E-CARe : une méthode d'ingénierie des systèmes d'information ubiquitaires

Ansem Ben Cheikh Ben Cheikh

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

« Ansem Ben Cheikh »

Thèse dirigée par **« Jean-Pierre GIRAUDIN »** et
codirigée par **« Agnès FRONT»** et **« Stéphane COULONDRE »**

préparée au sein du **Laboratoire Informatique de Grenoble**
dans **l'École Doctorale Mathématique, Sciences et Technologies de l'Information, Informatique**

E-CARe : une méthode d'ingénierie des Systèmes d'Information ubiquitaires

Thèse soutenue publiquement le **« 4 Juin 2012 »**,
devant le jury composé de :

Mme Corine CAUVET
Professeur des universités, Université Aix-Marseille, Rapporteur
Mr Philippe ANIORTE
Professeur des universités, Université de Pau et des Pays de l'Adour, Rapporteur
Mme Danielle BOULANGER
Professeur émérite, Université de Lyon 3, Examinatrice et présidente de jury
Mlle Jolita RALYTE
Maître d'Enseignement et de Recherche, Université de Genève, Examinatrice
Jean-Pierre GIRAUDIN
Professeur des universités, Université Pierre Mendès France, Directeur de thèse
Mme Agnès FRONT
MCF HDR, Université Pierre Mendès France, Codirectrice de thèse
Mr Stéphane COULONDRE
MCF, INSA de Lyon, Codirecteur de thèse

À mon mari,
À mes enfants,
À mes parents,
À mes frères,
À ma sœur.

J'espère que vous vous réjouissez de cette réussite qui est le résultat de longues heures de travail et de grands sacrifices. Merci pour votre amour et avant tout merci au grand Dieu. Hamdoulellah.

Remerciements[1]

Ma thèse de doctorat a constitué un long parcours riche en rencontres et marqué par l'apport scientifique et moral de plusieurs personnes que je tiens à remercier vivement.

Tout d'abord, je tiens à remercier mes directeurs de thèse : Agnès Front, Jean-Pierre Giraudin et Stéphane Coulondre.

Agnès Front, tu m'a encadrée depuis mon M2R et tu as toujours été présente pour m'orienter et me guider en permanence. Tu as partagé ton savoir-faire et tes connaissances pour que je réussisse cette thèse et tu as toujours été présente dans les moments les plus difficiles et j'en suis très reconnaissante.

Jean-Pierre Giraudin, je suis très reconnaissante pour tes conseils et remarques constructifs qui ont permis de bien mener ce travail. Ton expérience et ta vision analytique ont beaucoup influencé cette thèse. Je suis heureuse d'avoir rencontré une personne aussi droite et honnête.

Stéphane Coulondre, ta contribution dans ce travail a permis d'orienter ce travail et de l'enrichir avec une vision nouvelle et originale. Je suis très reconnaissante à l'effort et le temps fournis de ta part pour être présent malgré l'éloignement géographique.

Je tiens à remercier également les membres du jury qui m'ont fait l'honneur de participer à ma soutenance de thèse. Je remercie mes deux rapporteurs M. Philippe Aniorté et Mme Corine Cauvet pour avoir accepté de lire et d'évaluer mon travail à travers ce manuscrit. Je remercie Mme Danielle Boulanger et Mlle Jolita Ralyté d'avoir accepté d'examiner ce travail.

Je remercie vivement tous les membres de l'équipe SIGMA avec qui j'ai échangé. Chacun a contribué à sa façon pour rendre ce travail aussi agréable et constructif. Merci à notre chef Christine Verdier qui est très dynamique et enthousiaste, et qui dirige cette équipe dans une ambiance agréable et conviviale.

Je remercie Dominique Rieu qui m'a encadrée durant mon M2R, qui m'a transférée un amour de ce domaine de recherche et avec qui j'ai eu une expérience très riche et intéressante.

Je souhaite remercier Madame Nadine Mandran, ingénieure à Marvelig, de m'avoir aidée et guidée pour la réalisation des expérimentations. Elle est une personne enthousiaste, rigoureuse et ouverte qui a contribué à la valorisation et la promotion de ce travail. Je tiens à remercier

1. Cette thèse a été financée dans le cadre du projet DéSIT par le cluster TTS de la région Rhône-Alpes.

également les sujets du focus group : Christine, Luz Maria, Marco, Lucinéa, Amin, Mario, Rémi, Alexandre, Alfonso, Kiev et Sébastien.

Merci à Rajaa Saidi, Luz-Maria Priego, Marco Santorum et Jorgé-Luis Medina mes co-équipiers de bureau avec qui j'ai eu des discussions longues et intéressantes et avec qui j'ai partagé des moments de joie et de stress.

Merci et bon courage aux futurs docteurs de SIGMA : Amin, Aurélien, Mario, Juan Pablo, Loic et Nicolas.

A tous les permanents de l'équipe SIGMA (Dominique, Christine, Claudia, Sophie et Cyril) merci pour vos remarques et conseils constructifs qui ont permis de promouvoir ce travail.

Merci aux membres du laboratoire de psychologie Laboratoire Interuniversitaire de Psychologie-Personnalité, Cognition, Changement Social - Grenoble (LIP / PC2S) pour leur participation à l'avancement du projet DéSIT.

Je remercie également les personnes qui m'ont accompagnée dans la réalisation des animations de la fête des sciences à Lyon : Corinne Sainte-Colombe, Laina Ngom et Julie Drouet.

Un grand merci à vous tous.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte

Depuis toujours, l'informatique ne cesse d'évoluer et nous sommes passés du stade de l'ordinateur central associé à plusieurs individus au stade la multiplication des ordinateurs pour une même personne. Les ordinateurs sont devenus plus petits et plus performants. Ils sont aussi embarqués dans les objets de la vie quotidienne, il s'agit de l'informatique ambiante (intégrée dans les objets ambiants enrichis de capacités de traitement de l'information). Grâce aux technologies sans fils, les objets sont interconnectés et communiquent pour tout échange d'informations. Nous parlons dans ce cas d'informatique ubiquitaire où l'information est accessible de n'importe où et n'importe quand.

Mark Weiser a déclaré en 1991 dans son livre « The computer for the 21st century » [Weiser, 1991], que les technologies les plus réussies sont celles qui disparaissent. En effet, elles s'associent à la vie de tous les jours jusqu'à ce qu'il devienne difficile de les discerner. Depuis, un développement significatif a marqué l'informatique grâce aux avancées perçues dans les techniques de communication, les communications sans fils et les capteurs. Ainsi, la vision de Weiser a prouvé son exactitude surtout avec l'apparition de l'informatique ubiquitaire ou pervasive. L'objectif des systèmes ubiquitaires est d'accéder à tout moment, de tout endroit et avec tout média à toute information et à tout service. Ainsi les objectifs des systèmes informatiques récents ne sont plus restreints à l'exécution de tâches commandées par l'utilisateur mais à faire communiquer plusieurs systèmes mobiles ou fixes pour fournir des services personnalisés à un seul usager.

En complément à l'informatique ubiquitaire, l'informatique mobile permet à l'utilisateur de gérer l'information depuis un terminal mobile. Un système mobile est en fait un système distribué doté de capacités de gestion de la mobilité. Pour assumer toutes ces fonctionnalités intelligentes de traitement autonome des informations ou de communication, les dispositifs mobiles doivent être dotés d'outils logiciels permettant d'exploiter les informations géospatiales dans le cadre d'activités ayant trait aux déplacements.

Parallèlement aux systèmes informatiques, les SI doivent s'adapter et supporter cette nouvelle vision. L'objectif d'un SI est la production, la collection, le traitement, l'enregistrement et la diffusion de l'information. Mais les SI d'aujourd'hui affrontent le défi de supporter la mobilité pour permettre de gérer les informations depuis ou vers des dispositifs mobiles. Ils doivent permettre de plus l'orchestration des services dans un environnement ubiquitaire et interactif. Un Système d'Information Interactif (SII) est un SI dans lequel une personne est à la fois une source de données et un destinataire de l'information produite. L'adéquation de l'information fournie avec les besoins de l'utilisateur reflète la performance d'un SII et se mesure par la pertinence et l'utilisabilité de l'information.

Pour permettre l'amélioration de la qualité des services fournis par les SII, l'usage des technologies ubiquitaires permet la prise en compte du contexte d'exécution pour que les applications soient sensibles au contexte. Il s'agit en fait d'une qualité à intégrer aux SI pour augmenter leur niveau d'intelligence et d'autonomie. La prise en compte du contexte est la perception de l'environnement pour interagir plus naturellement avec l'utilisateur. Pour supporter cette fonctionnalité, des données

en relation avec le contexte sont utilisées telles que les données des capteurs de l'environnement physique, des matériels auto-descriptifs, la description des personnes ou encore les métadonnées sur les applications.

Les exigences issues de la mobilité représentent de nouveaux défis qui doivent être considérés et traités par le SI. La détection des évènements survenant dans l'environnement d'exécution est primordiale pour les applications mobiles pour qu'elles soient sensibles au contexte. Pour être réactifs au contexte, ces SI doivent réagir à ces évènements de façon efficace et en temps réel. De nombreux domaines d'application existent où la gestion des évènements est d'une importance cruciale tels que les marchés financiers, le commerce, la sécurité et les services mobiles. Dans ces domaines il est important de répondre en temps réel (ou dans un temps très peu différé) aux évènements pour gérer des situations exceptionnelles ou indiquer des opportunités ou des risques.

Un SI ubiquitaire doit assumer certaines caractéristiques telles que la scalabilité, l'invisibilité et la pro-activité. La scalabilité désigne le passage à l'échelle et la capacité de gérer un nombre croissant d'utilisateurs, d'applications et de matériels. L'invisibilité nécessite un minimum d'intervention humaine et l'adaptation aux changements se fait par auto-apprentissage. La pro-activité est la capacité d'analyser les situations et les problèmes pour suggérer et proposer à l'utilisation des solutions en réponse à son contexte actuel ou prévu.

Les caractéristiques de ces systèmes ubiquitaires et leurs capacités de traitement et de communication élargissent leur spectre d'application et justifient le nombre d'applications qui ne cesse de croître. Parmi ces applications, nous citons les systèmes d'alertes, d'assistance, les applications de travail à distance et les applications sociétales décrites ci-après.

Applications sociétales de l'informatique ubiquitaire

Plusieurs domaines s'orientent de plus en plus vers l'adoption des applications ubiquitaires pour gagner la fidélité de leurs clients en leur proposant des services mobiles et intelligents qui nécessitent de moindres efforts de la part de l'utilisateur. Les applications sociétales sont parmi les applications qui peuvent bénéficier le plus des technologies ubiquitaires. Nous nommons une application sociétale toute application proposant des services ayant trait à la vie quotidienne. Ces applications considèrent le contexte de la personne cible pour proposer des services personnalisés concernant différents domaines comme la santé, les loisirs, les voyages, les banques et le transport.

Parmi ces domaines nous citons par exemple le domaine médical où le patient est assisté en permanence par des capteurs qui détectent les informations sur sa santé et permettent grâce à des dispositifs mobiles d'accéder à des informations de diverses sources concernant la santé du patient, de conclure à son état et de détecter les situations d'urgence.

La recherche d'informations touristiques ou de loisir peut également trouver dans les systèmes ubiquitaires un champ d'application large puisque l'utilisateur souhaite dans la plupart des cas être informé, dans un contexte mobile, des lieux intéressants ou des hôtels près de sa localisation.

Nous nous intéressons plus spécifiquement au domaine du transport qui exige de plus en plus la prise en compte de la mobilité des voyageurs et une vigilance accrue aux évènements et aux incidents ainsi qu'une prise en compte des données contextuelles.

Le domaine des transports en commun

Comme de nombreux domaines, le domaine du transport présente un besoin croissant d'une gestion des évènements pour faire face à des situations critiques ou des incidents, et permettre de fournir une information en temps réel ou presque réel à l'usager. Le domaine du transport nécessite une gestion

efficace des évènements qui peuvent avoir plusieurs sources (travaux, pannes, absences d'un agent, conditions climatiques, etc.). Ainsi il est essentiel de prendre en compte les situations complexes de gestion des évènements depuis la phase de spécification pour permettre au système d'être flexible et autonome en envisageant, par exemple, le comportement souhaité suite à une séquence particulière d'évènements.

L'information dans le domaine du transport est fournie par l'environnement et est également diffusée dans l'environnement. Le traitement des informations de transport est caractérisé par l'utilisation d'outils et de techniques qui sont souvent sujets d'innovations et de modification. Parmi ces techniques, nous citons la gestion du profil, la recherche d'information, l'interrogation de capteurs, la géo-localisation et la diffusion d'informations.

Les Système d'Information de Transport (SIT) rencontrent de nombreux défis qui concernent l'hétérogénéité des données et des utilisateurs. En effet, les données utilisées sont très différentes et les mécanismes d'accès à ces données sont à leur tour différents. Par exemple, ces SI utilisent des cartes géographiques et des cartes de réseau de transport, des horaires théoriques et des horaires temps réel, des données météo et trafic, etc. De plus, les besoins fonctionnels des utilisateurs et les contextes d'utilisation sont de même très différents. Ainsi, ces SI représentent un cas idéal pour la conception et l'intégration des technologies ubiquitaires visant à la création d'un système utilisable et satisfaisant aux besoins des utilisateurs.

Les applications concernées par de tels systèmes d'information sont nombreuses, nous nous focalisons essentiellement sur des applications de type accompagnement de l'usager pendant son voyage, en particulier recalcul en temps réel d'un itinéraire en cas de perturbation (manifestation engendrant l'arrêt des tramways, travaux impliquant une modification de l'itinéraire d'un bus, etc.) et accompagnement d'un usager gêné durant son parcours (en particulier les personnes mal-entendantes qui n'entendent pas les informations données par les conducteurs lors de perturbations ou les personnes déficientes mentales vite paniquées en cas de perturbation sur leur itinéraire habituel). De tels besoins ont d'ailleurs été exprimés par les usagers de la SEMITAG (Transports de l'Agglomération Grenobloise) lors d'une enquête de satisfaction en 2009. Le projet DéSIT est une réponse à ces besoins.

Le Projet DéSIT

Cette thèse s'intègre dans le cadre du projet Démarche d'ingénierie des Systèmes d'Information de Transport pervasifs, sécurisés et personnalisables (DéSIT) [10] du cluster Transports Territoires et Société de la Région Rhône-Alpes. Ce projet est réalisé au sein de trois laboratoires : Laboratoire Informatique de Grenoble (LIG), Laboratoire d'infoRmatique en Image et Systèmes d'Information - Lyon (LIRIS) et LIP / PC2S. Les partenaires industriels de ce projet sont la société Tic et siT (Cabinet de conseil dans le domaine des SIT) et le Conseil Général de l'Isère (Collectivité territoriale).

