the knowledge in the ontology to activate adaptation. This adaptation of the application can take place using push or pull mechanism as we have discussed it earlier in this section.

The adaptation processes that can be applied in a specific context situation are defined by a combination of predicates and facts on the A*pplication domain description part* and its components (application, services and their output and input parameters). To integrate the knowledge required for adaptation within the context ontology, rules in our application are described using the SWRL language. Predicate based rules in SWRL, are easily transported and integrated into OWL. The Jena parser and query engine tools give an easy interpretation and access to the content of our OWL representation of the context ontology hierarchy, rules and facts. Section V gives concrete examples of these rules in the scope of adapting an existing medical application to new contexts.

We remind that the administrator of the platform provides only two simple configuration files to adapt an application to new context situations: the functional model and the adaptation rules of the application. The entire adaptation process is done automatically by our platform. This process is based on 13 generic steps:

1.An administrator uses the application manager to deploy the functional model descriptor
2.The platform prepares the adaptation layer by instantiating an adapter for each service
3.The platform asks the designer to provide the necessary adaptation rules for services adaptation.
4.The application manager uses adaptation rules to link between the adapters and the context broker to make the context parameters accessible to them.
5.The service adaptation manager configures the adapters using operators specified in the rules.
6. The service adaptation manager updates the functional model according to the implied transformations by the service adaptation operators.
7. The service adaptation manager sends the “end service adaptation planning” message to the application manager.
8. The content adaptation manager identifies all the services that provide non adapted content to the current context and instantiates a content adapter for each one of them.
9. For each non adapted parameter, the associated content adapter instantiates a local adaptation proxy to identify the necessary content adaptation services to perform adaptation of its corresponding content.
10. The content adaptation manager sends the “end content adaptation planning” message to the application manager
11. The presentation adaptation manager generates the abstract description of the user interface that ensures the interaction with the different services of the application.
12. The user interface generator provides an adapted code of the user interface according to the user device and preferences.
13. When the context changes, the adaptation process is restarted from step 5.

6.2 A case study from the medical domain

Figure 13 illustrates a screenshot of an existing application for consulting a dialysis patient record. It is composed of three services: the first one offers a service returning general information about the dialysis record of a patient and his/her prescribed treatment; the second one presents a service that returns the evolution of the patient’s temperature; the third entity displays a list of images from the same record. A menu bar ensures the navigation among these services. Figure 11 presents the initial functional model of these medical services, as used by a practitioner on a standard PC. Figure 12 presents the adapted functional model of these services when the context situation has changed: the same practitioner uses a smart phone and not his desktop PC.

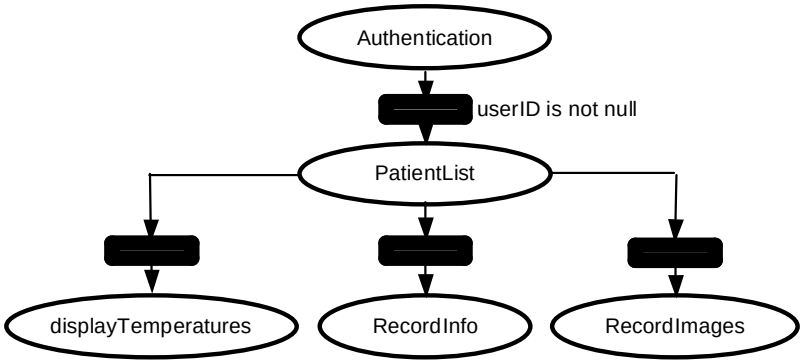


Figure 11 - The initial functional model of the medical application of Figure 12

Figure 13 is the equivalent of the same application of Figure 14 for mobile terminals. The same first service is used (i.e. there is no adaptation for this service). The application manager decomposes the second service displayTemperatures() into two services by applying the adaptation Rule (1).

```
Rule 1
  //context part
  (actual_device hasMemory ?M)∧ (?M sys:lessThan 3) ∧ (actual_application hasService ?S)∧
  (?S hasOutputVector ?Ov)∧ (?Ov length ?N)∧
  (?N, sys:greaterThan 1) →
  //adaptation part
   (?S hasAdaptation ?A1)∧ (?A1 hasName firstValues)∧ (?A1 hasOperator projectOutput)∧ (?A1
   hasAdaptationParemeter ?Op1)∧(?S hasOuputVector ?Ov1)∧ (?Ov1 hasParameter ?Op1)∧(?Op1
   hasIndex 1) ∧ (?S hasAdaptation ?A2)∧(?A2 hasName selectedInstance)∧
   (?A2 hasOperator selectOutput)∧ (?A2 hasAdaptationParemeter ?Op21)∧ (?S hasOuputVector
   ?Ov2)∧ (?Ov2 hasParamter ?Op21)∧ (?Op21 hasIndex 1) ∧ (?A2 hasAdaptationParameter
   ?Op22)∧(?Op22 hasValueFromParamter ?Op21 )∧ (actual_application hasAdaptation ?A3) ∧
   (?A3 hasOperator insertService) ∧ (?A3 hasAdaptationParemeter ?S) ∧ (?A3
   hasAdaptationParemeter ?A1) ∧ (actual_application hasAdaptation ?A4) ∧ (?A4 hasOperator
   insertService) ∧ (?A4 hasAdaptationParemeter ?S) ∧ (?A3 hasAdaptationParemeter ?A2)
```

In this rule, the service firstValues() is defined as returning the projection of the result R (the output vector of a service) on its first parameter. The second service, selectedInstance() is defined as returning the selection in R of lines which have the first component R[i] corresponding to the selection of the user among the results of the service firstValues(). The insertService operator replaces the call to the original service by calls to the adapted service. The adapted service itself calls the original service. The third entity providing access to a list of images is also firstly adapted by applying Rule (1). At a second stage, Rule (2) is applied to obtain three services in the new context situation: the first, firstValues(), gives the list of the returned images by the initial non adapted service. The second, displayImage(), presents the selected image. The third service, displayNoImage(), gives the textual description of the selected image.

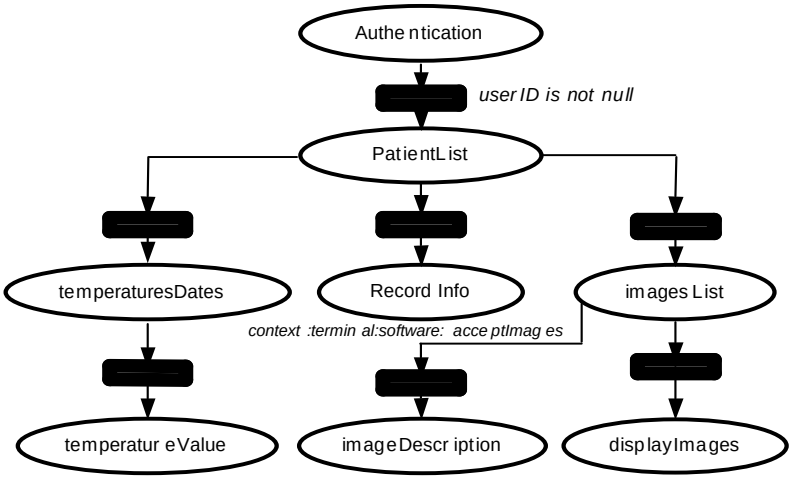


Figure 12- The adapted functional model of the same medical application where the terminal context has changed (PC smartphone)

Rule 2
//context part
(actual_device acceptContentType image)∧ (actual_device hasDeviceType cldc) ∧ (actual_application hasService ?S)∧ (?S hasOutputVector ?Ov) ∧ (?Ov length ?N)∧ (?N Sys:greaterThan 1) ∧ (?Ov hasParameter ?P) ∧ (?P, rdf:type image) →
//adaptation part
(?S hasAdaptation ?A1) ∧ (?A1 hasName displayNonImage)∧ (?A1 hasOperator projectOutput)∧ (?A1 hasAdaptationParemeter ?P1) ∧ ¬(?P sys:isEqual ?P1) ∧ (?S hasAdaptation ?A2) ∧ (?A2 hasName displayImage) ∧ (?A2 hasOperator projectOutput) ∧ (?A2 hasAdaptationParemeter ?P) ∧ (actual_application hasAdaptation ?A3) ∧ (?A3 hasOperator insertService) ∧ (?A3 hasAdaptationParemeter ?S) ∧ (?A3 hasAdaptationParemeter ?A1) ∧ (actual_application hasAdaptation ?A4) ∧ (?A4 hasOperator insertService) ∧ (?A4 hasAdaptationParemeter ?S) ∧ (?A3 hasAdaptationParemeter ?A2)

These three adapted services are inserted in the functional model using the insert service operator, and the calls to the original service are re-directed to the new services.

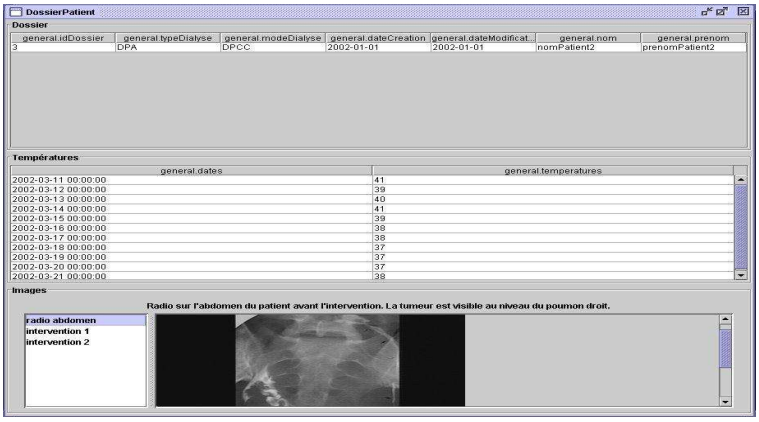


Figure 13- A dialysis patient record on standard PC

Figure 14- The same dialysis patient record on a smartphone

If the terminal doesn't support images, the selection of an image directly leads the user to its textual description (window E6 on figure 14) and the displayImage() service is locked by the generic rule (3).