Le projet DéSIT est organisé en trois lots :

- **Lot 1 :** proposer une déclinaison allégée des systèmes de requêtes par flux, adaptée aux équipements mobiles (optimisation mémoire et temps de calcul) et valider l'expressivité des requêtes en fonction des besoins métiers. Ce lot est pris en charge par les laboratoires LIRIS et LIG.
- **Lot 2 :** proposer une démarche d'ingénierie complète des SIT ubiquitaires. Ce lot est pris en charge par le LIG.
- **Lot 3 :** évaluer l'appropriation des SIT conçus par les usagers ainsi que leur utilisabilité. Ce lot est pris en charge par des chercheurs en psychologie du laboratoire LIP / PC2S.

L'objectif global de ce projet concerne la conception de systèmes d'information transport embarqués sur des téléphones portables, des iPhones, des tablettes tactiles, etc. tout en garantissant que de tels systèmes d'information s'adaptent au contexte d'utilisation (selon la localisation, les caractéristiques du dispositif mobile, les droits d'accès), soient personnalisés pour l'utilisateur (selon son profil,

ses préférences, ses activités), garantissent la confidentialité et la propriété des informations ainsi que la vie privée des personnes, et soient vulgarisables.

L'objectif de cette thèse est de définir et de résoudre les besoins conceptuels des systèmes ubiquitaires représentant les mêmes défis et la même complexité que les SI de transport. Le projet DéSIT et l'étude de cas du domaine de transport permettent l'illustration et l'application de nos contributions. La problématique généralisée de cette thèse est décrite ci-après.

Problématique de la thèse

En étudiant la vision actuelle des systèmes ubiquitaires à travers les différents travaux de la littérature et les systèmes existants récemment conçus nous remarquons que l'intérêt porte toujours sur les nouvelles technologies ubiquitaires et leur usage dans les systèmes. Ceci permet la création de systèmes très dépendants des technologies et nous remarquons l'existence d'applications mobiles et ambiantes qui utilisent un grand nombre de capteurs et de dispositifs pour recueillir des données sur l'utilisateur et son environnement et les utiliser dans leurs fonctions. Ceci permet de fournir des services personnalisés qui ne gagnent pas toujours la confiance des usagers : dans la plupart des cas, les usagers restent prudents et conservateurs. Cette situation est exprimée par une évolution corrélative entre les technologies ubiquitaires et les systèmes sous-jacents. Cette problématique est illustrée dans la vue A de la figure 1.

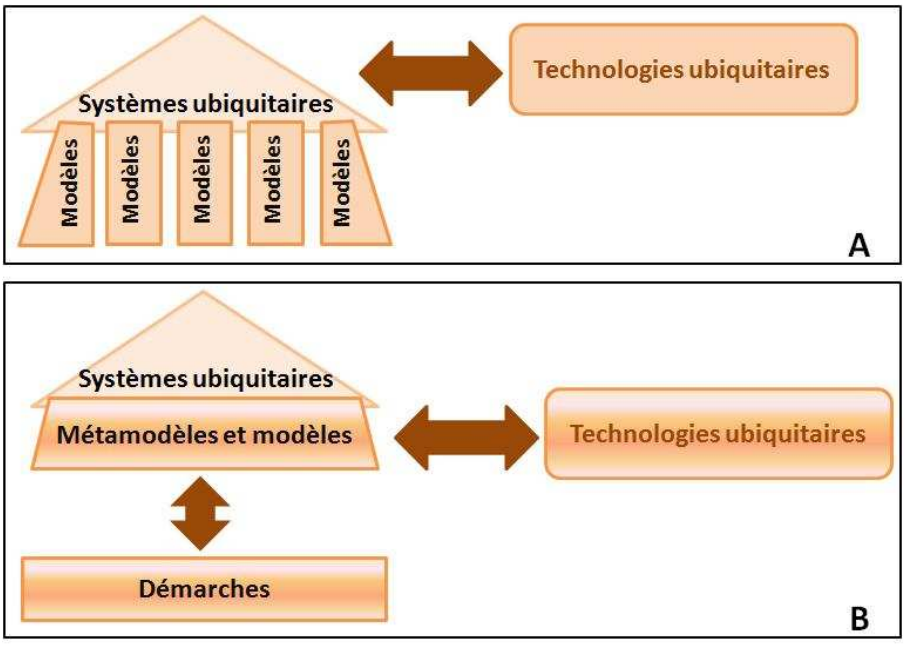


FIGURE 1 – Problématique de la conception des SI ubiquitaires

Malheureusement, dans cette vision des systèmes ubiquitaires, les technologies permettent de paramétrer les fonctionnalités des applications pour impressionner l'utilisateur et non pour garantir l'utilisabilité et l'exploitation efficace des services proposés. Ceci est dû en partie au manque de modèles standards supportant ces applications et garantissant le bon choix et le bon paramétrage des fonctionnalités tout en bénéficiant des technologies interactives et ambiantes qui permettent de servir les besoins des utilisateurs. Nous critiquons ici le manque de standards et la diversité et la non continuité des propositions de modèles (voir vue A de la figure 1).

Ainsi, un changement de vision est obligatoire pour renforcer les bases des systèmes ubiquitaires et minimiser les dépendances avec les technologies. Cette nouvelle vision est schématisée par la vue B de la figure 1 qui montre que ces systèmes doivent se baser sur un ensemble cohérent et complet de modèles. Ces modèles évoluent avec les technologies proposées sur le marché et l'intégration de ces technologies passe par le choix des modèles et la spécification des besoins des utilisateurs et ne se fait pas directement dans le système. De plus, pour pouvoir spécifier et appliquer ces modèles, il faut se référer à des démarches de développement et d'ingénierie pour faciliter le travail des développeurs et améliorer la qualité des systèmes conçues. Cette problématique actuelle de la conception des systèmes ubiquitaires regroupe plusieurs problématiques en étroite liaison et qui s'influencent mutuellement. Ces problématiques sont décrites ci-dessous.

- **Gestion des données.** Du point de vue de la gestion des données, les bases de données classiques doivent cohabiter avec des sources de données non conventionnelles comme les flux de données, les services et les événements. En effet, avec la mobilité et l'hétérogénéité des supports des applications, le défi majeur concerne la localisation des services fournissant les données. De plus, la gestion des données nécessite une optimisation des interactions en vu de fournir à l'utilisateur une information temps réel tenant compte des évènements imprévus, mais qui soit en même temps utile et répondant aux besoins de l'utilisateur. Les applications ubiquitaires doivent entre autres tolérer les contraintes physiques telles que les déconnexions sans interrompre le service. La mobilité des dispositifs entraîne un changement continu de l'environnement de l'application, ce qui nécessite une analyse des flux de données provenant de capteurs dynamiques pour passer d'une information brute de l'environnement à une donnée de contexte porteuse de sens pour l'application. Ainsi, le défi se manifeste dans la cohérence et la synchronisation des actions collaboratives de création, accès et modification des données partagées.
- **Gestion du contexte.** La gestion des données contextuelles et leur traitement sont parmi les défis majeurs des systèmes ubiquitaires. Les modèles proposés pour la définition et la spécification du contexte sont très nombreux et très diversifiés, ce qui constitue un obstacle devant l'interopérabilité des applications et des services. De plus, l'hétérogénéité des données nécessite l'usage de modèles génériques de haut niveau supportés par des métamodèles standards.
- **Gestion de la dynamicité.** Le contexte est un concept à caractère dynamique car il désigne des données personnelles ou non qui changent en fonction de la situation et de l'environnement. Ainsi, ce contexte est fortement dépendant des évènements qui l'entourent. Certaines fonctionnalités des applications ubiquitaires nécessitent une vigilance et une prise en compte en temps réel de ces évènements : tel est le cas des systèmes d'alerte, des services d'assistance ou des services financiers. Ainsi, les modèles supportant les systèmes ubiquitaires doivent permettre une gestion efficace et rapide du contexte et des évènements dans une représentation couplée, ce qui est rarement le cas des modèles existants.
- **Gestion de la sécurité.** Les données contextuelles sont en général des données personnelles qui représentent des données statiques sur l'utilisateur comme ses préférences et ses habitudes et des données dynamiques comme sa localisation et les tâches qu'il effectue. Ainsi, ces données possèdent un caractère confidentiel et l'échange de ces données entre applications doit respecter les mesures de sécurité définies par l'utilisateur. En effet, des utilisateurs peuvent présenter des attitudes conservatrices envers les applications ubiquitaires à cause de l'usage de leurs données personnelles. Pour gagner la confiance des utilisateurs, ces applications doivent garantir la confidentialité des données et la prise en compte des besoins de sécurité exprimées par l'utilisateur même.
- **Ingénierie des besoins ubiquitaires.** La garantie de l'utilisabilité et de la satisfaisabilité des applications ubiquitaires nécessite la spécification des besoins ubiquitaires des utilisateurs. La

littérature n'accorde pas un intérêt spécifique à l'ingénierie de ces besoins et à l'attitude des utilisateurs envers l'usage des technologies ubiquitaires.

- **Démarche d'ingénierie.** La spécification de ces besoins et la définition de modèles ubiquitaires complets et surmontant les lacunes précitées ne peuvent être possibles sans l'adoption d'une méthode de développement qui permet d'organiser le travail de spécification et de fournir les modèles architecturaux et fonctionnels adaptés. La méthode doit contenir une démarche d'ingénierie facilitant la conception des applications et permettant le support de toutes les caractéristiques ubiquitaires de tels systèmes.

La conception des systèmes sensibles au contexte relève des défis reliés principalement à la collection, capture, modélisation, stockage, distribution et contrôle des informations contextuelles. Cette thèse est une réponse à ces défis par la proposition d'une méthode de développement des SI ubiquitaires décrite brièvement dans la section suivante.

Contributions

Notre objectif dans ce travail de thèse est de proposer une méthode d'ingénierie des SI ubiquitaires en considérant les différentes exigences reliées à la nature mobile et grande échelle de ces systèmes. Cette méthode est basée sur une démarche de développement qui fait usage d'un ensemble de métamodèles et de langages favorisant la spécification complète de ces systèmes. Les caractéristiques de cette démarche nommée " E-CARe" sont décrites ci-après.

- La démarche réutilise les phases de développement d'une démarche classique de conception des SI pour garantir la couverture des spécifications fonctionnelles et techniques et éviter la définition de phases inutilement. En effet, le domaine des SI bénéficie d'un grand nombre de démarches qui doivent être réutilisées et exploitées.
- La démarche sépare les spécifications fonctionnelles, techniques et ubiquitaires pour permettre la définition de modèles indépendants qui évoluent de façon séparée, favorisant ainsi la maintenance et les modifications. Ces modèles fusionnent dans les phases de conception pour fournir un ensemble cohérent d'architectures et de modèles contenant les caractéristiques ubiquitaires du système.
- L'étude des besoins des utilisateurs est une étape essentielle de cette démarche qui permet d'utiliser les objectifs pour déterminer les besoins sous forme de règles métier, ubiquitaires et techniques. Ces règles permettent la divergence de trois branches correspondant aux spécifications fonctionnelles, ubiquitaires et techniques. Les règles ubiquitaires permettent la définition des besoins d'adaptation et de prise en compte du contexte.
- La branche ubiquitaire est prise en charge par un nouveau rôle de concepteur de contexte. Ce rôle doit avoir des compétences spécifiques qui sont nécessaires pour la définition des modèles contextuels et la spécification des besoins d'adaptation. Ces compétences sont aussi bien du domaine informatique que d'autres domaines sociaux et psychologiques.
- La modélisation structurelle du contexte supportée par cette démarche permet de définir un modèle de contexte comportant toutes les données contextuelles nécessaires pour les fonctionnalités d'adaptation et de sensibilité au contexte. Un métamodèle générique est proposé pour être instancié et pour construire le modèle structurel de contexte. La généricité de ce métamodèle est validée grâce à des expérimentations auprès de spécialistes de différents domaines.
- L'aspect dynamique de l'application est spécifié après la définition de modèles évènementiels intervenant dans les fonctionnalités ubiquitaires. Ces évènements permettent de détecter différents changements à l'intérieur ou dans l'environnement de l'application. Ils permettent l'acquisition de données contextuelles, le déclenchement de comportements ubiquitaires et la communication

avec les acteurs externes.
- L'exécution des règles ubiquitaires est supportée par un moteur d'évènements qui représente un composant indépendant des composants métiers facilitant ainsi la maintenance et les modifications ubiquitaires.
- Les aspects confidentiels et privés du système sont supportés par un niveau conceptuel et un niveau architectural adaptés. L'architecture adoptée est une architecture distribuée respectant un échange minimal de données du contexte par la distribution de la prise de décision et la multiplication des moteurs d'évènements. D'un point de vue conceptuel, une approche de contrôle d'accès est adoptée par l'usage de règles d'accès contextuelles.
- Une architecture de traitement des évènements permet de définir des sous-systèmes distribués qui communiquent par échanges d'évènements. Cette architecture respecte le découplage entre les modèles fonctionnels.
- Les caractéristiques ubiquitaires du système sont intégrées dans les modèles fonctionnels et techniques par fusion des modèles.

Pour valider cette démarche, nous avons réalisé des expérimentations qui permettent de définir les habitudes des concepteurs pour la modélisation du contexte et l'intégration de la dynamicité dans les systèmes ubiquitaires.

De plus, nous avons appliqué cette démarche dans le domaine des transports en commun pour concevoir une application d'assistance au voyageur. Cette application doit considérer les besoins des personnes à mobilité réduite ou des personnes déficientes mentales. cette démarche constitue l'objectif du lot 2 du projet DéSIT dans lequel s'intègre cette thèse.

Plan du manuscrit

Ce mémoire présente la méthode d'ingénierie proposée de façon progressive.

Nous décrivons en première partie les éléments de l'état de l'art sur lesquels s'appuie cette méthode en présentant une étude comparative :

- Le **chapitre 1** présente le domaine de la sensibilité au contexte en étudiant différentes approches d'appropriation et d'usage du contexte. Ce chapitre explique les différents termes reliés à ce thème et explore l'importance de la sensibilité au contexte pour les différents niveaux du système : métier, présentation, gestion des données et contrôle d'accès. Il étudie de plus différentes approches de modélisation et de métamodélisation du contexte.
- Le **chapitre 2** étudie les approches existantes sur l'ingénierie des SI ubiquitaires. Il aborde les bases conceptuelles et architecturales des approches orientées évènements et leur adéquation pour le support des caractéristiques ubiquitaires des systèmes. De même, la prise en compte de ces caractéristiques est évaluée dans les démarches classiques et unifiées d'ingénierie des SI. Des démarches d'ingénierie spécifiques aux systèmes ubiquitaires proposées dans la littérature sont présentées et comparées.