Rule 3

//context expression

¬(actual_device acceptContentType image)∧ (actual_application hasService ?S) ∧ (?S hasOutputVector ?Ov) ∧ (?Ov hasParameter ?P) ∧ (?P sys:Type image)→

//adaptation action

(actual_application hasAdaptation ?A) ∧ (?A hasOperator lockService) ∧ (?A hasAdaptationParemeter ?S)

The content adaptation module resizes and transforms the provided images of the adapted service displayimage(). The presentation adaptation module provides the complete code of the displayed user interface in figure 13. For each service, the user interface generator of this module produces an input panel if the service needs user input (which is not the case of for these services) and an output panel to display the output parameters resulting from its invocation.

7. CONCLUSION AND PERSPECTIVES

In this paper, we have presented a comprehensive context model and a web services architecture that guarantees the adaptation of applications to new contexts. The context model uses ontology representation based on the basic context descriptors. These descriptors are considered as base classes in the ontology hierarchy to represent domain independent concepts. The domain specific subclasses are defined as sub classes of the base ontology classes. The model we proposed, also benefits from the features of OWL and related ontology rule languages for managing context data.

Adaptation of the application to context is based on a Petri net representation of its different services. We have proposed to implement this adaptation by using web services because they offer an easy, flexible and reusable mean for deploying and integrating the adaptation to context at many levels in the application. We have also defined a complete approach to adapt applications to new context situations. This adaptation concerns the three components of applications: services, content and user interface. First, we adapt the services of the application by modifying their behavior using a set of adaptation operators on the services of the application and on its Petri net representation. These operators are to be used in specific or generic adaptation rules using the OWL language.

Second, we adapt the delivered content to the user by selecting and composing different content adaptation services. In addition, our architecture provides a presentation adaptation module that generates adapted user interface to interact with the available services of the application. We have developed a case study to use our adaptation architecture and our context model in a medical system environment using concrete adaptation rules. The architecture is general enough to let users add new adaptation operators and rules to achieve a thorough degree of service adaptation in pervasive environments.

As a continuation to this work, we plan to work on a benchmark that will help to evaluate the performance of our proposed modeling and adaptation approaches. Another important continuation to this work on which we are currently engaged is to develop a multi-domain development platform that supports reasoning and decisions in a pervasive context-aware systems independent of the domain of application. This work will also consider the issues of plenty of participating tiny devices that require collaboration for sharing computing, data and other resource. We also intend to provide a comprehensive tool that helps the administrators of our architecture to create the functional model of legacy applications. In this context, we propose to use communication monitoring between clients and the different services of the application to identify and describe their inputs and outputs.

REFERENCES

[1] J. Birnbaum "Pervasive Information Systems," Communications of the ACM 40 no 2, february 1997.

[2] J. Coutaz, J. Crowley, S. Dobson and D. Garlan, "Context is key," Communications of the ACM 48(3). March 2005.

[3] T. Strang and C. Linnhoff–Popien, "A Context Modelling Survey," In Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp4). Nottingham/England, 2004.

[4] K. Henricksen, J. Indulska and A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," Proceedings Pervasive 2002 -Zurich August 2002.

[5] X. Wang, D. Q. Zhang, T. Gu and H. K. Pung "Ontology Based Context Modeling and Reasoning using OWL," Workshop on Context Modeling and Reasoning at IEEE International Conference on Pervasive Computing and Communication, Orlando, Florida, March 2004.

[6] A. Held, "Modeling of Context Information for Pervasive Computing Applications," Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002), Orlando, FL, July 2002.

[7] H. Chen, T. Finin and A. Joshi, "An ontology for context-aware pervasive computing environments," Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.

[8] P. Dockhorn Costa, L. Ferreira Pires and M. van Sinderen, "Architectural Patterns for Context-Aware Services Platforms". IWUC 2005 : 3-18 Soraya Kouadri Mostéfaoui , Zakaria Maamar (Eds.): Ubiquitous Computing, Proceedings of the 2nd International Workshop on Ubiquitous Computing, IWUC 2005, In conjunction with ICEIS 2005, Miami, USA,. INSTICC Press 2005, ISBN 972-8865-24-4, May 2005.

[9] A. K. Dey, D. Salber and G. D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," Human-Computer Interaction Journal 16(2-4), pp. 97-166, 2001.

[10] T. Kindberg and J. Barton, "A Web-Based Nomadic Computing System," Computer Networks, Elsevier, 35(4), pp. 443–456, 2001.

[11] K. Henricksen , J. Indulska, T. McFadden and S. Balasubramaniam, "Middleware for Distributed Context-Aware Systems,". OTM Conferences (1) Agia Napa, Cyprus, October 31 - November 4, 2005,

Proceedings, Part II. Lecture Notes in Computer Science 3761 Springer 2005, ISBN 3-540-29738-3.: 846-863, 2005.

[12] C. Efstratiou, A. Friday , N. Davies and K. Cheverst, “A Platform Supporting Coordinated Adaptation in Mobile Systems,”. 128-137 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), 20-21, Callicoon, NY, USA. IEEE Computer Society 2002, ISBN 0-7695-1647-5 20, June 2002.

[13] R. W. DeVaul and A. S. Pentland, “The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications,” MIT Technical Report, 2000.

[14] K. Eustice, T. J. Lehman, A. Morales, M. C. Munson, S. Edlund and M. A. Guillen, “Universal Information Appliance,” IBM Systems Journal, 38(4), pp. 575–601, 1999.

[15] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman and D. Zukowski, “Challenges: An Application Model for Pervasive Computing,” Proc. 6th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom 2000), pp. 266–274, Massachusetts, USA, 2000.

[16] A. K. Dey, and G. D. Abowd, “Towards a Better Understanding of Context and Context-Awareness,” CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, The Hague, The Netherlands, 2000.

[17] T. Winograd, “Architectures for Context,” Human-Computer Interaction, Vol. 16, No. 2, 3 & 4, Pages 401-419, 2001.

[18] OWL web Ontology Language overview, http://www.w3.org, available in October 2006.

[19] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, Version 0.5 of 19 November 2003, http://www.daml.org/2003/11/swrl/, available in October 2006.

[20] Protégé, an ontology editor and knowledgebase framework, http://protege.stanford.edu/, available in October 2006.

[21] Jena–A Semantic Web Framework for Java, SourceForge.net, http://jena.sourceforge.net/, available in October 2006.

[22] D. Austin, A. Barbir, C. Ferris and S. Garg, “Web Services Architecture Requirements” W3C Working Draft, 19 August 2002, http://www.w3.org/TR/2002/WD-wsa-regs20020819, available in October 2006.

[23] R. David, H. Alla. Petri Nets and Grafcet, “Tools for Modeling Discrete Event Systems,” Prentice-Hall, London, 1992.

[24] J. Billington et al. “The Petri Net Markup Language: Concepts, Technology, and Tools,” In W. van der Aalst and E. Best, eds., 24th Int. Conf. on Application and Theory of Petri Nets, LNCS 2679, pp. 483–505, 2003.

[25] G. Berhe, L. Brunie, J. M. Pierson, "Content adaptation in distributed multimedia systems", Journal of Digital Information Management, Special Issue on Distributed Data Management, Volume 3, No 2, pp.95-100, June 2005,

[26] T. Chaari, B. Elloumi, F. Laforest, A Generic Description Language for the Automatic Generation of Pervasive Medical User Interfaces: The SEFAGI Project. Health Pervasive Systems Workshop, edited by the IEEE, Lyon, France, June 2006.

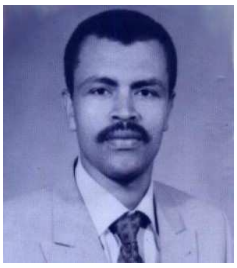
[27] A. Goldberg, "Smalltalk-80: The Interactive Language Environment", Addison-Wesley, 1984.

[28] T. McFadden, K. Henricksen, J. Indulska, Automating context-aware application development. In: UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham 90-95, 2004.

[29] T. McFadden, K. Henricksen, J. Indulska, P. Mascaro, Applying a disciplined approach to the development of a context-aware communication application. In: 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society 300—306, 2005.

[30] K. Henricksen, J. Indulska, Developing context-aware pervasive computing applications: Models and approach, Pervasive and Mobile Computing, In Press, Elsevier, 2005.