La deuxième partie du mémoire présente nos contributions et la méthode proposée qui est constituée par des concepts de base et une démarche :

- Le **chapitre 3** présente les concepts de base de la méthode, à savoir les approches et les métamodèles proposés. Nous décrivons notre vision d'un SI ubiquitaire ainsi que notre approche orientée évènements et notre approche intentionnelle permettant la spécification des besoins dynamiques et ubiquitaires. Nous proposons un ensemble de métamodèles génériques pour la modélisation structurelle et évènementielle du contexte.
- Le **chapitre 4** décrit la branche ubiquitaire de la démarche "Engineering - Context Aware and Reactive systems (E-CARe)" qui permet de produire les modèles ubiquitaires du système en

focalisant sur l'adaptation et la sensibilité au contexte. Elle permet la spécification intentionnelle des besoins ubiquitaires et la modélisation structurelle et évènementielle du contexte.

– Le **chapitre 5** présente la démarche complète d'ingénierie des SI ubiquitaires E-CARe. Cette démarche intègre la branche ubiquitaire dans la démarche unifiée "Symphony" et permet de spécifier les différents besoins fonctionnels, ubiquitaires et techniques dans un cycle de développement en forme de "ψ". Ainsi avons-nous ajouté une branche "Analyse des besoins ubiquitaires" aux branches traditionnelles d'un cycle 2TUP [Rocques et Vallée, 2002]. La conception permet de rapprocher et de fusionner les différents modèles.

La troisième partie permet de valider nos propositions de démarche et de métamodèles :

– Le **chapitre 6** présente des expérimentations de validation des métamodèles proposés et illustre l'application de la démarche sur une application d'assistance au voyageur. Des expérimentations réalisées auprès de concepteurs d'applications ubiquitaires de différentes spécialités permettent de valider la généricité des métamodèles structurels et évènementiels de contexte et de capturer les habitudes de conception des différents sujets.

  De plus, un prototype d'une application d'assistance au voyageur est implémenté sur un dispositif mobile en utilisant une architecture orientée évènements et des échanges d'évènements avec la technologie Bluetooth.

Finalement, nous synthétisons nos contributions, identifions les limites de nos travaux et proposons de nouvelles perspectives d'ouverture.

Première partie

Etat de l'art

Chapitre 1

Sensibilité au contexte dans les SI

En 1991, Mark Weiser du centre de recherche Palo Alto de Xerox déclarait dans [Weiser, 1991], que les technologies les plus réussies sont celles qui s'associent avec la vie de tous les jours jusqu'à ce qu'il devient difficile de distinguer entre les deux. Weiser a proposé les prémisses d'un nouveau paradigme d'interaction, qui se focalise sur le but de l'interaction plutôt que sur l'interaction elle-même. Cette vision de Weiser était derrière la naissance de ce qu'on appelle l'informatique ubiquitaire (ou aussi pervasive ou ambiante) qui vise à utiliser, ensemble, les ressources informatiques enfouies dans l'environnement et les mettre en communication pour proposer des services innovants. Aujourd'hui, les travaux de recherche accordent un grand intérêt aux systèmes sensibles et réactifs au contexte, dans le cadre du paradigme de système ubiquitaire. Un aperçu étendu sous forme d'inventaire de ces travaux est fourni par [Hong *et al.*, 2009]. Ces travaux appartiennent à plusieurs domaines de recherche visant à fournir des bases conceptuelles et techniques solides pour ces systèmes. Nous remarquons l'existence de deux axes qui permettent de définir les problématiques supportées. Le premier axe représente les différents domaines d'application qui possèdent des besoins d'adaptation différents et des caractéristiques diverses tels que le domaine médical, le domaine des transports, le domaine financier, le domaine du tourisme et le domaine éducatif. Le deuxième axe représente les disciplines de recherche qui vont des infrastructure des systèmes, aux réseaux de capteurs, à la gestion des données, à la structure métier et aux interactions hommes-machines. Le premier axe en intégrant les besoins ubiquitaires fait émerger de nouvelles fonctionnalités et de nouvelles applications liées à la mobilité, l'interactivité et/ou la flexibilité. Le deuxième axe, concernant les différentes disciplines de recherche, est en harmonie avec le processus de gestion du contexte ou le cycle de vie de l'information contextuelle qui va de l'acquisition du contexte à l'appropriation du contexte et à l'usage du contexte. La figure 1.1 illustre ce cycle de vie.

La première phase permet d'acquérir le contexte en collectant les informations à partir de sources diverses (capteurs, utilisateurs, services tiers, etc) puis d'appliquer des opérations de gestion en vue de filtrer, composer et traiter ces informations. Dans une deuxième phase, ces informations seront modélisées et stockées dans des bases de données. La dernière phase concerne un besoin d'une donnée de contexte, que le système peut chercher dans les bases correspondantes et raisonner pour produire des actions d'adaptation.

La première phase concerne plutôt l'acquisition du contexte qui sera en partie abordée dans le chapitre suivant. Dans ce chapitre, nous présentons un état de l'art des deux dernières phases de ce cycle de vie, à savoir l'appropriation et l'usage du contexte. Mais tout d'abord nous présentons une étude du vocabulaire de ce domaine de recherche et de domaines connexes pour éviter toute ambiguïté liée aux termes utilisés dans ce mémoire.

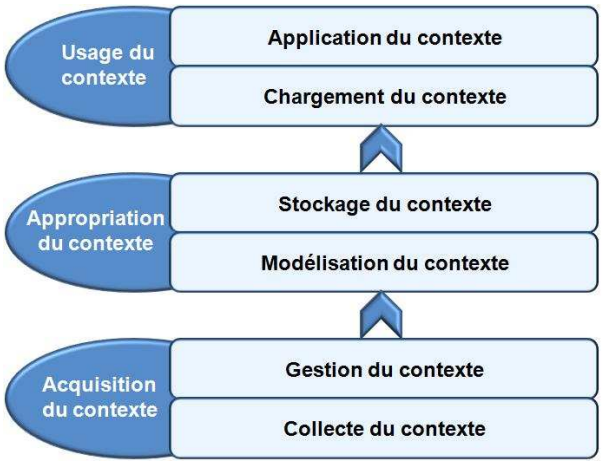


FIGURE 1.1 – Cycle de vie du contexte dans un SI ubiquitaire

1.1 Étude du vocabulaire de l'adaptation

Les travaux s'inscrivant dans le thème de la mobilité et de l'ubiquité utilisent des vocabulaires différents et des concepts divers pour exprimer les performances qui découlent de la nature de ces systèmes à savoir : l'adaptabilité, la flexibilité, la sensibilité au contexte et la personnalisation. Tous ces termes appartiennent à un vocabulaire que nous considérons propre aux systèmes dotés d'une forme de changement. Nous définissons dans cette section ces différents concepts et nous présentons notre prise de position par rapport à l'utilisation des différents termes.

1.1.1 Adaptation

L'adaptation facile et rapide aux changements internes et externes est une faculté recherchée et souhaitée par les systèmes ubiquitaires. L'adaptation est ainsi un facteur positif améliorant l'utilisabilité et la performance du système. En informatique, l'adaptation peut prendre deux formes : adaptativité et adaptabilité.

D'après l'encyclopédie Larousse, l'adaptativité est le synonyme de l'autoadaptation qui signifie « l'aptitude d'un système à modifier ses paramètres de structure de manière à ce que son fonctionnement demeure satisfaisant en dépit des variations de son environnement ». Ainsi un système est adaptatif s'il peut s'adapter automatiquement à une situation ou un contexte d'utilisation.

L'adaptabilité est définie dans l'encyclopédie Larousse comme étant la capacité de s'adapter à de nouvelles situations. Cette définition est assez vague et ne montre pas la différence entre adaptabilité et adaptativité. Mais en informatique il existe une différence décelable : un système adaptable est un système qui peut exécuter les demandes de personnalisation de l'utilisateur. Ainsi l'utilisateur doit intervenir volontairement et de façon explicite pour exprimer des situations d'adaptation et la réponse associée à chaque situation.

Les systèmes ubiquitaires doivent être dotés d'au moins une de ces formes d'adaptation.

1.1.2 Flexibilité

D'après le dictionnaire McGraw-Hill Dictionary of Scientific Technical Terms, la flexibilité en informatique est la capacité d'un logiciel de changer facilement en réponse à différents utilisateurs et différents besoins du système. Selon l'encyclopédie Larousse, la flexibilité signifie l'aptitude d'un

système construit à se plier à une utilisation évolutive ou différente. Il s'agit ici d'un système qui peut être modifié sans être remplacé complètement ce qui convient avec la définition donnée par [Regev et Wegmann, 2005] stipulant que « la flexibilité est la capacité de se décliner aux changements sans disparaître ». Tandis que [Golden et Powell, 2000] définissent la flexibilité comme la capacité à s'adapter.

La flexibilité est plus générale que l'adaptation et elle l'englobe. L'adaptation est une façon de réaliser la flexibilité en prenant en compte certaines situations pour modifier le comportement et l'apparence d'un système. La flexibilité peut intéresser non seulement les utilisateurs mais aussi les concepteurs, pour désigner la facilité de modifier un système en fonction d'autres facteurs comme l'évolution des technologies et des besoins. Ainsi un système est flexible s'il présente une capacité de changement en fonction de n'importe quel facteur et pour toute partie impliquée dans une phase de développement et d'exploitation du système.

1.1.3 Personnalisation

D'après l'encyclopédie Larousse, la personnalisation est l'adaptation d'un service à celui à qui il est destiné. Les différents travaux menés sur la personnalisation sont d'accord sur le fait que la personnalisation sert à adapter une information, un service, un logiciel ou un dispositif à une personne, nécessairement l'utilisateur, en prenant en compte ses données personnelles et environnementales. La notion de personnalisation est étroitement reliée au concept de « profil ». Un profil est toujours associé à une entité du système. Il constitue un cadre conceptuel groupant des informations caractérisant cette entité et intéressantes pour le système avec une durée de temps dédiée à cet intérêt. Les données du profil sont collectées de deux façons différentes :

- Collecte explicite des informations : l'utilisateur exprime et saisit ses préférences. Ce mode implique la volonté et le temps de l'usager. Le profil peut devenir inapproprié avec le temps si les données saisies changent.
- Collecte implicite des informations : (implicit user feedback), par l'intermédiaire de techniques pilotant les activités de l'utilisateur pour dériver des données sur ses habitudes ou ses préférences.

Avec la personnalisation, un seul service offert peut être perçu différemment par différents utilisateurs au même moment.

1.1.4 Contexte

Le mot « Contexte »prend ses origines du latin. Il est lié à deux concepts Contextus (assemblage) et contextere (tisser avec). Ainsi, l'origine latine du mot met l'accent sur l'aspect constructif et associatif du terme contexte.

De même, les dictionnaires donnent des définitions intéressantes du contexte. Dans l'encyclopédie Larousse, « le contexte est l'ensemble des circonstances dans lesquelles se produit un évènement, se situe une action ». Hachette Multimédia définit le contexte comme "l'ensemble des éléments qui entourent un fait et permettent de le comprendre". Ces deux définitions vues d'une perspective informatique montrent que le contexte est un ensemble de données utilisées pour justifier une action. Ainsi le contexte peut être utilisé pour adapter des actions ou des fonctionnalités du système à des circonstances jugées pertinentes et utiles.

Le contexte est défini dans les travaux informatiques de trois façons : soit par des synonymes, soit par des explications [Hull *et al.*, 1997] (l'environnement d'une application), soit par des exemples [Brown *et al.*, 1997] en énumérant des éléments comme la localisation, la plateforme d'interaction, les préférences, le temps, etc. L'utilisation des synonymes souffre forcément de la non exactitude des

termes choisis, l'utilisation des explications textuelles est générale et abstraite, alors que l'utilisation des exemples ne peut pas couvrir tous les aspects et reste ainsi incomplète.

Dans plusieurs travaux, le concept de contexte est très relié à la notion de situation. Parmi ces travaux nous citons la définition donnée par Schmidt et al [Schmidt *et al.*, 1999] : « le contexte est la situation dans laquelle se trouve un utilisateur », qui reste une définitions abstraite. La définition de [Dey, 2001] est plus explicative : « Le contexte est toute information utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application même ». La définition de Dey est la plus répandue et la plus adoptée dans les travaux portant sur la sensibilité au contexte. Elle considère que la tâche est un concept important faisant partie du contexte. Mais selon [Henricksen, 2003] la définition de Dey ne contient pas une séparation claire entre les concepts contexte, modèle de contexte et information de contexte. Pour [Henricksen, 2003] le contexte d'une tâche est l'ensemble des circonstances qui l'entourent et qui sont considérées comme pertinentes pour sa réalisation. D'autres définitions considèrent le contexte comme un ensemble de caractéristiques qui dépendent des différents courants de recherche tels que les activités de l'entité, la plateforme de communication utilisée et son rôle dans une organisation.

Une autre vision intéressante du contexte est fournie par [Zimmermann *et al.*, 2007] qui permet de couvrir les différents aspects pour définir le contexte et de dépasser les lacunes des définitions précédentes. Il définit le contexte comme toute information qui peut être utilisée pour caractériser la situation d'une entité. Les éléments pour la description de ces informations de contexte appartiennent à 5 catégories : individualité, activité, localisation, temps et relation. L'activité est le concept prédominant pour déterminer la pertinence des autres éléments de contexte dans une situation spécifique. Le temps et la localisation sont les concepts permettant de créer des relations entre les entités et d'autoriser l'échange d'informations contextuelles entre elles.

Les données contextuelles peuvent être invalides ou erronées, c'est pourquoi la notion de contexte est généralement liée à des mesures de qualité appelées Quality of Context (QoC). Le contexte doit refléter les changements de situation d'une entité dans le temps et l'espace, ce qui donne au contexte un caractère extrêmement dynamique.

1.1.5 Sensibilité au contexte

La sensibilité au contexte ou encore la prise en compte du contexte (context awareness) est une approche prometteuse permettant de supporter les approches ubiquitaires et orientées utilisateurs. Pour [Schmidt *et al.*, 1999], la sensibilité au contexte est la sensibilité à la situation dans laquelle se trouve un utilisateur. Selon Dey [Dey, 2001] un système sensible au contexte est « un système qui utilise le contexte pour fournir des informations et/ou des services pertinents à un utilisateur. La pertinence dépend de la tâche de l'utilisateur », alors que [Henricksen, 2003] considère qu'une application est sensible au contexte si elle s'adapte aux changements de l'environnement et aux besoins de l'utilisateur.

Dans nos travaux, la sensibilité au contexte est un paradigme permettant d'adapter les fonctionnalités, les caractéristiques et les interactions d'un système au contexte. Ceci doit être accompagné surtout d'une vision structurée et unifiée du monde dans lequel le système s'exécute et qui évolue avec le processus d'interaction et les changements de l'environnement [Coutaz *et al.*, 2005].

1.1.6 Synthèse : relations entre concepts

Pour les systèmes ubiquitaires qui doivent interagir de façon efficace et performante, il est indispensable de fournir des services utiles et personnalisés aux usagers. Parfois le système doit modifier

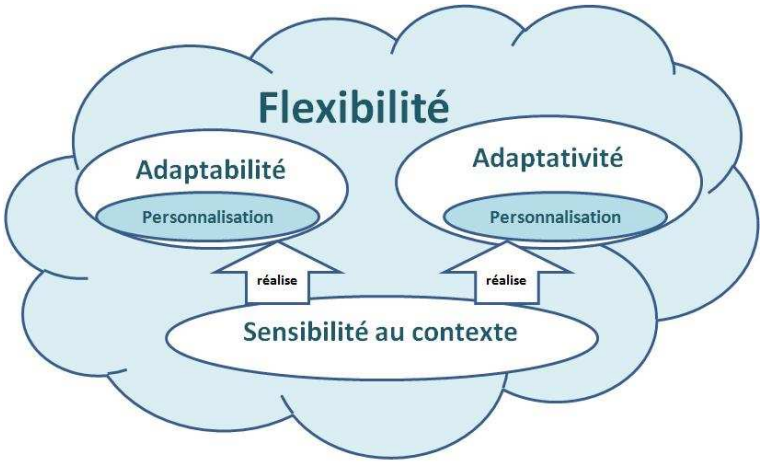


FIGURE 1.2 – Relations entre concepts autour de l'adaptation

son comportement et sa composition pour pouvoir convenir à plusieurs circonstances et plusieurs usagers. Nous parlons ainsi de l'adaptation qui peut être soit une adaptabilité (l'usager définit comment se fait l'adaptation et quand) soit une adaptativité (le système s'adapte automatiquement sans intervention externe). L'adaptation peut être généralisée pour concerner une déclinaison du système à des circonstances l'influençant directement ou indirectement et faisant intervenir des acteurs impliqués dans n'importe quelle étape de son cycle de vie. On parle d'un système flexible.

La personnalisation est un cas particulier de l'adaptation qui permet de fournir des services (notamment des informations et des logiciels) adaptés à l'utilisateur et à ses besoins.

La sensibilité au contexte est la technique permettant d'introduire l'adaptation dans les systèmes sachant que le contexte peut contenir n'importe quelle circonstance caractérisant une action, une interaction ou une entité. Les liens entre les différents concepts selon notre prise de position sont résumés dans la figure 1.2.

Dans la conception des SI ubiquitaires, l'adaptation sous ces deux formes (adaptabilité ou adaptativité) constitue l'objectif visé pour réaliser des systèmes utilisables et satisfaisants. Dans cette perspective, il est indispensable d'inclure la sensibilité au contexte comme technique pour réaliser l'adaptation. Ainsi, il s'avère intéressant de prendre en compte les méthodes et les approches basées contexte pour comprendre comment le contexte peut être construit, géré et exploité. La section suivante traite l'aspect modélisation et métamodélisation du contexte.

1.2 Modélisation et métamodélisation du contexte

La technique adoptée pour introduire l'adaptation dans les SI ubiquitaires est de prendre en compte le contexte. Mais le contexte doit être construit de manière structurée et ordonnée pour qu'il soit exploité de façon efficace et facile. Ainsi le respect d'un métamodèle s'avère utile voire nécessaire. Cette section présente des travaux de recherche pour résumer les différentes approches de modélisation et de métamodélisation du contexte.

Comme la modélisation du contexte est indispensable pour le développement d'un SI ubiquitaire, les travaux sur la modélisation du contexte sont beaucoup plus nombreux que ceux sur la métamodélisation.

1.2.1 Modélisation du contexte

La modélisation a pour objectif de représenter une idée ou un phénomène du monde réel dans une description simplifiée. L'objectif de la modélisation du contexte est de fournir des représentations abstraites des concepts utilisés pour prendre en compte le contexte. « Un modèle de contexte identifie un sous-ensemble du contexte qui est accessible de façon réaliste à travers des capteurs, des applications et des utilisateurs et qui peut être exploité pour l'exécution de la tâche. Le modèle de contexte qui est utilisé par une application sensible au contexte est généralement spécifié par le développeur de l'application, mais peut changer dans le temps »[Henricksen, 2003].

La définition de modèles de contexte est toujours accompagnée d'un développement de systèmes de gestion de contextes appropriés pour la collecte, la dérivation, la gestion et l'application des informations contextuelles. Le contexte peut être fourni au système selon des niveaux d'abstraction différents et le modèle doit permettre de représenter ces différents niveaux d'abstraction et de les relier pour pouvoir raisonner sur le contexte.

Les informations contextuelles proviennent de plusieurs sources de natures diverses et hétérogènes telles que les capteurs, les entrées utilisateurs, les données systèmes ou les interactions avec d'autres systèmes. Ces informations présentent des caractéristiques différentes. Par exemple les données provenant de capteurs sont généralement très dynamiques et peuvent présenter des erreurs et du bruit, alors que les informations fournies par l'utilisateur sont plutôt fiables mais deviennent rapidement obsolètes. Ainsi le besoin de modèles de contexte adéquats et solides devient une nécessité pour construire des applications et des systèmes ubiquitaires qui raisonnent de façon pertinente.

Selon [Strang et Linnhoff-Popien, 2004], un ensemble de défis doit être considéré lors de la modélisation du contexte tels que la composition distribuée des données, la validation partielle, la qualité de l'information, l'ambiguïté et l'incomplétude. Plusieurs techniques de modélisation du contexte ont émergé dans la littérature et ont été classifiées par plusieurs travaux comme [Strang et Linnhoff-Popien, 2004] et [Bettini *et al.*, 2009]. Un aperçu de ces modèles de contexte est présenté ci-dessous.

Approche de modélisation par mots clés

Cette approche consiste à associer à un utilisateur un ensemble de mots clés pondérés. Ces mots clés sont extraits du domaine de l'application. Les poids associés à chaque mot clé sont une représentation numérique de l'intérêt de l'utilisateur par rapport au mot concerné. Ces mots clés sont généralement utilisés pour représenter des préférences ou des centres d'intérêt qui constituent une dimension du profil de l'utilisateur. Ces mots clés pondérés peuvent être groupés et ordonnés dans des catégories pour une navigation et une gestion plus facile des données. Cette représentation du profil est parmi les premières techniques et elle est utilisée pour les systèmes navigationnels [Moukas, 1997], les systèmes de recommandation [Balabanovic et Shoham, 1997] et les systèmes de recherche d'information [Lieberman, 1995]. Cette technique de modélisation représente plusieurs inconvénients. En effet, elle ne permet pas de représenter des types d'informations contextuelles autres que les préférences et les intérêts. De plus elle ne permet pas de capturer des relations et des dépendances entre les informations du contexte.

Languages à balises (Markup schema)

Les langages à balises tels que XML ont été utilisés pour représenter des modèles de contexte. Tous les modèles à base de balises représentent une structure de données hiérarchique sous forme de balises contenant des attributs et leurs contenus. Le standard du W3C pour la description des systèmes mobiles, Composite Capabilities/ Preference Profile (CC/PP) [Klyne *et al.*, 2004] est parmi

les premières approches utilisant Resource Description Framework (RDF) pour modéliser le contexte. CC/PP utilise des contraintes élémentaires et des relations entre les types de données contextuelles. Cette approche permet de représenter typiquement les profils, composant essentiel pour la construction du contexte. Un deuxième standard est UAProf (User Agent Profile) [Wapforum, 2001] qui utilise à la fois Resource Description Framework Schema (RDFS) et eXtensible Markup Language (XML) pour plus d'expressivité. Des exemples peuvent être trouvés dans [Strang et Linnhoff-Popien, 2004].

Modèles graphiques

Le modèle graphique le plus répandu et le plus utilisé est le standard de l'Object Management Group (OMG) : UML. Les diagrammes de classes d'UML sont utilisés pour modéliser de façon statique les concepts qui constituent le contexte et les relations qui les connectent. Les classes UML sont utilisées pour représenter des éléments du contexte avec leurs attributs et les associations représentent les liens entre les concepts. Cette modélisation profite des avantages des langages orientés objets tels que la possibilité de représenter des associations d'héritage, et d'encapsulation. Un exemple d'utilisation d'UML est introduit dans [Sheng et Benatallah, 2005].

Le deuxième exemple de langage graphique est le modèle orienté rôle Context Modeling Language (CML) qui est une extension de Object-Role Modeling (ORM) et qui est proposé par [Henricksen et Indulska, 2006]. Ce langage représente les avantages suivants :

- capturer les différentes classes et les sources de contexte et les qualifier comme : statiques, capturées (par capteur), dérivées ou fournies par l'utilisateur (ou profilées) ;
- permettre de saisir de l'information imparfaite à l'aide de métadonnées de qualité et mettre en évidence les données contradictoires ;
- capturer les dépendances entre les types de données de contexte ;
- sauvegarder l'historique de certaines données contextuelles et définir des contraintes sur les données historisées.

Cependant, l'inconvénient majeur de CML est qu'il n'est pas standardisé et qu'il n'existe pas d'outil support.

Modèles de contexte basés ontologies

Les ontologies constituent un moyen puissant pour représenter les concepts et les relations entre eux. En effet, elles possèdent un pouvoir expressif élevé bénéficiant de techniques de raisonnement bien définies et supportées par des outils automatisés. Plusieurs systèmes sensibles au contexte utilisent les ontologies pour représenter le contexte. Les ontologies permettent de fournir une sémantique formelle des données de contexte en vue de partager le contexte entre différentes sources. Des outils de raisonnement peuvent être utilisés à la fois pour vérifier la cohérence de l'ensemble des relations décrivant un scénario de contexte et pour construire une vue plus abstraite du contexte de haut niveau.

Une des premières approches utilisant les ontologies pour la modélisation du contexte a été proposée par Otzturk et al. dans [Otzturk et Aamodt, 1997].Puis plusieurs modèles d'ontologies ont été proposés pour représenter des descriptions de données contextuelles tels que SOUPA [Chen *et al.*, 2004b] pour modéliser le contexte dans des environnements ubiquitaires et CONON [Zhang *et al.*, 2005] pour modéliser le contexte dans le domaine du bâtiment intelligent.

Les modèles ontologiques OWL-DL ont été utilisés dans plusieurs systèmes sensibles au contexte parmi lesquels nous citons CoBrA (Context Broker Architecture) [Chen *et al.*, 2004a] qui utilise SOUPA et l'intergiciel SOCAM [Gu *et al.*, 2004] qui utilise CONON.

Tous les modèles de contexte, ci-dessus décrits, sont à titre égal d'utilisabilité et nous ne présentons aucune préférence d'un langage par rapport à un autre. Cependant, nous préférons l'existence

de métamodèles supportant le travail de modélisation, dans le but de fournir une référence aux développeurs pour guider le travail de modélisation. De plus, l'usage d'un métamodèle référentiel facilite l'interopérabilité entre les différentes applications.

1.2.2 Métamodélisation du contexte

La section précédente présente des approches de modélisation du contexte, qui permettent la représentation du contexte dans les applications ubiquitaires. La métamodélisation consiste à structurer et présenter une vue plus abstraite des modèles de contexte dans le but de montrer comment ces derniers sont construits. La métamodélisation du contexte constitue un défi majeur pour l'identification des concepts concernés par la manipulation du contexte, leurs liens, leur formalisation et la présentation de leur sémantique. Plusieurs travaux de recherche dans ce domaine se contentent de définir les modèles de contexte des applications ou des systèmes concernés sans se soucier du métamodèle qui supporte ces modèles.

Or, pour proposer des démarches de conception d'applications ubiquitaires, une structuration de haut niveau est nécessaire pour guider et organiser la modélisation du contexte. Ainsi cette section s'intéresse aux approches de métamodélisation du contexte en vue de les comparer.

Dans plusieurs travaux et domaines de recherche, le contexte est vu comme un ensemble de dimensions de même niveau et de même importance. Ceci explique la vision multi-dimensionnelle pour modéliser le contexte qui se décèle clairement dans des travaux comme Caméléon reference Framework [Calvary *et al.*, 2003] qui considère que le contexte est un triplet (plateforme, usager, environnement) ou encore dans [Bouzeghoub et Kostadinov, 2005]. L'approche proposée par [Zimmermann *et al.*, 2007] est intéressante puisqu'elle s'appuie sur une analyse textuelle réaliste et bien fondée des dimensions du contexte et des liens qui existent et qui peuvent évoluer en fonction du temps ou du contexte (une boucle itérative de ce terme concept). Ces dimensions sont : l'individualité, les relations, l'activité, le temps et la localisation (voir section 1.1.4).

Malgré le fait que ces approches multi-dimensionnelles trouvent une acceptation et une adoption par plusieurs approches, nous sommes convaincus qu'elles ne peuvent pas être généralisées pour s'appliquer à des systèmes plus complexes. En effet, ces approches ne permettent pas de modéliser des liens entre éléments et entités du contexte. De plus, les informations du contexte sont représentées dans un même niveau ce qui ne permet pas d'exprimer des structures hiérarchiques. Ainsi un besoin d'un métamodèle de contexte plus riche et plus expressif persiste. Ci-dessous nous présentons une étude d'un certain nombre de métamodèles proposés dans la littérature. Notre but n'est pas d'étudier l'implémentation et le support technique de ces métamodèles, mais plutôt d'analyser la structure et les composants de ces métamodèles. L'objectif est d'étudier la généricité et la bonne construction de ces métamodèles pour répondre aux besoins de plusieurs domaines sans pour autant être surchargés et difficiles à instancier.