Tarak CHAARI, is a PHD student in the CNRS UMR 5205 LIRIS Laboratory (Lyon Research Center for Images and Intelligent Information Systems). His research interests focus on adaptation in the scope of pervasive computing and context–aware systems. He holds a master degree in information systems from INSA Lyon on automatic user interface generation. He teaches information systems modeling and development in INSA Lyon. He is a Member of dynamic adaptation to runtime environments action and member of the research working group on impact of grid computing, peer-to-peer-computing and mobile computing on database systems. More information can be found at:
http://liris.cnrs.fr/tarak.chaari



Dejene Ejigu is a Ph.D. candidate at INSA de LYON in the CNRS UMR 5205 LIRIS Laboratory (Research Center for Images and Intelligent Information Systems). His research interests include pervasive computing, intelligent systems and distributed computing. His current research work focuses on context management, modeling, and reasoning, development of reusable context-aware application platforms, and their deployment in pervasive computing. He received his M.Sc. degree from the University of Wales Swansea in the area of Intelligent Computing. In the past, he worked as a computer science lecturer and researcher at the Addis Ababa University. He can be reached at: dejene.ejigu@insa-lyon.fr.

Frédérique LAFOREST is associate professor in the CNRS UMR 5205 LIRIS Laboratory (Lyon Research Center for Images and Intelligent Information Systems). She teaches pervasive information systems and applications modeling at the INSA Lyon. Her research interests concern document-based user interfaces to databases, multi-terminal user interfaces generation, and adaptation of applications to the context. More information can be found at:
http://liris.cnrs.fr/frederique.laforest

Vasile-Marian Scuturici is associate professor at INSA of Lyon since 2004. He received his PhD in computer science in 2001 at Lyon-2 University. His research interests include pervasive systems, video-on-demand systems, multimedia mining. His current work concerns the management of multimedia data in pervasive environments. More information can be found at:
http://liris.cnrs.fr/vasile-marian.scuturici

Projet SoCQ

[18]
Yann Gripay, Frédérique Laforest, and Jean-Marc Petit. Towards Action-Oriented Continuous Queries in Pervasive Systems. In *Bases de données Avancées 2007 (BDA'07)*, October 2007.

Towards Action-Oriented Continuous Queries in Pervasive Systems

Yann Gripay, Frederique Laforest, Jean-Marc Petit
Université de Lyon, INSA-Lyon, LIRIS – UMR 5205 CNRS
7 avenue Jean Capelle
69621 Villeurbanne cedex, France
Yann.Gripay,Frederique.Laforest,Jean-Marc.Petit@insa-lyon.fr

Abstract

Pervasive information systems give an overview of what digital environments should look like in the future. From a data-centric point of view, traditional databases have to be used alongside with non-conventional data sources like data streams, services and events to deal with new properties of such information systems including dynamicity, autonomy and decentralization.

In this context, the definition of continuous queries combining standard relations, data streams and services in a declarative language extending SQL is clearly an ambitious and motivating goal. Those continuous queries could express an event management functionality (e.g. event filtering, event composition), associate events with data from legacy systems, and perform cost-based optimal calls to services.

In this paper, we define virtual tables with binding patterns to represent services of the pervasive environment. By the way, relations, data streams and virtual tables can be homogeneously queried using a SQL-like language, on top of which query optimization can be performed. We also introduce a new clause defining the optimizing criteria to dynamically choose the best way to handle each event.

A prototype on top of STREAM, a DBMS devoted to data streams, has been devised on which first experiments have been carried out on synthetic data.

Keywords Continuous Query; Query Optimization; Non-Conventional Data Sources; Data Streams; Services; Pervasive Systems

1 Introduction

Pervasive information systems give an overview of what digital environments should look like in the future. Information systems tend to be more and more decentralized and autonomous, at the infrastructure level as well as at the data and process level. On the one hand, personal computers and other handheld devices are now democratized and take a large part of information systems. On the other hand, data sources may be distributed over large area through networks that range from a world-wide network like the Internet to local peer-to-peer connections like for sensors.

Even data tend to change their form to handle information dynamicity. The relational paradigm has been widely adopted in DataBase

Management Systems (DBMS) for many years, but other forms of data sources are now emerging, mainly as data streams and services.

Data Streams Queries in traditional DBMS are "snapshot queries" expressed in SQL: a query is evaluated with the current state of the database, and the result is a static relational table. The "snapshot" term expresses that the result represents only the state of the database at the moment of the query, and is never updated. With dynamic data sources, "snapshot queries" may be not sufficient as it would be computation-expensive to periodically execute them and obtain up-to-date results.

Data streams open new opportunities to view and manage dynamic systems, such as sensor networks. The concept of queries that last in time, called *continuous queries* [10], allows to define queries whose results are continuously updated as data "flow" in the data streams. Data Stream Management Systems (DSMS) have been studied in many works [1, 3, 8, 11, 15, 23, 26].

Services With the development of autonomous devices and location-dependent functionalities, information systems tend to become what Mark Weiser [24] called *ubiquitous systems*, or *pervasive systems*. Pervasive systems [6, 7, 17, 19, 20] are distributed systems of devices able to communicate with others through network links. They offer to users access to devices and control over their environment through various types of interfaces.

The abstraction of device functionalities allows the system to automate some of the possible interactions between heterogeneous devices, in order to facilitate the use of the whole system. Such device functionalities are often represented by services. Service discovery is a common issue [27] in distributed systems (pervasive systems, grids, or even Internet), that tackles service representation, knowledge sharing, and remote execution. In dynamic environments like pervasive systems, the discovery should also be dynamic in order to reflect the currently available services.

As devices may be sensors or actuators [13], services may represent some interactions with the physical environment, like taking a photo from a camera or displaying a picture on a screen. These *actions* bridge the gap between the computing environment and the user environment, and can be managed by the pervasive system. This paper will not consider service discovery techniques, but will consider a way to use and compose services with the notion of *action-oriented* queries.

From a data-centric point of view, traditional databases have to be used alongside with non-conventional data sources like data streams, services and events to deal with new properties such as dynamicity, autonomy and decentralization. Query languages and query processing techniques need to be adapted to those data sources. In this context, the definition of continuous queries combining standard relations, data streams and services in a declarative language extending SQL is clearly an ambitious and motivating goal. We begin by illustrating the problem with an example that will be used as a running example throughout this paper.

1.1 Motivating Example

The motivating example is inspired by the night surveillance scenario presented in Aorta [25]. It illustrates the need for the integration of services from a dynamic environment in a declarative query language and for associated optimization techniques.

The night surveillance scenario considers a room containing motion sensors and network cameras. The surveillance consists of handling events from motion sensors to trigger a photo of the location of the involved sensor and to send it to the administrators via their cell phones. The cameras need to pan/tilt/zoom to focus on a given location (if achievable) before actually taking the photo. This configuration phase is costly in term of response time to an event and depends on the dynamic state of the device (current head position of the camera), so a cost-based evaluation of the optimal device is needed, or even a group optimization to benefit from parallelism.

In order to express this behavior in a declarative way, the environment can be described using data schemas for the entities and the events, and functions for the actions and the predicates. Then, a query language similar to SQL can express the specified behavior in terms of joins, selections and functions. Query optimization techniques can be applied to optimize the entire process.

In Aorta [25], this environment is modeled using three data sources: a relation containing phone numbers of administrators, a data stream for sensor events (indicating its current location and its horizontal acceleration value 'accel_x'), and a "virtual device table" for cameras. Three functions are also needed for the scenario: the action to take a photo, the action to send a photo, and a predicate that checks that a camera is able to take a photo of a location. We made a synthesis of this environment in Figure 1.

The continuous query for the night surveillance scenario is given in AortaSQL [25] in Figure 2: an *Action Query* called "night_surveillance" is active from midnight to 6:00 am every day (cf. START and STOP clauses).

```
Data Sources:
  relation phone( id, owner, number );
  stream sensor( id, accel_x, location );
  virtual device table camera( id, ip_address );

Functions:
  photo(camera_ip, location, directory_name) :
    file_name
  sendphoto(phone_number, file_name) : void
  coverage(camera_id, location) : boolean
```

Figure 1: The environment for the night surveillance scenario from Aorta [25]

```
CREATE AQ night_surveillance AS
  SELECT sendphoto(p.number,
           photo(c.ip_address,s.location,
                 "photos/admin"))
  FROM   sensor s, camera c, phone p
  WHERE  s.accel_x > 500
    AND  coverage(c.id, s.location)
    AND  p.owner = "admin"
  START  atTime(0,0,0) -- 00:00:00
  STOP   atTime(6,0,0) -- 06:00:00
```

Figure 2: The query for the night surveillance scenario from Aorta [25]

Despite the interest of Aorta, the following observations can be made:

1. At the query language definition level, no clear distinction is made between event management and stream management. For example, in the above scenario, an event is represented as a tuple in the 'sensor' data stream, but is however still handled as an event: it triggers a single action (taking one photo) and may not be duplicated due to a join with a relation or another stream. This semantics is not compatible with other DSMS like in [3, 26, 15, 8].

2. The optimizing criteria are implicit: in the above scenario, the goal of the query is to choose the camera with the least estimated

response time for each event, and cannot be declaratively modified to choose another criterion like, for example, the photo quality.

3. At the query processing level, logical and physical steps seem to be merge in a single step. This choice limits the opportunities for query optimization techniques.

4. Only limited support is provided for continuous query processing. Specific operators for streams, like windows over streams [12] or relation-to-stream operators [3], are not tackled, as well as joining several streams, relations and virtual device tables.

Expressing queries such as the night surveillance scenario requires a framework that remains compatible with standard continuous query processing, allowing to reuse the query optimization techniques of DSMS, and that integrates the notion of action like in Aorta.

1.2 Evolution of Continuous Queries

In this paper, we present an ongoing effort to develop a framework for *Action-Oriented Continuous Queries* (AOCQs), whose aim is to integrate services, *i.e.* distributed functionalities, in continuous queries over data streams. AOCQs allow the definition of queries combining standard relations, data streams and services using a homogeneous representation, in a declarative language extending SQL.

The first requirement to achieve this ambitious goal is to define a common framework to deal with non-conventional data sources. Relations and data streams can share the same representation as time-varying multisets of tuples like in [3]. We represent sets of similar services as virtual tables containing a tuple per service and associated with one or more binding patterns [14, 18, 22] indicating which virtual attributes correspond to input and output parameters of the service functions. We keep backward compatibility with standard DBMS as we use standard relations, while extending the power of expression of queries to handle the notion of time. Event flows are represented as data streams, in order to avoid the mismatch between events and standard data tuples.

AOCQs can imply services that are statically bound [18, 22] or dynamically discovered in the pervasive information system, like in [21]. The optimal services (at a given time for a given data set) are selected and called during query execution. AOCQs can then express an event management functionality like event filtering and composition, and perform cost-based optimized calls to services. In pervasive environments, those queries can use services of the pervasive system to execute actions: continuous queries can smoothly evolve from data-oriented queries to action-oriented queries.

In this setting, the main contributions of this paper are:

- An extension of SQL to homogeneously express operators over relations, data streams and services, and an associated query processing technique to handle time-variations of data and dynamic calls to services during execution. An additional SQL clause, called the COLLAPSE clause, is proposed to define an optimizing criterion over groups of tuples.

- The development of a prototype of a query processor for AOCQs, from which first experimental results over synthetic data are described. For the time being, the AOCQ processor is built on top of the STREAM

prototype [3], a DSMS developed at Stanford University, and allows to show both the power of expression of AOCQs and the capabilities of the query processor.

In Section 2, we situate our problem within the related works. In Section 3, we define a homogeneous representation for non-conventional data sources as virtual tables. We tackle query processing techniques for virtual tables and the COLLAPSE clause in Section 4. We describe our implementation prototype and discuss some experimental results in Section 5. We then conclude and discuss some open issues in Section 6.

2 Related Work

Data Streams In modern information systems, some data sources may generate continuous unbounded streams of data elements. For compatibility with the relational model, data streams are commonly modeled as an append-only multiset of timestamped tuples whereas relations are considered as time-varying multisets of tuples (creation, update, deletion) as in [3]. This widely adopted model [23, 26, 11, 1, 8, 15] allows to manage structured data streams along with relations.

Time is an important notion for data streams. Tuples have an order in the stream, which is often supposed to be the order of arrival, and are timestamped. Timestamps are also supposed to reference a shared system clock, otherwise a synchronization mechanism is required [5].

Various data sources may generate data streams: *e.g.* sensors, that ranges from physical sensors (light, temperature...) to logical sensors (network monitoring, applications...). However, some other data may be considered as streams: transfers of large tuple sets from distributed databases are equivalent to data streams [22], even if they are bounded streams.

Costly Data Sources Data streams are often seen as virtual relational tables, but the generation of the tuples depends on several factors that can make it slow or unsafe. Some tuple attributes may also be expensive (in term of time or resources) to be acquired.

Unsafe data sources like sensor networks may introduce some latency and disorder for the tuples. Synchronization among events from different streams may then be erroneous. [5] proposes a system to cope with these problems thanks to a stream conditioning mechanism that reorders and synchronizes tuples.

Some safe data sources or function evaluations may also be slow, like web services or sensed attributes. Introducing asynchronous calls to data sources and synchronization operators in query execution plans, like in [18], allows to process incomplete tuples until their costly attributes are required, which gives time to complete the asynchronous call and fill in the missing attribute values. [25] introduces a selection among possible candidates (devices offering the same service) based on their current state, to choose the optimal way of evaluating a function, here executing an action in a pervasive environment. Furthermore, group optimization allows to optimally distribute simultaneous function evaluations among the possible candidates.

Continuous Query Definition Continuous queries over data streams are based on the relational paradigm. Standard query operators on relations (Select, Project, Join, Aggregate...) are then used, but their semantics may be unclear or ambiguous. [3] identifies three cate-

gories of operators to work with streams and relations: relation-to-relation (standard operators), relation-to-stream, and stream-to-relation. Stream-to-stream operators are absent because they can be composed from other operators. A continuous query is a tree of operators with streams and/or relations as input, and a stream or a relation as output. Some systems [26, 1, 15] do not express the difference between operator categories, and work, in their semantics, only with data streams.

Selection and projection operators keep a clear semantics with data streams. Selection operators filter tuples based on the values of their attributes, and projection operators keep only a subset of the attributes of tuples. There is no difference between data streams and relational tables for these operators.

With continuous queries, data streams are supposed to be unbounded. However, aggregation operators for relations need to have a complete view of all tuples (*e.g.* the COUNT operator in SQL), which is impossible for unbounded streams. A mechanism of punctuations [12], indicating the end of a group of related tuples, is needed in order to allow the aggregation operator to output its resulting aggregated tuples, thus creating an aggregated stream. In [3], aggregation operators are seen as relation-to-relation operators: the transformation of the input stream into a relation is done by other operators, and the output is a time-varying relation.

Join operators face the same problems as aggregation operators. Unbounded tuple streams potentially require unbounded memory space in order to be joined, as every tuple should be stored to be compared with every tuple coming from the other stream. The sets of tuples should then be bounded. A window defines a bounded subset of tuples from a stream (it is the only stream-to-relation operator in [3]), based on time or on the number of tuples. Sliding windows [3, 12] have a fixed size and continuously move forward (*e.g.* the last 100 tuples, tuples within the last 5 minutes). Hopping windows [26] have a fixed size and move by hop, defining a range of interval (*e.g.* 5-minute window every 5 minutes). In [8], windows can be defined in a flexible way: the window upper and lower bound are defined separately (fixed, sliding or hopping), allowing various type of windows. [3] defines also a partitioned window as the union of windows over a partitioned stream based on attribute values (*e.g.* the last 5 tuples for every different ID). With windows, join operators handle bounded sets of tuples and traditional techniques can be applied. Although the output is intuitively thought as a stream, join operators are seen in [3] as relation-to-relation operators: the output is a time-varying relation.

Continuous queries can be expressed in a declarative language. Most of the articles [3, 26, 15, 8] propose an extension of SQL in order to work with both relational databases and data streams. Some articles [10] tackle continuous querying over distributed XML data sets and propose an extension of XML-QL. Others [1] are based on a box representation of operators, expressing queries as a flow of tuples. However, when working with the data stream semantics mixed with the relational paradigm, SQL tends to be widely adopted as a base for query language extensions. Data streams are represented using a relation schema, like for relational tables.

The traditional query structure (SELECT – FROM – WHERE) can still express selection, projection, join, even aggregation (GROUP BY – HAVING), except that the FROM clause contains references to streams. The main extension is the definition of windows for the streams. In some

articles [3, 15], window specifications are added in the FROM clause for each stream, defining the time-based or count-based size. Other extensions [26, 8] add a clause to express a global window for every stream of a query (only time-based).

As continuous queries may be running forever, a continuous query management system should allow an administrator to express when and how long a query should be activated. Some extensions propose commands (like SQL commands CREATE, DROP) to activate and deactivate queries [10, 15], whereas others [8, 25] integrate clauses in the language to express start time and expiration time.

Continuous Query Processing The long-running nature of continuous queries changes the definition of execution plans. An execution plan is composed of operators that may handle data streams, making the execution more dynamic than for standard queries. Two methods appear in the literature to cope with this dynamicity.

The first method is the construction of a global execution plan, like in [3, 15, 26, 1], which is an extension of a standard execution plan where input and output of operators are queues of tuples instead of relations. As several queries may be running simultaneously, the system can share common operators (on the same streams) among the different queries.

The second method is to dynamically distribute tuples to one of their possible next operators, each tuple creating its own execution plan. Operators (called Eddies [8]) are responsible of the choice of the destination for each tuple they have processed, depending on the dynamic state of other candidate operators.

The optimization process always depends on a chosen cost metric. In traditional DBMS, standard queries are often optimized based on the total execution time of the query. This cost metric is no longer available for continuous queries due to their long-running nature. Other cost metrics are then proposed in the literature. The processing time by tuple seems to be the natural extension of the previous cost metric adapted to data streams. Another approach is the bottleneck metric [22] that optimizes the throughput of the queries.

3 Dealing with Non-Conventional Data Sources

Non-conventional data sources are data sources that cannot be represented as tuples in standard relations, like in conventional databases. The transactional paradigm cannot be directly applied to a data management system that handles dynamic sources like data streams, or dynamically discovered services.

For the purpose of integrating non-conventional data sources in an augmented DBMS, we propose a homogeneous representation of relations, data streams and services with some renewed definitions for relations, streams, tables, and virtual tables. We keep the presentation rather informal, the basic notions being simple.

3.1 Relations and Data Streams

A *relation schema* is a name associated with a set of attributes. Each *attribute* has a name and a definition domain of atomic values. A *tuple* over a relation schema is an element of the Cartesian product of its attribute domains.

A *relation* over a relation schema is a multiset

of tuples. Tuples can be inserted in a relation, and later deleted from it.

A *stream* can be defined as a relation where tuples cannot be deleted, *i.e.* an append-only multiset of tuples. Tuples inserted in a stream are associated with their insertion date.