Métamodèles supportant des modèles ontologiques

Sous cette catégorie nous groupons les métamodèles utilisés pour construire des modèles ontologiques de contexte. Ces métamodèles partagent des propriétés comme : la simplicité, l'expressivité et l'abstraction. Concernant la simplicité, elle désigne le nombre restreint de concepts et ainsi la taille réduite du métamodèle. En effet, ces métamodèles se concentrent sur la définition des métaclasses et des relations ontologiques de haut niveau pour la classification et le typage des concepts du contexte. L'expressivité de ces métamodèles est due à l'usage des ontologies qui permettent d'instancier les métaclasses pour construire des réseaux de concepts. L'usage des ontologies permet également de raisonner sur ces concepts par la définition de propriétés et de règles de description en logique du

premier ordre. Les métamodèles définis ainsi sont de plus assez abstraits pour ne pas faire apparaître des concepts spécifiques à certains domaines ou certaines applications.

Par exemple, [Fuchs *et al.*, 2005] propose un métamodèle qui utilise un constructeur "Entity class" pour représenter des entités telles que des personnes, des lieux, des objets et des évènements ; entités qui partagent des propriétés communes. Ces propriétés sont typées par le constructeur "Datatype class". Un constructeur particulier permet de mesurer la qualité du contexte et de représenter les aspects qualitatifs des données de contexte (précision, certitude, résolution, etc). Un mapping formel est fait entre ce métamodèle et OWL-DL.

Tchienehom dans [Tchienehom, 2005] propose un métamodèle divisé en quatre compartiments représentant respectivement la structure logique des données du profil, la sémantique de cette structure, le contenu des données ainsi que la sémantique de ce contenu. Ce métamodèle supporte des langages comme RDF, RDFS et Web Ontology Language (OWL).

Le point commun entre ces métamodèles est le souci de classifier et de typer les concepts et de les relier avec des valeurs et des mesures typées. Ces métamodèles sont génériques mais pas assez précis pour prévoir des liens dynamiques entre les concepts et entre les concepts et les composants de l'application. La structure statique est bien définie, mais la dynamicité n'est pas supportée. Dans la suite nous nous intéressons aux métamodèles objets censés être indépendants des applications et des domaines.

Métamodèle basé sur la notion de “focus”

Le métamodèle de contexte proposé par Vieira et al. dans [Vieira *et al.*, 2010] montre la prise de position des auteurs par rapport à la définition du contexte. En effet, selon eux "un contexte d'une interaction entre un agent et une application pour exécuter une tâche est l'ensemble des éléments de contexte instanciés et qui sont nécessaires pour supporter la tâche en cours". De même un élément de contexte est « tout morceau de donnée ou toute information utilisée pour caractériser une entité dans un domaine d'application ». Cette vision du contexte et des éléments de contexte permet de justifier le métamodèle proposé et illustré par la figure 1.3. La classe centrale de ce métamodèle est l'élément contextuel (ContextualElement) qui représente une propriété caractérisant une entité contextuelle (ContextualEntity). Une entité contextuelle est une représentation concrète d'un objet du monde réel jugé pertinent pour décrire un domaine. Une entité contextuelle est utilisée pour identifier des individus avec des caractéristiques similaires décrits avec les attributs encapsulés. Les éléments contextuels sont identifiés à partir des attributs de l'entité et des relations avec son entourage. Chaque élément contextuel doit avoir au moins une source pour l'acquisition des données constituant les valeurs de l'instance. Un élément de contexte peut être construit par une aggrégation d'éléments de contexte.

Le caractère distinctif de ce métamodèle est l'utilisation de la notion de "focus" pour déterminer les éléments contextuels à considérer. Le focus est défini par [Brésillon et Pomerol, 1999] comme étant « une étape pendant l'exécution d'une tâche pour résoudre un problème ou prendre une décision ». Dans cet article, le focus est déterminé par la tâche et l'agent qui l'exécute et qui peut être une personne, un groupe de personnes, un processus ou aussi un agent logiciel. Ainsi, pour les auteurs, le focus représente l'intérêt accordé par un agent aux éléments du contexte qui lui permettent d'exécuter cette tâche ou de prendre une décision selon un rôle prédéfini. C'est le focus qui identifie la pertinence des éléments contextuels.

Le métamodèle comporte une partie dynamique qui représente l'aspect comportemental du contexte. La dynamicité est claire avec l'usage du concept de focus qui est très dynamique et dépendant de la tâche en cours. De plus, la notion de règle est utilisée ici pour représenter la réactivité du système à certaines conditions. La partie action des règles sert à mettre à jour des données du

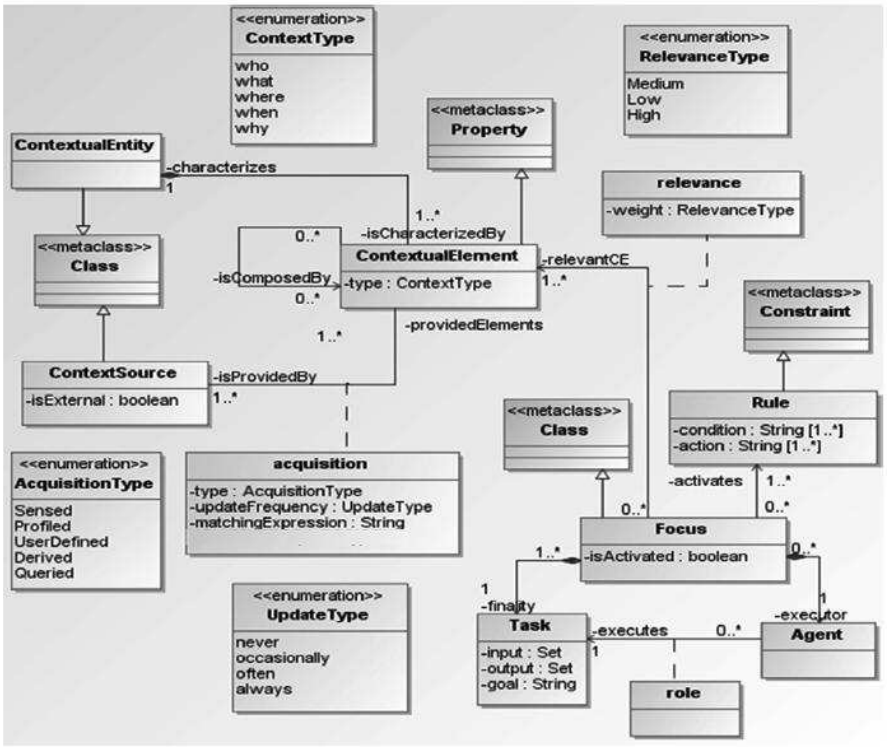


FIGURE 1.3 – Métamodèle avec focus [Vieira *et al.*, 2010]

contexte (notamment des éléments de contexte et des mesures de pertinence entre le focus et les éléments contextuels).

Il est clair que ce métamodèle est assez original dans la mesure où il utilise les notions de focus et de règles. Cependant, la nature du concept focus est réduite à une relation entre un agent et une tâche exécutée alors que le focus dépend lui même du contexte de l'agent qui dépasse le cadre de l'activité exécutée (environnement, profil de l'agent, etc). De plus la notion de source de contexte est très abstraite et elle mélange des sources hétérogènes (capteurs physiques, bases de données, Interaction Homme-Machine (IHM), etc). Enfin, l'usage des règles doit être approfondi et plus fondé. La classification et l'énumération des types de contexte et des fréquences de mise à jour à partir des sources de contexte sont trop strictes pour être appliquées dans des applications particulières.

Métamodèle de type profil UML

Dans [Simons et Wirtz, 2007] un profil UML est utilisé pour adapter le métamodèle Meta Object Facility (MOF) à la modélisation du contexte. Ce profil UML appelé Context Modeling Profile (CMP) étend les métaclasses du MOF en définissant des stéréotypes appropriés à la gestion et au raisonnement sur le contexte. Le métamodèle proposé est composé de trois parties visant : le typage du contexte, la modélisation des relations entre éléments du contexte et la modélisation des situations contextuelles. Ces différents composants du métamodèle sont représentés respectivement par les 3 diagrammes de la figure 1.4.

Concernant le typage des éléments du contexte, dans la figure 1.4a, un stéréotype "ContextItem" est utilisé pour faire référence aux éléments de contexte. Le stéréotype "ContextItemEnum" est une extension de la métaclasse "Enumeration" du MOF pour permettre une énumération des types d'éléments de contexte. Finalement, un nouveau stéréotype "CompositeType" est utilisé pour désigner

les types composites de données référant aux propriétés et aux attributs des classes utilisées dans le modèle de contexte.

Concernant la modélisation des relations entre concepts du contexte, des stéréotypes sont proposés dans la figure 1.4b. "ContextAssociation" est le stéréotype qui étend la métaclasse association pour désigner les liens entre les éléments de contexte "ContextItems" et qui peut être une association "SourceAssociation" ou une association "AccessAssociation". Le stéréotype "SourceAssociation" permet de relier les éléments de contexte à leurs sources et possède des spécialisations pour identifier la manière utilisée pour l'acquisition de la donnée contextuelle. L'acquisition de ces données peut être fournie par l'utilisateur ("userprovided"), capturée ("sensed"), ou dérivée d'autres données ("derived"). Concernant la dérivation des données contextuelles, les auteurs proposent l'utilisation de règles de dérivation qui sont des instances du stéréotype "DerivationRule". Ces règles permettent d'instancier des contraintes en Object Constraint Language (OCL) exprimant la règle appliquée pour dériver des informations.

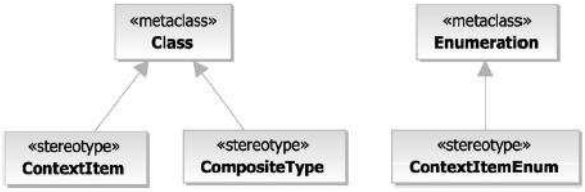
Pour la préservation des données personnelles de l'utilisateur, l'association d'accès appelée "AccessAssociation" peut prendre quatre valeurs représentées par les stéréotypes : propriétaire ("owner"), limité ("restricted"), groupe ("group") et tous ("all").

Le troisième composant du métamodèle est représenté par la figure 1.4c, il permet de supporter la modélisation des situations contextuelles. Une situation contextuelle décrit une situation spécifique qui se produit dans le cadre du système et qui doit être considérée pour appliquer une action d'adaptation. Le stéréotype "ContextConstraint" est une condition utilisée pour contraindre la situation contextuelle. Cette condition, si elle est vraie, affirme l'occurrence de la situation contextuelle qui lui correspond.

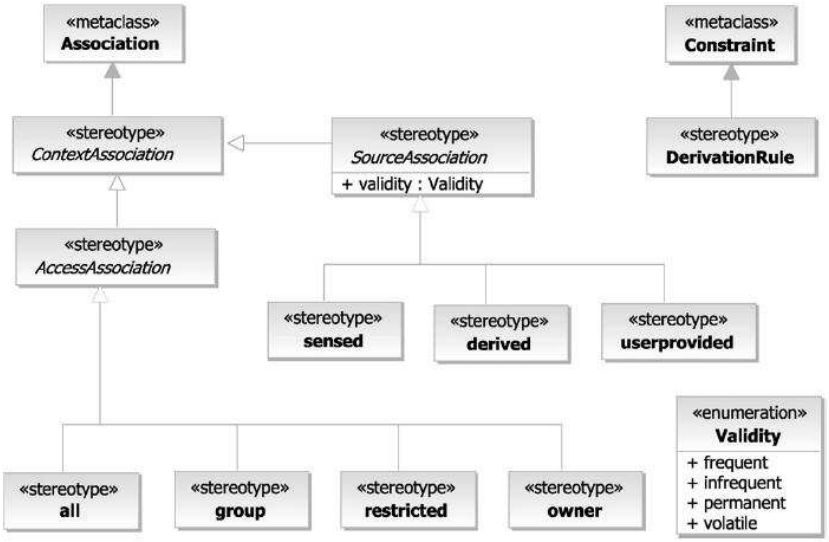
Pour résumer, il est clair que les modèles de contexte construits sur la base de ce métamodèle ont besoin de contraintes OCL pour assurer la validité du modèle. L'approche reste assez abstraite ce qui garantit bien la généricité des métamodèles. Cependant, l'usage d'un seul terme "ContextItem" pour représenter les différents concepts du contexte n'est pas très judicieux puisque ces concepts présentent des rôles différents impliqués à la construction du contexte. Certains concepts représentent des entités consommatrices (comme l'utilisateur) alors que d'autres sont simplement des objets dans l'environnement. De plus, l'usage de ces métamodèles pose un problème. En effet, malgré la variété des outils UML, ces outils ne fournissent pas de moyens standards pour accéder aux stéréotypes et forcer les contraintes définies.

Métamodèle de type framework

Dans [Achilleos *et al.*, 2010], les auteurs proposent une approche dirigée par les modèles pour la conception des applications sensibles au contexte. Cette approche s'appuie sur un métamodèle générique utilisé pour produire, par des transformations de modèles, le modèle de contexte d'une application. Ce métamodèle est représenté par la figure 1.5 et décrit les éléments de contexte, leurs propriétés et les relations qui les relient.



(a) Stéréotypes pour le typage du contexte



(b) Stéréotypes pour la modélisation des relations entre concepts du contexte

(c) Stéréotypes pour la modélisation des situations contextuelles

FIGURE 1.4 – Différentes vues du métamodèle de type profil UML [Simons et Wirtz, 2007]

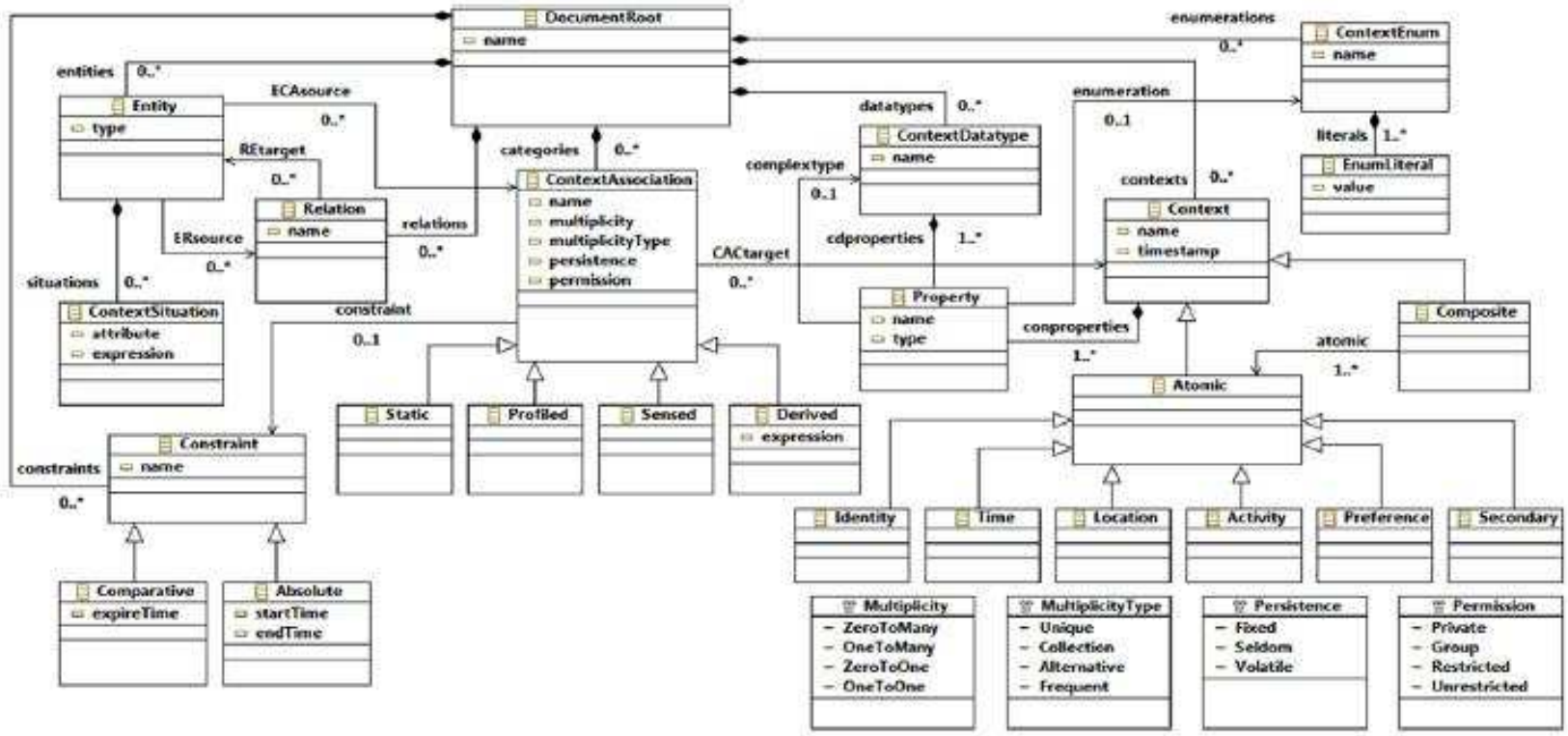


FIGURE 1.5 – Métamodèle de type framework [Achilleos *et al.*, 2010]

Dans ce métamodèle, une classe centrale "DocumentRoot" est utilisée pour jouer le rôle de conteneur de tous les éléments du contexte y compris le contexte même. Le concept entité ("Entity") est utilisé pour représenter tous les objets qui sont reliés par une association ("ContextAssociation") à une information contextuelle. Il s'agit ici des objets caractérisés par une donnée contextuelle. Pour caractériser la situation d'une entité, la métaclasse "ContextSituation" permet de représenter, avec une expression en OCL, une situation évaluée par un attribut booléen. Il est possible d'avoir des relations entre entités qui peuvent exprimer aussi des relations comportementales (utilisateur possède dispositif mobile).

La métaclasse abstraite "ContextAssociation" permet de montrer la façon d'acquérir de l'information contextuelle par l'usage des spécialisations : statique ("Static"), profilé ("Profiled"), capturé ("Sensed") et dérivé ("Derived"). Les propriétés de cette métaclasse permettent de définir les multiplicités de l'association, l'occurrence de l'information contextuelle ("MultiplicityType"), les permissions d'accès et la persistence de l'information contextuelle. L'acquisition de l'information contextuelle peut avoir une contrainte temporelle qui peut être soit un intervalle absolu soit une date d'expiration exprimée dans ce métamodèle par la métaclasse "Constraint".

L'information contextuelle est représentée par la métaclasse "Context" et elle peut être soit atomique, soit composite. Un contexte atomique contient des propriétés atomiques qui représentent l'information contextuelle de plus faible granularité. Un contexte atomique possède les spécialisations suivantes : Identité, Temps, Localisation, Activité, Préférence et Secondaire.

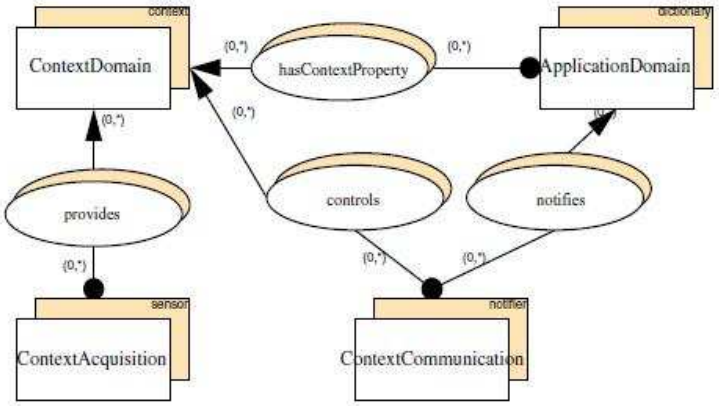
Les auteurs de ce métamodèle ont mis en évidence plusieurs aspects intéressants pour la construction des modèles de contexte tels que la définition des relations entre entités, la définition des situations contextuelles et le typage de l'information contextuelle. Cependant, nous remarquons l'introduction de certains éléments considérés comme des détails de modélisation avec d'autres éléments plus abstraits tels que les contraintes temporelles sur les informations contextuelles et le type énumération. De plus, la catégorisation de l'information contextuelle est stricte, ce qui n'est pas souhaité pour la généricité du métamodèle. Finalement le lien du contexte avec ses origines n'est pas explicité.

Métamodèle multivues avec notification

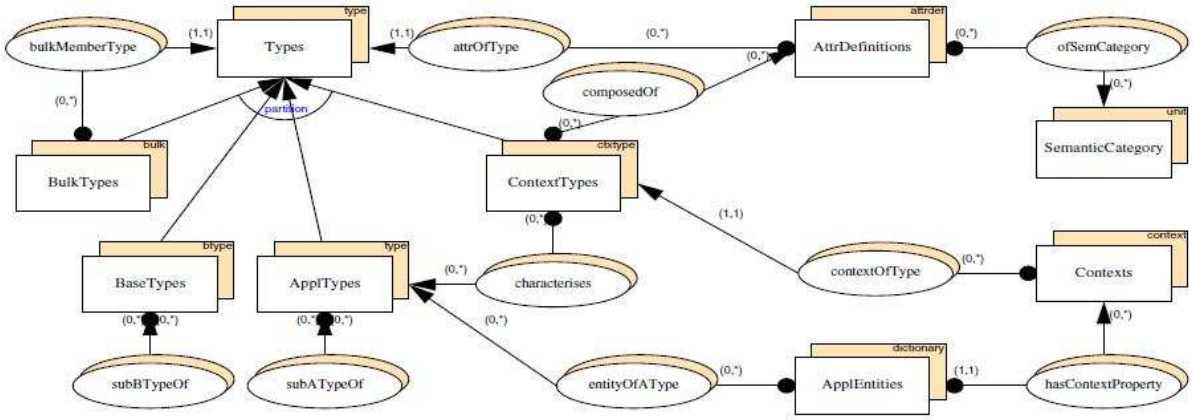
[Belotti *et al.*, 2004] propose un ensemble de métamodèles supposés être génériques pour supporter les besoins de plusieurs approches orientées contexte tout en fournissant une description sémantique de haut niveau. Ces métamodèles sont exprimés avec un modèle objet défini dans [Norrie, 1993] dans lequel un concept est représenté par un rectangle à deux niveaux : un concept abstrait concernant l'application même et un niveau inférieur représentant les définitions des objets de l'application (concept ambré). Les formes ovales représentent les associations. Le métamodèle de contexte proposé dans [Belotti *et al.*, 2004] est composé de plusieurs vues représentées séparément mais reliées l'une à l'autre. Le premier métamodèle, représenté par la figure 1.6a, met l'accent sur les concepts centraux pour la modélisation du contexte et qui sont appelés les modules du métamodèle. Le contexte est considéré comme une information caractérisant une entité. Cette information contextuelle est fournie par un module d'acquisition du contexte représentant les fournisseurs (capteurs). Les notificateurs font le lien entre les entités et leur environnement pour contrôler l'information contextuelle et notifier les entités intéressées.

La deuxième vue du métamodèle, illustrée par la figure 1.6b, représente les concepts à utiliser pour typer le contexte et ses éléments. Ce métamodèle fait la classification des types de données soit : des types de base, des types caractéristiques de l'application, des types propres au contexte ou des types composés (Bulk Types). Chaque information contextuelle est caractérisée par un type qui possède un ensemble d'attributs et qui sont classifiés dans des catégories sémantiques.

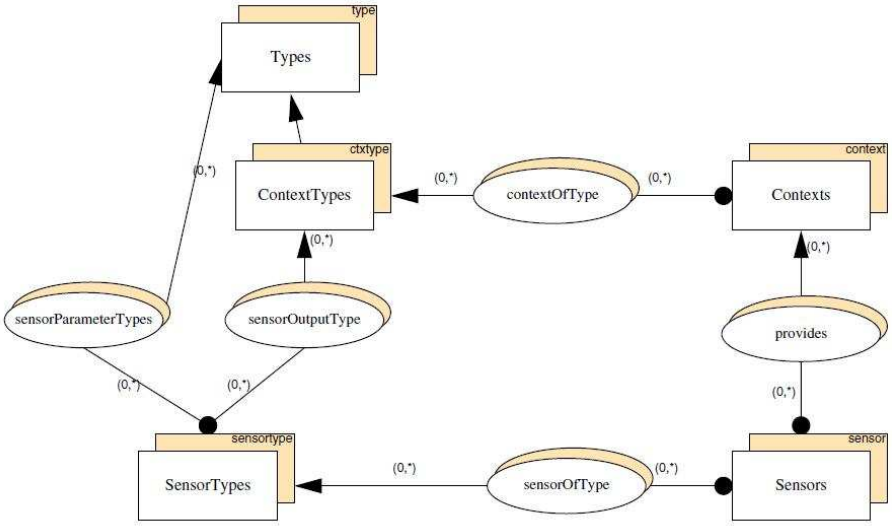
La troisième vue du métamodèle, illustrée par la figure 1.6c, représente les concepts utilisés pour



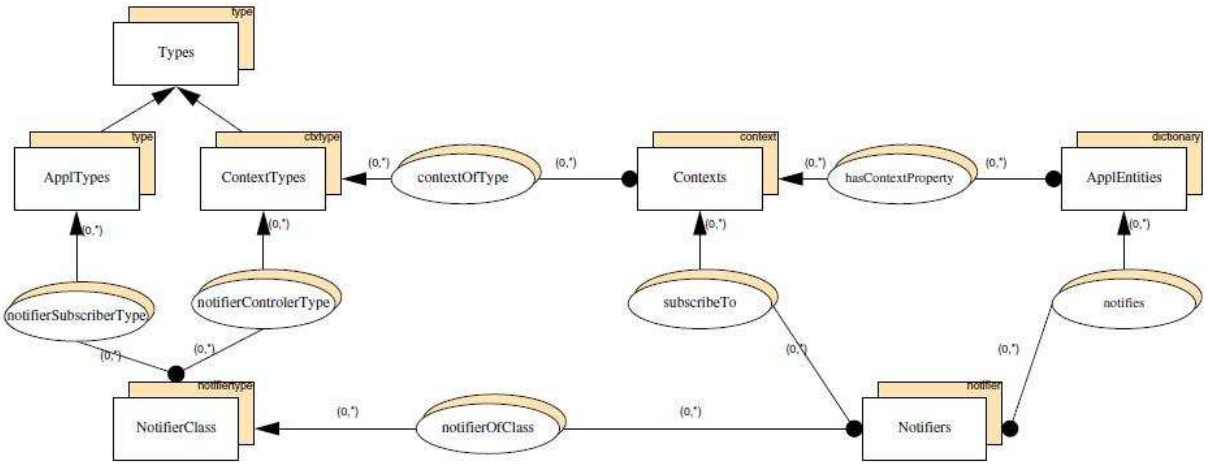
(a) Modules du contexte



(b) Typage du contexte



(c) Métamodèle d'acquisition du contexte



(d) Métamodèle de notification du contexte

FIGURE 1.5 – Métamodèle multivues avec notification [Belotti *et al.*, 2004]

l'acquisition du contexte. Le concept central ici est le capteur ayant un type et un ensemble de paramètres ainsi qu'une donnée en sortie qui est l'information contextuelle.

La quatrième et la dernière vue est représentée par la figure 1.5d. Dans cette vue, les auteurs focalisent sur la notification du contexte. Le concept clé ici est le notificateur (Notifier) qui est souscrit à des informations contextuelles et invoqué à chaque fois que ces informations changent pour notifier les entités intéressées. Le métamodèle représente des classes de notificateurs permettant la définition de plusieurs instances de notificateurs.

Ce métamodèle à quatre vues présente l'avantage de fragmenter les aspects du contexte selon le niveau d'abstraction et la nature des concepts. De plus, deux aspects intéressants ont été considérés : la notification et l'acquisition du contexte. Par contre, le langage de modélisation utilisé n'est pas un standard mais un modèle objet ancien ce qui pose un problème lors de la compréhension et de l'usage du métamodèle. De plus, nous remarquons une concentration sur le typage des concepts au profit d'autres aspects caractérisant le contexte et la sensibilité au contexte. Finalement, l'usage du terme capteur met à l'écart plusieurs modes d'acquisition du contexte très hétérogènes et qui constituent un défi à mettre en valeur par les modèles de contexte.

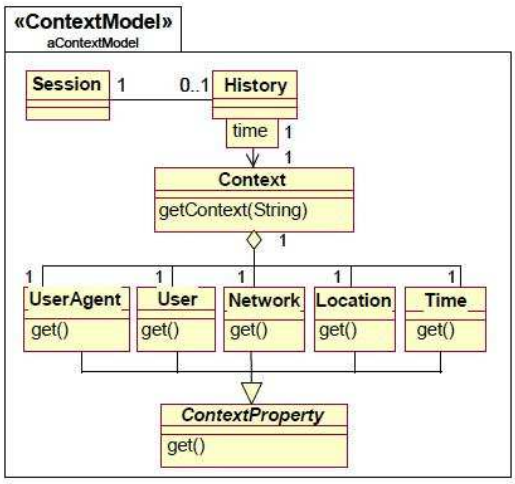
Métamodèle centré applications web et évènements

L'article [Kappel *et al.*, 2001] propose un ensemble de métamodèles pour la modélisation des applications web présentant un aspect ubiquitaire nécessitant la manipulation du contexte. Ces métamodèles sont supposés génériques c'est-à-dire indépendants du domaine. Cette approche s'appuie sur une extension d'UML pour le web, Web Unified Modeling Language (WUML). Le métamodèle proposé est composé de quatre vues visant la modélisation du contexte de haut niveau, la modélisation du profil, la modélisation des règles et la modélisation des évènements.