The following definition of a table is inspired by the work on data streams in [3] and the associated prototype. As data sources are dynamic, the notion of time needs to be explicit, in contrast with the transactional paradigm. Time is represented as a discrete and ordered domain of *timestamps* (*e.g.* positive integer values). Two events are considered simultaneous if they are both associated with the same timestamp.

In order to homogeneously represent a relation and a stream, we define a *table* over a relation schema as a multiset of tuples associated with their insertion timestamps. In other words, a table represents a relation where each tuple is associated with its insertion timestamp. A table represents a stream if no tuples can be deleted from the table. It can then homogeneously represent a relation or a stream.

We consider the *instantaneous relation* [3] of a table at a given timestamp as the multiset of tuples that have been inserted until this timestamp included, and that have not yet been deleted. Note that a tuple can be inserted and deleted simultaneously, *i.e.* at the same timestamp. For a table representing a stream, the number of tuples of its instantaneous relation may only grow, as no tuple can be deleted: a stream is unbounded.

EXAMPLE 1 *Tables for relations and streams*
Figure 3 and Figure 4 show two tables representing a relation "phone" and a stream "sensor". The instantaneous relations for both tables are represented at timestamp 25 and at timestamp 30. Note that at timestamp 30, the tuple "Bob" has been deleted from the "phone" table. Note also that several tuples can be inserted simultaneously, like at timestamp 27 in the "sensor" table.

```
TABLE phone( id INTEGER, owner CHAR(10),
               number CHAR(10))

Timestamp @ 25
(34,"Alice","069911XXXX") @ 10
(25,"Bob"  ,"069922XXXX") @ 12

Timestamp @ 30
(34,"Alice"  ,"069911XXXX") @ 10
(18,"Charlie","069933XXXX") @ 26
(24,"David"  ,"069944XXXX") @ 28
```

Figure 3: Schema and two instantaneous relations at different timestamps for the table representing the "phone" relation

```
TABLE sensor( id INTEGER, accel_x FLOAT,
              location BYTE)

Timestamp @ 25
(18, 362.15, 'a') @ 16
(65, 569.42, 'e') @ 25

Timestamp @ 30
(18, 362.15, 'a') @ 16
(65, 569.42, 'e') @ 25
(18, 236.78, 'a') @ 27
(17, 718.64, 'd') @ 27
(98, 624.16, 'c') @ 28
```

Figure 4: Schema and two instantaneous relations at different timestamps for the table representing the "sensor" stream

3.2 Services

A *service* is an external entity (in regard to the query management system) that can compute one or more functions. We define a *service interface* as a group of semantically related functions.

A function can have several input parameters (may be none) and several output parameters (at least one). When called with atomic values for its input parameters, a function returns zero, one or several result lines of atomic values, each line containing all output parameters.

EXAMPLE 2 *Service Interface*
Figure 5 shows the definition of a service interface providing three functions: checkCoverage() *that indicates if the service can take a photo of a given location,* checkCost() *that indicates the cost of taking this photo, and* takePhoto() *that actually takes it.*

```
SERVICE INTERFACE cameraInterface {
  FUNCTION checkCoverage( target BYTE ) :
            ( status BOOLEAN )
  FUNCTION checkCost( target BYTE ) :
            ( status FLOAT )
  FUNCTION takePhoto( target BYTE ) :
            ( result BLOB )
}
```

Figure 5: Example of Service Interface

To smoothly integrate services in our framework, we propose to use the notion of *binding patterns*. A *binding pattern* models an access pattern to a relational data source as a specification of "which attributes of a relation must be given values when accessing a set of tuples" [14]. A relation with binding patterns can represent an external data source with limited access patterns [14] in the context of data integration. It can also represent an interface to an infinite data source like a web site search engine [18], providing a list of URLs corresponding to some given keywords. In a more general way, it can represent a data service, *e.g.* web services providing data sets, as a virtual relational table like in [22].

In our framework, we propose to define a *virtual table* using a service interface as a generalization of a table: its schema can contain *virtual attributes* and is associated with *binding patterns* involving functions from the service interface. A *virtual attribute* is an attribute whose value is set during query execution, *i.e.* is not set when the tuple is retrieved from the data source. A *binding pattern* is a rule that indicates which function from the service interface has to be invoked in order to retrieve the values of some virtual attributes (the output parameters) when values are set for some other virtual attributes (the input parameters).

EXAMPLE 3 *Binding Patterns*
Figure 6 shows the definition of a virtual table "camera" and its associated binding patterns using the service interface cameraInterface *given in Example 2. The virtual table schema contains one non-virtual attribute* id *and four virtual attributes: when a value is given for the virtual attribute* location*, the three binding patterns can be invoked if needed to independently retrieve the values of the other virtual attributes* coverage*,* cost *and* photo*.*

```
VIRTUAL TABLE camera ( id INTEGER,
                       location BYTE VIRTUAL,
                       coverage BOOLEAN VIRTUAL,
                       cost FLOAT VIRTUAL,
                       photo BLOB VIRTUAL )

BINDING PATTERNS FOR camera
    USING cameraInterface {
  FUNCTION checkCoverage( location ) :
            ( coverage )
  FUNCTION checkCost( location ) : ( cost )
  FUNCTION takePhoto( location ) : ( photo )
}
```

Figure 6: Schema and binding patterns for the virtual table "camera"

A virtual table, like non-virtual tables, contains tuples. However, as those tuples contains

virtual attributes, we refer to them as *virtual tuples*. Each virtual tuple is bound to one service that implements the service interface used by the virtual table. During query execution, when a binding pattern is invoked for a virtual tuple, the required function is invoked from the service to which this virtual tuple is bound. Like tuples in a table, virtual tuples can be inserted in a virtual table, and deleted from it.

Example 4 *Virtual Tuples*
Continuing the previous example, Figure 7 shows instantaneous relations for the virtual table "camera", i.e. *the virtual tuples it contains, at timestamp 25 and 30. Only the non-virtual attribute* id *has a value. The '*' indicates that no value is set for the four virtual attributes* location*,* coverage*,* cost *and* photo*. Each virtual tuple is bound to a service, indicated by the service reference,* e.g. *'Camera2', 'Camera3'. Note that the tuple bound to the service 'Camera2' at timestamp 25 no longer belongs to the table at timestamp 30, because the service itself is no longer available in the pervasive environment.*

```
Timestamp @ 25
(2, *, *, *, *) # Camera2 @ 12
(3, *, *, *, *) # Camera3 @ 12
(5, *, *, *, *) # Camera5 @ 25

Timestamp @ 30
(3, *, *, *, *) # Camera3 @ 12
(5, *, *, *, *) # Camera5 @ 25
(8, *, *, *, *) # Camera8 @ 27
(6, *, *, *, *) # Camera6 @ 28
```

Figure 7: Two instantaneous relations at different timestamps for the virtual table "camera"

In other words, a virtual table represents a set of services providing the same functionalities, *i.e.* implementing the same service interface. Tuples can be dynamically inserted and deleted as such services are discovered in a pervasive environment. The services can also be manually added by a system developer. An extreme case is a virtual table containing one and only one *static virtual tuple*, *i.e.* a virtual tuple that cannot be deleted: the virtual table is then a simple interface to one statically bound service, or even one function, as it is used in previous works [14, 18, 22]. We call such a virtual table, a *static virtual table*, as opposed to the general case, a *dynamic virtual table.*

Example 5 *Environment for the Night Surveillance Scenario*
The environment for the night surveillance scenario is represented in Aorta [25] with a relation, a stream, a virtual device table and three functions (Figure 1). Using our framework, it can be represented in a homogeneous way with four tables.
Along with the "phone" and "sensor" tables defined in Example 1, and the "camera" virtual table defined in Example 3, one more table is required: a static virtual table "sendMMS", defined in Figure 8, representing a function that sends a MMS (Multimedia Message) to a cell phone. It is statically bound to a service from the environment implementing this function (not represented in the figure).

To end up, virtual tables generalize the notion of tables representing relations and streams. It can then be thought as a homogeneous representation for all data sources needed in a pervasive environment: relations, streams, static and dynamic virtual tables. Table 1 summarizes the constraints for each type of data sources.

System developers can work with a common representation of the different data sources available in their computing environment. More importantly, they can devise their queries involv-

Type of Data Source	Tuple Insertion	Tuple Deletion	Binding Patterns
Relation	yes	yes	no
Stream	yes	no	no
Static Virtual Table	no	no	yes
Dynamic Virtual Table	yes	yes	yes

Table 1: Summary of constraints for each type of data sources

```
SERVICE INTERFACE sendMmsInterface {
  FUNCTION send( message CHAR(255),
                 picture BLOB,
                 destination CHAR(10)) :
             ( status BOOLEAN )
}

VIRTUAL TABLE sendMMS( text CHAR(255) VIRTUAL,
                       image BLOB VIRTUAL,
                       phone_no CHAR(10) VIRTUAL,
                       result BOOLEAN VIRTUAL )

BINDING PATTERNS FOR sendMMS
    USING sendMmsInterface {
  FUNCTION send( text, image, phone_no ) :
             ( result )
}
```

Figure 8: Schema and binding patterns for the static virtual table "sendMMS"

ing different types of data sources using a single SQL-like declarative language, without worrying about the particular implementations of the data sources.

4 Query Processing for AOCQs

AOCQs are continuous queries over relations and data streams, with the addition of virtual tables for functions and services. Simple queries could be expressed using a SQL-like declarative language. CQL (Continuous Query Language [3]) provides syntax extensions to SQL in order to handle the specificities of data streams and to allow continuous queries.