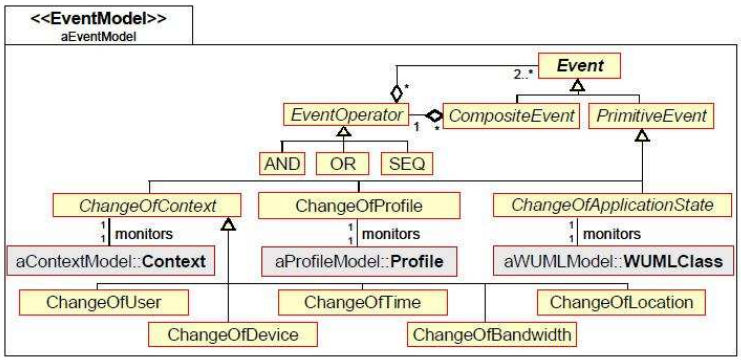
Selon les auteurs de cet article, le contexte est une réification de l'environnement par mise en œuvre de ses propriétés qui doivent être mises à jour de façon continue. D'après la figure 1.6a, le contexte est un ensemble de propriétés :

- *l'agent de l'utilisateur :* c'est le composant du système responsable de la communication avec l'application web.
- *l'utilisateur :* le consommateur du service ou de l'information.

- *le réseau de communication.*
- *la localisation de l'usager.*
- *le temps.*



(a) Métamodèle de contexte



(b) Métamodèle d'évènements

FIGURE 1.6 – Métamodèle centré applications web et évènements [Kappel *et al.*, 2001]

Le métamodèle de contexte met l'accent sur le concept de "session" qui est propre au domaine des applications web. Chaque session a un contexte qui correspond au contexte actuel obtenu par l'intermédiaire du port "temps". Ce port est associé à la classe "historique" qui est une séquence de plusieurs contextes indexés par le temps. Le métamodèle de profil est décrit comme un ensemble de propriétés semblables à celles du modèle de contexte. L'extension de ces métamodèles est possible par le concepteur de l'application web pour ajouter des propriétés au contexte ou au profil.

Concernant les règles, elles sont de la forme Event Condition Action (ECA) et possèdent de plus, un ensemble de propriétés (priorité, état, etc.).

La figure 1.6b illustre le métamodèle d'évènements nécessaire pour l'utilisation des règles ECA. Ce métamodèle fait la différence entre un évènement primitif et un évènement composite. Les évènements primitifs peuvent représenter un changement dans l'état de l'application, un changement de profil ou un changement de contexte. Ainsi chaque type d'évènement primitif influence un modèle : le modèle WUML, le modèle de profil ou le modèle de contexte.

Ces métamodèles forment un ensemble cohérent de techniques supportant les applications web

ubiquitaires indépendamment des domaines et prend en compte la manipulation des règles et des évènements. De plus, l'utilisation de la notion d'historique rappelle le besoin et l'importance de cette notion dans le raisonnement sur le contexte et la prise de décision dans certaines applications. Par contre, nous pensons que ces vues sont insuffisantes et négligent des aspects comme l'acquisition et la manipulation de l'information contextuelle. De plus, la relation entre information contextuelle et entités du système est masquée.

Métamodèle orienté service

L'article de [de Farias *et al.*, 2007] propose un métamodèle de contexte, illustré par la figure 1.7 et exprimé en UML. L'information contextuelle est obtenue par l'association d'attributs à des entités. De même les entités peuvent être reliées entre elles par le même type d'association ("CWAssociation"). Les associations peuvent être profilées, capturées ou dérivées, ce qui permet de déduire l'origine, la persistance, le niveau de confiance et la temporalité des informations contextuelles. L'usage de l'association dérivant des informations contextuelles s'appuie sur une expression en OCL pour transformer cette information, ce qui facilite son abstraction.

Ce métamodèle est complété par une vue orientée service (voir figure 1.8). Cette vue représente les concepts reliés à l'acquisition de l'information contextuelle par un service. Un fournisseur peut avoir plusieurs services parmi lesquels certains sont responsables de collecter et distribuer l'information contextuelle sous forme de messages. Ces messages sont reliés à leur tour à des entités et des attributs (les porteurs de l'information contextuelle).

L'approche de [de Farias *et al.*, 2007] est générique et puissante. En effet, elle permet de dériver des informations sur la nature et la validité de l'information contextuelle à partir des types d'associations. De plus, elle permet de façon intelligente de supporter à la fois la modélisation du contexte et du domaine sans alourdir le métamodèle. Cependant, le métamodèle est incomplet à cause de sa restriction à la vision orientée service qui ne supporte pas d'autres modes d'acquisition du contexte, notamment l'usage des évènements. De plus, la notion d'attribut reste abstraite et mérite d'être davantage qualifiée pour ne pas considérer que tout attribut est une information contextuelle.

1.2.3 Évaluation des métamodèles de contexte

Pour l'évaluation des métamodèles de contexte, nous nous sommes inspirés du framework de Kappel et al. [Kappel *et al.*, 2003] qui propose un cadre pour évaluer les applications ubiquitaires selon deux axes : la gestion du contexte et l'adaptation (voir figure 1.9). L'axe concernant le contexte présente des critères sur la modélisation et la gestion du contexte comme la composition du contexte en propriétés ("Property") extensibles ("Extensibility"), la prise en compte de données passées ou futures du contexte ("Chronology"), la période de validité d'une donnée de contexte ("Validity"). D'autres critères concernent la représentation du contexte comme la réutilisation avec d'autres applications ("Reusability") et la possibilité de représenter des données de haut niveau ("Abstraction"). L'acquisition du contexte est un critère important pour représenter le degré d'automatisation de l'acquisition des données qui peut aller du manuel à l'automatique ("Automation") et la fréquence d'acquisition du contexte ("Dynamicity"). Le dernier critère, dans l'axe du contexte, concerne les mécanismes d'accès au contexte ("Mechanism"). L'axe horizontal concerne l'adaptation au contexte dans les applications ubiquitaires et qui ne nous intéresse pas pour l'évaluation des métamodèles présentés ci-dessus.

De l'axe contexte de la figure 1.9, nous adoptons les besoins de modélisation du contexte pour évaluer les métamodèles décrits précédemment. Certains de ces critères d'évaluation sont traités différemment par rapport à l'approche de [Kappel *et al.*, 2003] afin qu'ils prennent en compte nos besoins d'un métamodèle de contexte applicable dans une démarche de conception de SI ubiquitaires.

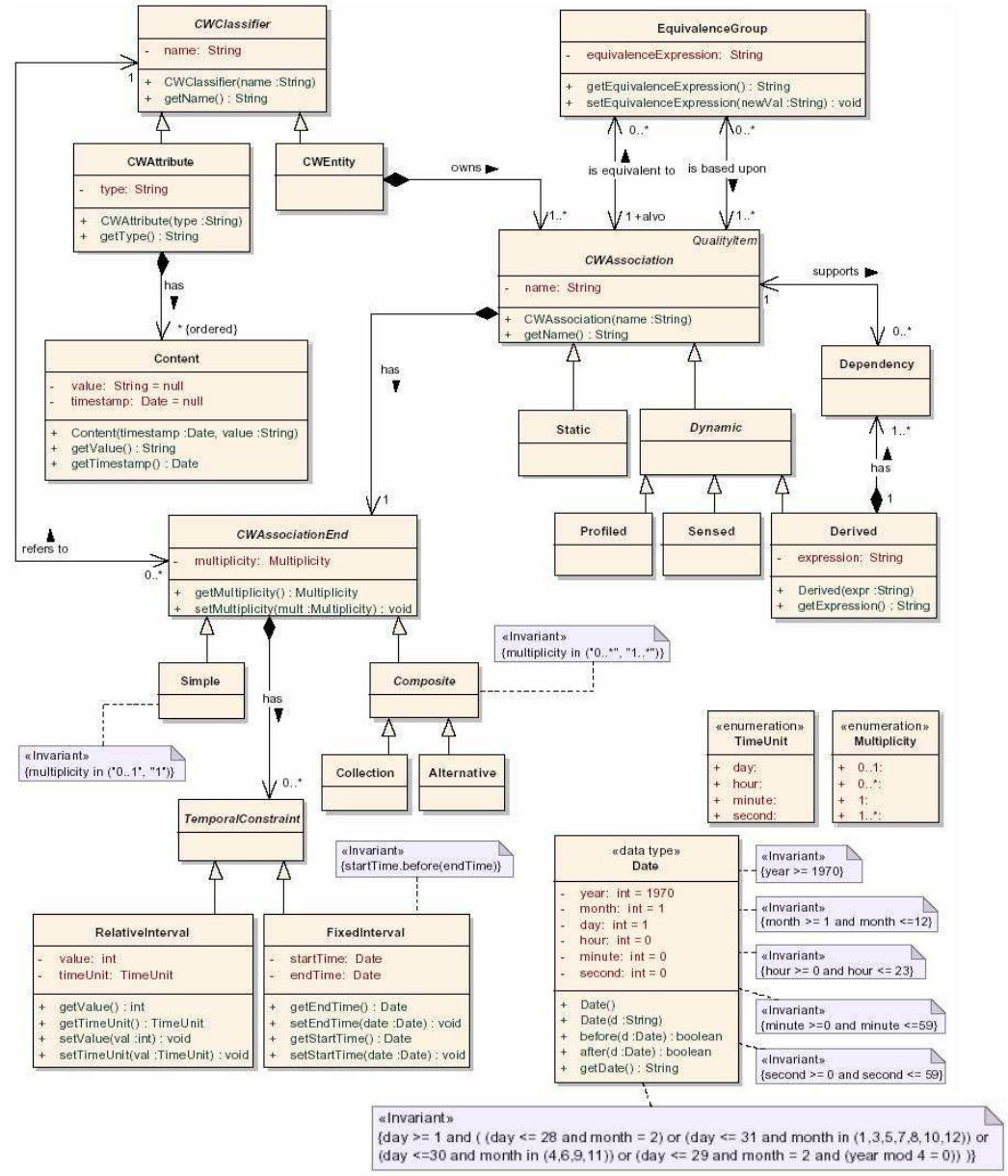


FIGURE 1.7 – Métamodèle de contexte [de Farias *et al.*, 2007]

Les critères retenus sont utilisés pour la comparaison des métamodèles dans un tableau présenté ci après. Mais tout d'abord nous présentons ces critères et leur signification.

Nous nous sommes particulièrement intéressés à la position des approches étudiées par rapport au concept "contexte" (qui diffère d'une approche à une autre) et à la possibilité d'étendre les propriétés du contexte, ce qui constitue une mesure de généricité. La prise en compte des contraintes temporelles et des valeurs passées ou futures (de l'information contextuelle) est un aspect important aussi pour concevoir le modèle de contexte générique et facile à implémenter. La validité de l'information

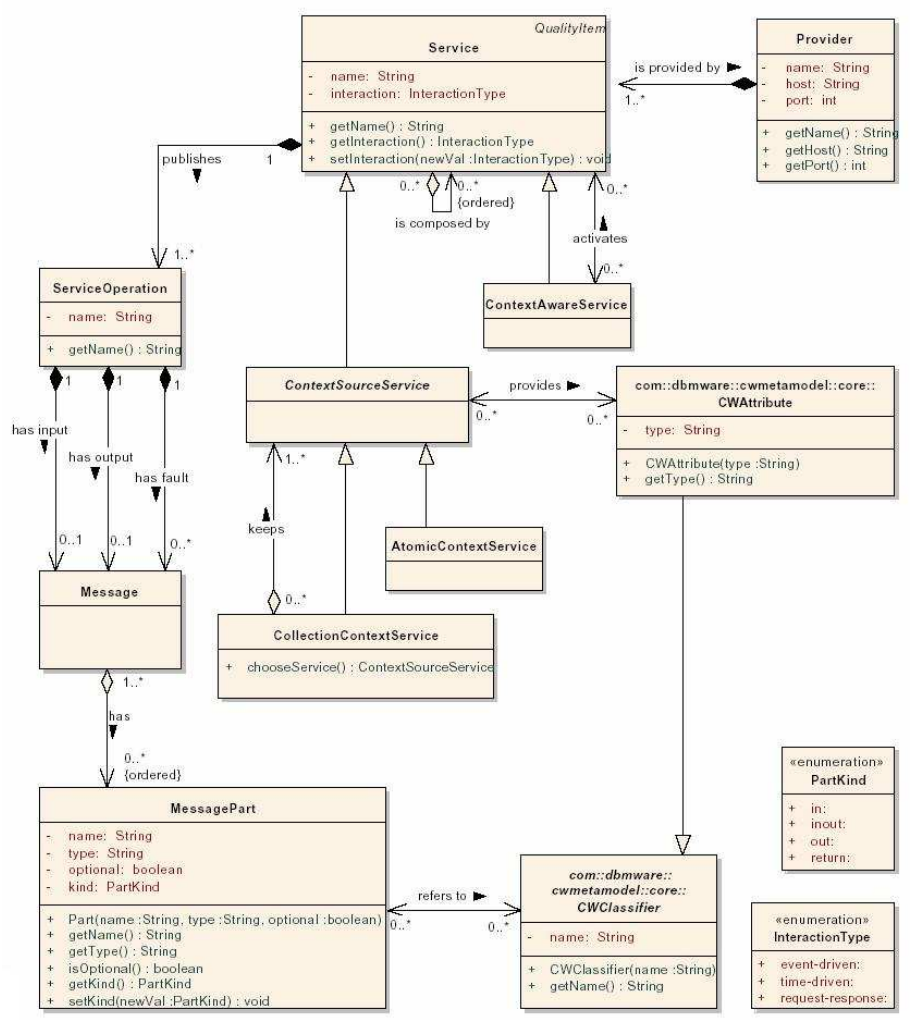


FIGURE 1.8 – Vue service du métamodèle de contexte [de Farias *et al.*, 2007]

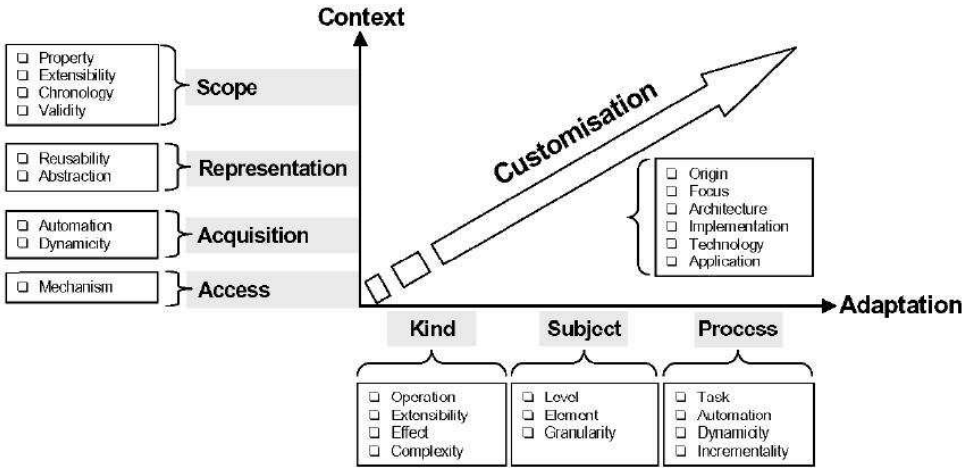


FIGURE 1.9 – Framework d'évaluation proposé par [Kappel *et al.*, 2003]

contextuelle représente la justesse de l'information qui supporte dans la plupart des cas des erreurs ou des incohérences.