As a query language for our framework, an extension of the semantics of CQL is required to include the notion of virtual tables and the associated processing techniques for virtual tuples.

However, the introduction of virtual tables raises the need to define a new functionality: expressing optimization criteria to choose the optimal tuple(s) among a group of possibilities. We need to choose the optimal virtual tuple corresponding to an event so that only the "optimal" service is actually invoked. We present a solution to this need through a new clause in SQL: the COLLAPSE clause.

EXAMPLE 6

For the night surveillance scenario, we need to handle events, represented as tuples in the "sensor" table. In order to take a photo of the event location, those tuples have to be associated with a "camera" service, represented as tuples in the "camera" virtual table. More than one service may be able to take the photo. However, only one photo is needed: the system should select the "optimal" service, i.e. *the service with the least estimated response time. The definition of "optimal" is context-dependent: it justifies the introduction, at the declarative level, of a new clause in SQL.*

4.1 Continuous Query Processing with Virtual Tables

4.1.1 Taking into account Virtual Tables

All data sources are represented as virtual tables associated with binding patterns. Non-virtual tables are only extreme cases with zero binding patterns. In a logical query plan, intermediary tables between operators are also virtual tables as well as the output table of the root operator.

After a query is parsed, its semantics is checked using the metadata catalog referencing the names and properties for tables and attributes. It is then transformed into a logical query plan of operators like joins, selections, projections, aggregations.

The metadata catalog also contains the binding patterns associated with virtual tables. A specific operator, the dependent join [14], is required to realize a binding pattern: it provides values for the binding pattern input attributes (by an equality predicate with another attribute or a constant value) and allows to retrieve the values for the binding pattern output attributes. Binding patterns add constraints on the join order for the tables: a dependent join operator should have values for its input attributes, so other dependent joins that retrieve those values (as the output attributes of their binding patterns) should occur before.

A dependent join operator produces an output table containing virtual tuples with values for the binding pattern input attributes. However, it is not already necessary to invoke the service function associated with the binding pattern to retrieve the output attribute values. On the contrary, it is interesting to keep the tuples as long as possible in a virtual form (with no values for the output attributes), in order to make asynchronous calls [18] to the functions and speed up the global query processing.

Two additional logical operators need to be integrated in the operator tree for each required binding pattern. An *invocation operator* makes asynchronous calls to the function associated with the binding pattern, and a *binding operator* actually sets the requested values into the tuple attributes. Note that the invocation operator is not blocking for the tuple whereas the binding operator can block a tuple as long as the asynchronous call has not returned its result lines. The blocking operator ensures that the virtual attributes involved in the binding pattern have their actual values for every output tuple it produces. In [18], the binding operator (called "Request Synchronizer") is present but the invocation operator is integrated in the table scan operator for the data source. The independence of the invocation operator allows a more flexible query plan and leads to further optimization possibilities.

Query optimizations techniques can be applied on the logical query plan. Operators can be reorganized in order to minimize the number and size of tuples to process, *e.g.* by pushing selection operators down before joins or introducing projections. The number of function calls can also be minimized, *e.g.* by pushing selection operators down before invocation operators. Further optimization techniques can be applied to the physical representation of the query plan, like merging some operators, in order to compute an optimal physical query plan. For this step, we rely on well-known logical optimization techniques and do not propose new ones.

4.1.2 Continuous Query Execution

In the execution phase, the query processor actually executes the physical query plan. Whereas in traditional DBMS, the query processor executes a query plan once to produce a resulting table, the continuous query processor needs to schedule each operators in (near) real-time, in order to process new tuples from the data streams and insertions/deletions of tuples from the relations, and to propagate them through the operator tree. [3] studies some scheduling algorithms for this context.

In order to realize the binding patterns, the virtual tuple processing technique follows the same principle as the *asynchronous iteration* technique in [18]. When processed by a *binding operator*, an input virtual tuple may be duplicated according to the number of result lines for the corresponding function call: each result line will produce one output tuple. Every output tuple contains a copy of all the attribute values from the input virtual tuple, including the input attributes of the binding pattern. It also contains the values for the output attributes of the binding pattern that are retrieved from the result line. The output tuples are virtual in the general case: the output table of the operator may still contain some binding patterns for other virtual attributes.

Example 7 and Example 8 demonstrate two AOCQs, one involving a static virtual table and one involving a dynamic virtual table.

EXAMPLE 7 *Using a Static Virtual Table*
In Figure 9, an AOCQ expressed in CQL [3] allows to define the following behavior: for each phone, send a MMS containing a "Hello (name) !" message and a "welcome.jpg" image (interpreted as a BLOB constant in the query). The query uses the "phone" table defined in Example 1 and the static virtual table "sendMMS" defined in Figure 8. As the query is continuous, all current and future phones inserted in the "phone" relation will receive a MMS. Note that the tuple corresponding to "Bob" does not belong to the resulting table at timestamp 30 because it is deleted from the "phone" table (see Figure 3). However, the corresponding call to the service function happens at timestamp 12 (when the tuple is inserted in the "phone" table). It is possible to keep a trace of that tuple by requesting the resulting stream of inserted tuples instead of the resulting relation (like with the ISTREAM *keyword in CQL [3]).*

```
SELECT phone.owner, phone.number, sendMMS.result
FROM phone, sendMMS
WHERE phone.number = sendMMS.phone_no
  AND sendMMS.image = BLOB("welcome.jpg")
  AND sendMMS.text = "Hello "||phone.owner||" !"

Resulting Table:

Timestamp @ 25
("Alice"  ,"069911XXXX",true) @ 10
("Bob"    ,"069922XXXX",true) @ 12

Timestamp @ 30
("Alice"  ,"069911XXXX",true) @ 10
("Charlie","069933XXXX",true) @ 26
("David"  ,"069944XXXX",true) @ 28

List of Function Calls:

sendMMS ("Hello Alice !", BLOB("welcome.jpg"),
          "069911XXXX") : (true) @ 10
sendMMS ("Hello Bob !", BLOB("welcome.jpg"),
          "069922XXXX") : (true) @ 12
sendMMS ("Hello Charlie !", BLOB("welcome.jpg"),
          "069933XXXX") : (true) @ 26
sendMMS ("Hello David !", BLOB("welcome.jpg"),
          "069944XXXX") : (true) @ 28
```

Figure 9: Example of a query using the static virtual table "sendMMS"

EXAMPLE 8 *Using a Dynamic Virtual Table*
In Figure 10, an AOCQ allows to handle events from the "sensor" stream (see Figure 4): each tuple that has a 'accel_x' value greater than 500 is associated with every service from the virtual table "camera" (defined in Example 3 and 4) that covers its location. This coverage is indicated by the boolean virtual attribute 'coverage'. The virtual attribute 'photo' represents an actual photo provided by the service. As the result table is a join between a stream and a virtual table, no result tuple can be deleted: the result table is itself a stream.

```
SELECT sensor.id, sensor.location,
       camera.id, camera.photo
FROM sensor, camera
WHERE sensor.accel_x > 500.0
  AND sensor.location = camera.location
  AND camera.coverage

Result (stream):

Timestamp @ 25
(65, 'e', 2, BLOB("photo001.jpg")) @ 25
(65, 'e', 3, BLOB("photo002.jpg")) @ 25

Timestamp @ 30
(65, 'e', 2, BLOB("photo001.jpg")) @ 25
(65, 'e', 3 ,BLOB("photo002.jpg")) @ 25
(17, 'd', 3, BLOB("photo003.jpg")) @ 27
(17, 'd', 5, BLOB("photo004.jpg")) @ 27
(17, 'd', 8, BLOB("photo005.jpg")) @ 27
(98, 'c', 5, BLOB("photo006.jpg")) @ 28
```

Figure 10: Example of a query using the virtual table "camera"

4.2 The COLLAPSE Clause

Virtual tables provide a mean to represent services that are dynamically discovered in a pervasive environment. In Example 8, each tuple from the "sensor" stream is joined with every tuple from the "camera" virtual table, *i.e.* all available services. Even if a condition on the coverage allows to discard some tuples, the result table may contain several tuples corresponding to one event: with the binding patterns, the system has to invoke the *takePhoto()* function for several services. Although this behavior may be wanted, the goal of the night surveillance scenario is to choose the best way to handle each event, *i.e.* to call only the best service to handle an event. With the "camera" virtual table, the best service for a given location is the one with the minimum value for the 'cost' virtual attribute.

It could be expressed using a nested query as in Figure 11. However, nested queries are not satisfying for this goal as it complexifies the query design. The optimizing criterion is not well identified and may still select several tuples in case of equality.

```
SELECT sensor.id, sensor.location, camera.photo
FROM sensor, camera
WHERE sensor.accel_x > 500.0
  AND sensor.location = camera.location
  AND camera.coverage
  AND camera.cost =
  ( SELECT MIN(camera.cost)
    FROM camera
    WHERE camera.location = sensor.location
      AND camera.coverage )
```

Figure 11: Query that selects the best service using a nested query

AOCQs may need to explicitly express criteria to choose the optimal service for each event. From a data-centric point of view, the goal is to extract the first tuple from a group of tuples according to a given ordering. On the one hand, it is similar to the definition of a top-K query (here with K=1) applied to sub-groups of tuples. On the other hand, computing one tuple from a

group of tuples is similar to an aggregation.

However, standard aggregation functions like MIN, MAX or AVG, accept only one parameter and return only one value. Some DBMS like PostgreSQL allow to define User Defined Aggregates (UDAs) that accept several parameters, but still return one value. Even if the return value may be composite, *i.e.* a structure composed of several attributes, it does not allow a simple syntax to express the required optimization. Furthermore, it requires the development of a new UDA adapted to the type and number of involved attributes for each query. Three functions are required for a UDA: an initialization function that initializes the aggregate state with the first tuple of the group, an iteration function that updates the aggregate state for each following tuple, and a finalization function that returns the aggregated value computed from the aggregate state. UDA function are developed using DBMS-specific language, with a non-declarative approach: query optimization opportunities are thus reduced for the query processor.

EXAMPLE 9 *Using a User-Defined Aggregate over several Attributes*
Figure 12 shows a possible query syntax using such a UDA: the UDA_MIN aggregation function works only for three attributes and returns a composite value containing this three attributes, retrieved from the tuple that minimizes the first attribute. This syntax is ambiguous as it does not show the composite nature of the function output.

In this setting, we propose a new clause for SQL in order to express such an aggregate in a generic and unambiguous way: the COLLAPSE clause. It allows to define an aggregate function returning several attributes that are retrieved from the optimal tuple for each group. Figure 13 shows the syntax of the COLLAPSE clause. It has to immediately follow the GROUP BY clause.

```
SELECT s.id, s.location,
       UDA_MIN(c.cost, c.id, c.photo)
FROM sensor s, camera c
WHERE s.location = c.location
  AND c.coverage
GROUP BY s.id, s.location
```

Figure 12: Example of a query using a User-Defined Aggregate UDA_MIN over three attributes

```
GROUP BY groupAtt1, groupAtt2, ...
COLLAPSE (att1,att2,...,attN) INTO name
  USING orderAtt1 [ASC|DESC],
        orderAtt2 [ASC|DESC],
        ...
```

Figure 13: Syntax of the COLLAPSE Clause

The set of attributes ('att1','att2',...,'attN') are the *collapsed attributes* returned by the aggregate function. The optimal tuple corresponds to the first tuple of the group when it is ordered according to the USING part (like with an ORDER BY clause in SQL). The INTO part defines the name for the set of collapsed attributes, so that they can be referenced as 'name.attribute' in the SELECT clause and/or the HAVING clause. Collapsed attributes can thus be used like other standard aggregate values in these both clauses.

EXAMPLE 10 *Using a* COLLAPSE *clause*
In Figure 14, a COLLAPSE *clause extracts for each group ('s.id','s.location') the tuple that minimizes the 'c.cost' value,* i.e. *the first tuple in each group ordered by the 'c.cost' value in ascending order. The name of this collapsed set is 'bestCamera': the collapsed attributes are identified by 'bestCamera.cost' and 'bestCamera.photo' in the* SELECT *clause and in the* HAVING *clause.*

A collapsed attribute set can be defined as an

```
SELECT s.id, s.location,
       bestCamera.cost, bestCamera.photo
FROM sensor s, camera c
WHERE s.location = c.location
  AND c.coverage
GROUP BY s.id, s.location
COLLAPSE (c.cost, c.photo) INTO bestCamera
  USING c.cost ASC
HAVING bestCamera.cost < 5
```

Figure 14: Example of a query using a COLLAPSE clause

implicit table whose schema contains the grouping attributes and the collapsed attributes. The *implicit table* contains the collapsed tuples for all groups. The query result is then a join between this *implicit table* and the other tables based on the equality between the grouping attributes. This definition allows a generalization of the COLLAPSE clause: the *implicit table* can contain more than one optimal tuple for a group, as in top-K queries. A query can specify the maximum number K of collapsed tuples for a group. The integration of ties, *i.e.* tuples with the same order level, in the collapsed result may be specified with a '+' mark after the number, indicating that more than K tuples can be integrated if they are ties with the K^{th} tuple. The default behavior is a collapsed result of strictly one tuple.

A special case is to use the Pareto optimality to express a multi-objective query [4]: the optimizing parameters are not an ordered list, but a set. The keyword PARETO replaces the maximum number of tuples in the collapsed result, as all Pareto-optimal tuples are integrated.

EXAMPLE 11 *Different forms of the* COLLAPSE *clause*
In Figure 15 (using the "camera" virtual table with an additional 'quality' attribute), the first COLLAPSE *clause is the default case: for one group, the collapsed set contains only one tuple that maximizes a quality attribute and, in the case of equality for the first criterion, minimizes the cost attribute. The second clause extracts the three least expensive camera, with the best quality in case of equality. The third one uses the same criteria to extract at least two cameras, but may also include cameras that have the same cost and quality as the second best one. The last clause extracts the Pareto-optimal tuples,* i.e. *the one with the best cost and the one with the best quality: as it may be the same tuple, the collapsed set may contain only one tuple or two tuples. Note that in the last clause, the order of the ordering attributes is not relevant.*

```
COLLAPSE (c.cost,c.photo) INTO bestCamera
  USING c.quality DESC, c.cost ASC
COLLAPSE (c.cost,c.photo) INTO bestCamera[3]
  USING c.cost ASC, c.quality DESC
COLLAPSE (c.cost,c.photo) INTO bestCamera[2+]
  USING c.cost ASC, c.quality DESC
COLLAPSE (c.cost,c.photo) INTO bestCamera[PARETO]
  USING quality DESC, c.cost ASC
```

Figure 15: Example of COLLAPSE clauses using two ordering attributes

Although we present it in the context of AOCQs to choose the optimal service(s) to be called for a given event, this clause can be applied to other cases, in particular in non-continuous query, *e.g.* in multi-objective query processing [4] or to declaratively define complex aggregations like in [9, 2].

5 Implementation

Continuous query processing techniques are inspired from standard query processing techniques [16]. However, the introduction of the notion of time impacts on the whole conception.

We propose an architecture of an AOCQ-enabled DSMS. We choose to build our AOCQ processor prototype on top of an open-source DSMS: STREAM [3], whose prototype is developed at Stanford University. We explain the integration of additional functionalities in order to handle some AOCQ concepts, and describe first experimental results from our prototype.

5.1 AOCQ Processor Architecture

The architecture of the AOCQ processor is composed of six main modules, as shown in Figure 16. Query analysis is performed first by the query parser, and then by the query optimizer. The query optimizer checks the query semantics with the metadata catalog, and produces a continuous query evaluation plan. The query plan manager is responsible for the simultaneous execution of all produced plans: it schedules the different query operators in a (near) real-time fashion. The data source manager provides access to the data sources and handles function calls via the service interface manager. This architecture is obviously compliant with the STREAM architecture described in the next section.

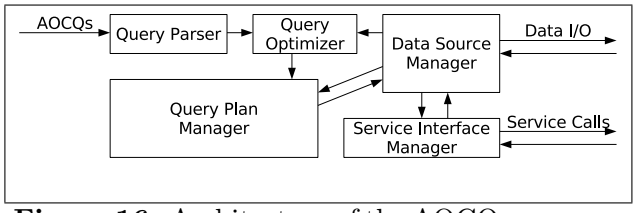


Figure 16: Architecture of the AOCQ processor

5.2 The STREAM Prototype

STREAM provides support for "a large class of declarative continuous queries over continuous streams and traditional stored data sets" [3]. It is composed of a CQL parser, a query analyzer that produces execution plans, and a plan manager that schedules operators to execute the continuous queries. Execution plans are optimized at the logical level, then at the physical level. The prototype allows to register relations and streams schemas, and to associate them with a physical data source. A physical data source is an interface (in C++) that is currently implemented as a file reader for both relations and streams. Support for four data types is provided: byte, integer, float, and fixed-length string.

In the current implementation, CQL allows to define queries similar to SQL: SELECT – FROM – WHERE – GROUP BY. The FROM clause is extended to define windows over the streams. The relation-to-stream operators (IStream, DStream, RStream) are expressed by a keyword with parenthesis surrounding the whole query text. Aggregation functions are limited to the MIN, MAX and AVG functions over integer and float attributes.

5.3 Implementation of new Operators

In order to handle AOCQs, we extend the STREAM prototype to integrate the COLLAPSE clause and the notion of binding patterns. For the time being, only limited support for binding patterns has been integrated in the implementation.

The COLLAPSE clause is integrated as a sort of polymorphic aggregation function: it can accept any type and number of input parameters, and its output parameters follow the same schema as the input parameters. We limit the COLLAPSE clause to the default case returning only one tuple, and with only one ordering attribute. Its implementation implies a modification of the analysis of the SELECT clause, and some impacts

on the execution of the aggregation operator in query plans.

5.4 Experimentation

We choose to experiment the night surveillance scenario described throughout the paper. The AOCQ is represented in Figure 17. Two tables and two virtual tables have been defined to represent the environment (cf. Example 5). The window specification '[now]' indicates that a tuple from the "sensor" table will not be joined with tuples inserted at a later timestamp in other tables.

```
SELECT s.TIMESTAMP, s.id, p.id
       best.id, best.cost, best.photo,
       best.result
FROM sensor s [now], camera c, phone p, sendMMS
WHERE s.accel_x > 500
  AND s.location = c.location
  AND c.coverage
  AND sendMMS.image = camera.photo
  AND sendMMS.phone_no = p.number
GROUP BY s.TIMESTAMP, s.id, p.id
COLLAPSE (c.cost, c.id, c.photo,
          sendMMS.result) INTO best
  USING c.cost ASC
```

Figure 17: AOCQ for the night surveillance scenario

In order to test the query, test data have been generated for the two tables "sensor" and "phone": 10000 random tuples in "sensor" with a timestamp between 1 and 9999 indicating a 'accel_x' value between 300 and 900 and a location label between 26 possibilities ('a' to 'z'), and 6 tuples in "phone" representing 6 administrators receiving the photos.

50 cameras have been simulated in the virtual table "camera": for each camera, the virtual tuple is represented by one tuple for each location, so that the predicate 's.location = c.location' will select one tuple, and with a random value for the 'coverage' boolean (a location is covered by at least one camera, one camera covers around 20% of the locations) and the 'cost' value. The virtual table "sendMMS" contains one tuple: to simulate the virtual tuple, we simply discard the predicates related to this table.

In order to monitor more closely the query in Figure 17, it is divided into one sub-query joining the tables "sensor" and "camera", producing a stream of tuples representing all possibilities to handle the events, and one main query selecting the optimal tuple by joining the previous stream with the tables "phone" and "sendMMS", and by applying the COLLAPSE clause.

As a result, the sub-query generates a stream of around 76000 tuples. Without the COLLAPSE clause, the main query result set contains more than 450000 tuples, whereas the COLLAPSE clause reduces this number to around 45000, *i.e.* by a factor of 10.

Along with the predictable saving in the number of tuples, this example shows the power of expression of AOCQs: the optimizing criteria being explicitly expressed as the 'cost' attribute of the camera, it can be declaratively changed in the query definition thanks to the COLLAPSE clause. Additional experiments have been scheduled to assess the validity of our prototype.

6 Conclusion

In this paper, we have presented our framework for *Action-Oriented Continuous Queries* (AOCQs) that allows to build queries over relations, streams and services. It is built on top of the CQL specifications [3] that manage streams and relations.

The AOCQ framework introduces tables and

virtual tables as a unified mean to represent relations, streams and services. A virtual table has virtual attributes and is related to a service interface, using binding patterns to indicate which virtual attributes should be used as an input for a service function call or retrieved as an output from a service function call. At the query plan level, a dependent join operator realizes a binding pattern. During query execution, an invocation operator makes asynchronous calls to functions in a non-blocking manner, and a binding operator is used to block until the data are effectively retrieved from the function call. The underlying principle of virtual tables can be used as a mean to take in charge the dynamicity of pervasive environments where services appear and disappear.

Many services may be able to provide a virtual attribute value for a specific query. We have thus introduced the COLLAPSE clause that declaratively defines a criterion for the selection of a sub-set of service function calls. The COLLAPSE clause builds an implicit table that contains the top-K tuples from a group of tuples according to a given ordering. The COLLAPSE clause intends to replace and augment the procedural and ad hoc user-defined aggregates that are available today in DBMS.

We have also presented first implementation and experimentation results of the COLLAPSE clause on top of the STREAM prototype [3]. For the time being, the prototype includes a mechanism to identify virtual attributes so as to insert dependent joins, invocation operators and binding operators in the execution plan of queries.

This first implementation and experimentation has presented the COLLAPSE clause used in the running example of our article. In future work, we plan to implement invocation and binding operators within STREAM and to develop a benchmark on real data sets and real services.

References

[1] D. J. Abadi et al. The Design of the Borealis Stream Processing Engine. In *CIDR 2005, Proceedings of Second Biennial Conference on Innovative Data Systems Research*, 2005.

[2] M. O. Akinde, D. Chatziantoniou, T. Johnson, and S. Kim. The MD-Join: An Operator for Complex OLAP. In *ICDE'01: Proceedings of the 17th International Conference on Data Engineering*, page 524, 2001.

[3] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom. STREAM: The Stanford Stream Data Manager. *IEEE Data Engineering Bulletin*, 26(1):19–26, 2003.

[4] W.-T. Balke and U. Güntzer. Multi-objective Query Processing for Database Systems. In *VLDB'2004: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 936–947, 2004.

[5] R. S. Barga and G. Chkodrov. Coping with Variable Latency and Disorder in Distributed Event Streams. In *ICDCSW'06, Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops*, 2006.

[6] C. Becker, M. Handte, G. Schiele, and K. Rothermel. PCOM – A Component System for Pervasive Computing. In *PerCom'04, Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*, page 67, 2004.

[7] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. EasyLiving: Technologies for intelligent environments. In *HUC 2000, Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing*, pages 12–29, 2000.

[8] S. Chandrasekaran et al. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *CIDR 2003, Proceedings of the First*

Biennial Conference on Innovative Data Systems Research, 2003.

[9] D. Chatziantoniou. The PanQ tool and EMF SQL for Complex Data Management. In *KDD'99: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–424, 1999.

[10] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 379–390, 2000.

[11] M. Cherniack et al. Scalable Distributed Stream Processing. In *CIDR 2003, Proceedings of the First Biennial Conference on Innovative Data Systems Research*, 2003.

[12] L. Ding and E. A. Rundensteiner. Evaluating Window Joins over Punctuated Streams. In *CIKM'04, Proceedings of the 13th ACM international conference on Information and Knowledge Management*, pages 98–107, 2004.

[13] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the Physical World with Pervasive Networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[14] D. Florescu, A. Levy, I. Manolescu, and D. Suciu. Query Optimization in the Presence of Limited Access Patterns. In *SIGMOD'99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 311–322, 1999.

[15] M. J. Franklin et al. Design Considerations for High Fan-In Systems: The HiFi Approach. In *CIDR 2005, Proceedings of Second Biennial Conference on Innovative Data Systems Research*, 2005.

[16] H. Garcia-Molina, J. Widom, and J. D. Ullman. *Database System Implementation.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[17] D. Garlan et al. Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.

[18] R. Goldman and J. Widom. WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 285–296, 2000.

[19] R. Grimm et al. System Support for Pervasive Applications. *ACM Transactions on Computer Systems*, 22(4):421–486, November 2004.

[20] K. Henricksen and J. Indulska. A Software Engineering Framework for Context-Aware Pervasive Computing. In *PerCom'04, Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*, page 77, 2004.

[21] C.-E. Pigeot, Y. Gripay, M. Scuturici, and J.-M. Pierson. Context-Sensitive Security Framework for Pervasive Environments. In *ECUMN'07: Fourth European Conference on Universal Multiservice Networks*, pages 391–400, 2007.

[22] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query Optimization over Web Services. In *VLDB 2006, Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 355–366, 2006.

[23] F. Tian and D. J. DeWitt. Tuple Routing Strategies for Distributed Eddies. In *VLDB 2003, Proceedings of the 29th International Conference on Very Large Data Bases*, pages 333–344, 2003.

[24] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1991.

[25] W. Xue and Q. Luo. Action-Oriented Query Processing for Pervasive Computing. In *CIDR 2005, Proceedings of the Second Biennial Conference on Innovative Data Systems Research*, 2005.

[26] Y. Yao and J. Gehrke. Query Processing in Sensor Networks. In *CIDR 2003, Proceedings of the First Biennial Conference on Innovative Data Systems Research*, 2003.

[27] F. Zhu, M. Mutka, and L. Ni. Service Discovery in Pervasive Computing Environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.

Résumé

L'apparition des terminaux mobiles intelligents a fait évoluer considérablement le rapport de l'humain à l'outil informatique. Le terminal mobile est considéré comme un nouveau composant de l'architecture des systèmes d'information, et les nouvelles conditions d'usage induites forment le socle des systèmes d'information pervasifs. L'objectif premier d'un système pervasif est de permettre de répondre au besoin de l'utilisateur où qu'il se trouve et à n'importe quel moment. Les maîtres mots des systèmes pervasifs sont : sensibilité au contexte, smartness, scalabilité, invisibilité et pro-action. Tous ces mots-clés sous-tendent des procédures d'adaptation. La sensibilité au contexte permet de fournir les informations impulsant l'adaptation. Smartness et invisibilité sont des contraintes à appliquer entre autres à l'adaptation. La pro-action a pour but de prévoir les services et données qui seront nécessaires ; c'est une forme adaptation prévisionnelle. Nos travaux concernent principalement l'accès adaptatif aux informations.

Dans un premier temps, nous nous sommes intéressés aux documents comme paradigme d'interaction avec l'utilisateur. Cette approche nous a permis de ne plus restreindre le document à un seul outil de consultation de l'information, mais de l'étendre selon trois directions : saisie souple d'informations sous forme semi-libre, recherche de documents, et partage de documents. La saisie et le partage de documents ont été associés à des principes d'extraction et de structuration de données pertinentes, permettant une recherche d'informations à la SQL. La saisie et le partage de documents ont été associés à des principes d'extraction et de structuration de données pertinentes, permettant une recherche d'informations à la SQL.

Dans un second temps, nous avons travaillé plus spécifiquement sur l'accès adaptatif aux données dans des systèmes pervasifs. Nous avons abordé l'adaptation d'interfaces graphiques à des terminaux multiples et la personnalisation des interfaces. Nous avons ensuite étudié l'adaptation dans le cadre plus général de la sensibilité au contexte, en proposant une méthode et une plate-forme pour l'adaptation d'applications legacy sur leurs dimensions interfaces graphiques, données et services.

Ce mémoire résume mon activité de recherche sur l'adaptation et les systèmes pervasifs durant une dizaine d'années. Il couvre le travail de plusieurs doctorants et de nombreux étudiants de master.