La distinction et la prise en compte du modèle doit être claire et lisible dans le métamodèle de contexte. Il est de plus nécessaire de connaître l'origine de l'information contextuelle et les évènements qui déclenchent des traitements et des changements de cette dernière. D'un autre côté, l'accès au contexte et la confidentialité de l'information font partie de l'aspect sécurité du contexte considéré dans plusieurs travaux comme un besoin et une exigence des SI ubiquitaires. Finalement, il est préférable que le métamodèle de contexte ouvre des horizons sur les usages possibles et la manipulation de l'information contextuelle.

Tous ces critères prédéfinis sont souhaités pour un métamodèle de contexte. Une comparaison des métamodèles étudiés précédemment est établie dans le tableau 1.1.

TABLE 1.1 – Comparaison entre les métamodèles de contexte

Travaux	Position du contexte	Extensibilité du contexte	prise en compte des historiques	Prise en compte des valeurs futures	Contraintes temporelles	Validité du contexte	Modèle de domaine	Acquisition du contexte	Prise en compte des évènements	Accès au contexte	Usage du contexte
[Vieira *et al.*, 2010]	Le concept de contexte est utilisé implicitement comme ensemble d'éléments de contexte	oui	non	non	implicite par l'usage des mises à jour et du focus	non	non	oui	non	non	non
[Simons et Wirtz, 2007]	Une situation contextuelle est un ensemble d'éléments de contexte	oui	non	non	non	non	non	oui	non	oui	non
[Achilleos *et al.*, 2010]	Le contexte est une information atomique ou composite	non	non	non	oui	oui	oui	oui	non	oui	non
[Belotti *et al.*, 2004]	Le contexte est une propriété atomique associée à une entité	oui	non	non	non	non	non	oui	non	implicite par souscription	Notification
[Kappel *et al.*, 2001]	Le contexte est un ensemble de propriétés finies à étendre si besoin	non	oui	non	oui	non	non	oui	oui	non	Modèle de règle ECA
[de Farias *et al.*, 2007]	Le contexte est l'ensemble des attributs associés à une entité	oui	non	non	oui	dérivée de la nature de l'association	oui	oui	non	non	non

1.3 Besoins en sensibilité au contexte

Comme vu dans l'introduction de ce chapitre, la sensibilité au contexte est un besoin qui intéresse plusieurs spécialités de l'informatique. D'un autre côté, la nature du cycle de vie du contexte fait intervenir plusieurs disciplines allant de l'infrastructure des systèmes à la modélisation conceptuelle. Aujourd'hui les interactions avec les systèmes mobiles et ambiants fournissent des fonctionnalités intelligentes qui expliquent le succès de plusieurs nouveaux dispositifs grâce aussi bien au hardware adapté (l' i-Phone, l' i-PAD, les écrans interactifs ...) qu'au logiciel adapté (les systèmes navigationnels, les GPS, les moteurs de recherche personnalisés ...).

Cette section se concentre sur les approches d'adaptation des fonctionnalités et des caractéristiques des systèmes ubiquitaires par prise en compte du contexte. En effet, cette capacité de capturer le contexte et de se l'approprier a soutenu l'évolution des techniques intelligentes d'adaptation mises en place et exploitées par plusieurs niveaux logiques des systèmes. Nous désignons par niveaux logiques les différentes couches qui constituent un système informatique dans une logique N tiers. Nous retenons quatre couches qui adoptent la sensibilité au contexte. La couche métier vise une adaptation des PM, des workflows et des services web au contexte (de l'utilisateur ou de l'application). La couche présentation cherche à adapter les interactions avec les utilisateurs en fonction du contexte. La couche gestion des communications cherche à fournir des accès personnalisés aux ressources. Finalement, la couche gestion de données permet avec la sensibilité au contexte de filtrer et d'adapter le contenu des données.

Plus de détails sur les besoins d'adaptation et les approches qui les supportent dans ces différentes couches sont étudiés ci-après. Nous nous intéressons à la vision du contexte dans chaque couche, à la modélisation du contexte et à l'usage du contexte pour chaque besoin d'adaptation spécifique à une couche.

1.3.1 Adaptation des métiers par la prise en compte du contexte

La couche métier dans une organisation présente les logiques métier sous forme de PM et de modèles métier reproduisant la structure stratégique de l'entreprise. Un processus métier est "un ensemble d'activités, entreprises dans un objectif déterminé. La responsabilité d'exécution de tout ou partie des activités par un acteur correspond à un rôle. Le déroulement du processus utilise des ressources et peut être conditionné par des événements, d'origine interne ou externe. L'agencement des activités correspond à la structure du processus" [Morley *et al.*, 2005]. Le concept Business Process Management (BPM) fait référence à une approche de gestion des processus métier visant la promotion et l'innovation des processus.

L'environnement évolutif et compétitif des entreprises impose un besoin croissant d'évolution et de flexibilité des métiers. En effet, les processus métier doivent s'adapter aux exigences des clients et subir un processus d'amélioration continue avec des mesures de performance. De plus, l'émergence des applications ubiquitaires fait surgir de nouveaux besoins tels que la mobilité, la sensibilité au contexte et les interactions avec l'environnement ambiant.

La question qui se pose à ce stade est la suivante : comment intégrer ces besoins d'adaptation dans les modèles et les méthodes de conception des PM ? Dans cette section, nous étudions la prise de position de l'état de l'art par rapport à l'intégration de la sensibilité au contexte dans les approches BPM. Plusieurs approches proposent des solutions pour adapter les PM et les workflows au contexte et augmenter ainsi la flexibilité du métier. Mais dans ce domaine, l'adaptation ne possède pas toujours le même objectif. Les différents objectifs visés sont catégorisés dans [Smanchat *et al.*, 2008] :

- la personnalisation (Customisation) : ce type d'adaptation est déclenché par un changement dans le contexte de l'utilisateur (faisant partie du processus) relatant un changement dans ses

besoins. Ainsi, une nécessité d'adapter le processus à ces nouveaux besoins surgit.
- La correction : le changement du contexte d'un processus peut aussi influencer la validité et la structure de ce dernier qui ne devient plus valable. Ainsi, une action corrective doit être prise pour adapter le processus et rectifier sa structure.
- L'optimisation : l'adaptation peut avoir pour but d'optimiser les performances d'un processus en prenant en compte les données historiques de ces processus et les étudier en vue d'améliorer certains aspects comme le temps d'exécution ou le travail collaboratif.

L'adaptation des PM qui nous intéresse dans ce contexte de recherche est la personnalisation qui concerne les SI ubiquitaires. Nous présentons dans la suite différentes approches liées à ce type d'adaptation.

Règles métier

L'approche des règles métier dans l'ingénierie des SI répond à certains besoins des spécialistes métier pour ajouter une dimension de flexibilité aux métiers [Valatkaite et Vasilecas, 2005]. Cette approche bénéficie d'un grand intérêt accordé par les travaux de recherche. Ceci est justifié par le grand nombre de standards et d'outils sur la gestion et l'exécution des règles métier. Citons parmi ces outils ARIS Business Rule Designer et Websphere ILOG JRules de IBM. Une règle métier est définie dans [Business Rule Group, 2000] comme « une déclaration qui définit ou contraint certains aspects du métier ; elle vise à supporter une structure métier ou à contrôler ou influencer le comportement du métier ».

Les règles métier sont exprimées soit sous la forme d'une contrainte, soit sous la forme « Si Condition Alors Action ». Elles sont très utiles et très présentes dans les systèmes appliquant massivement des politiques métier et des processus de décision comme les domaines des assurances, de la banque, et des systèmes d'aide à la décision ou de recommandation. Aujourd'hui, elles sont utilisées dans les systèmes ubiquitaires et mobiles pour adapter des processus collaboratifs au contexte des participants. Selon la classification de Wagner [Wagner, 2005], les règles métier peuvent être :
- *des règles d'intégrité :* ce sont des contraintes ou des assertions qui doivent être satisfaites. Par exemple, « le voyageur doit être abonné pour permettre l'inscription au service ».
- *des règles de dérivation :* ce sont des déclarations permettant de dériver des concepts. Par exemple, « un bus est disponible s'il n'est pas en maintenance et s'il n'a pas un plan de circulation ».
- *des règles de production :* ce sont des règles contenant une condition et l'action correspondante. Par exemple, « si le voyageur est une personne à mobilité réduite alors vérifier l'équipement "handicap" du bus ».
- *des règles de réaction :* ce sont des règles déclenchées par un (ou des) évènement(s) pour générer une action. Par exemple, « un évènement retard doit déclencher un processus de calcul d'itinéraire ».
- *des règles de transformation :* ce sont des règles qui contrôlent des changements d'états des entités dans le système. Par exemple, « la localisation du voyageur doit être mise à jour de façon continue ».

Dans une perspective ubiquitaire, ces règles métier sont utilisées pour contraindre différents aspects du métier. En effet, elles peuvent porter sur le métier (ses stratégies et son organisation), on parle de règles globales. Elles peuvent aussi porter sur les processus métier pour influencer leurs structures. Finalement, elles peuvent concerner les workflows. Elles peuvent être raffinées du niveau global jusqu'au niveau workflow.

Côté implémentation, les règles métier ne doivent pas être intégrées dans le code exécutable comme c'est le cas dans plusieurs systèmes. L'idéal est qu'elles soient séparées dans un composant

indépendant en vue de faciliter la maintenance et les changements de politiques. De plus, ceci permet d'augmenter la flexibilité du système et la transparence des règles métier. Dans la pratique, l'exécution de la règle métier est partagée entre deux sous-systèmes : un moteur d'évènements et un système de workflow.

Vision du contexte dans les PM

L'adaptation au contexte est un besoin exprimé par plusieurs travaux de recherche qui mettent l'accent sur la nécessité de fournir une prise en compte plus explicite des données de l'environnement dans l'enchaînement d'un processus. Mais le besoin d'une vue générale du contexte est essentielle pour mieux comprendre le contexte des PM. Dans [Rosemann *et al.*, 2008], les auteurs proposent par exemple un framework "Onion framework" qui représente le contexte d'un processus sous forme de couches superposées (voir la figure 1.10).

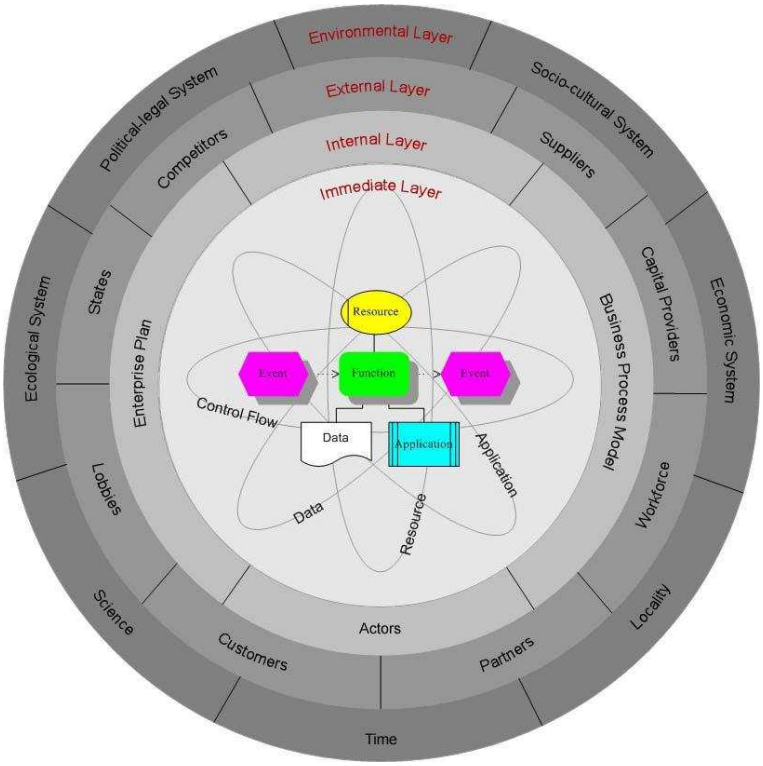


FIGURE 1.10 – Framework par couches du contexte des PM [Rosemann *et al.*, 2008]

Ces quatre couches sont identifiées en fonction de leur proximité par rapport au cœur métier.

- *Le contexte immédiat :* il comprend les éléments faisant partie des constructeurs du PM : il s'agit des éléments nécessaires pour la compréhension et l'exécution du PM (les données, les ressources organisationnelles, les étapes du processus, les applications reliées, etc).
- *le contexte interne :* le PM fait partie d'un système plus large : l'organisation. Plusieurs éléments de l'organisation peuvent avoir une influence sur le PM tels que ses normes, ses ressources, ses stratégies, ses acteurs, etc.
- *Le contexte externe :* il contient les éléments en dehors du cadre de l'organisation mais qui font partie de l'environnement métier de l'organisation. Ces éléments peuvent avoir une influence sur la façon de concevoir et d'exécuter les PM : il s'agit des parties prenantes (fournisseurs, clients, partenaires, concurrents), de l'état du marché, etc.

- *Le contexte environnemental :* il contient des éléments non compris dans l'environnement métier de l'organisation mais qui peuvent tout de même affecter le cœur métier. Il peut contenir des éléments de l'environnement social, économique et politique de l'organisation ou encore des données écologiques (météo) et technologiques.

Ce framework fournit une taxonomie qui peut être utilisée pour identifier, classifier, comprendre et intégrer les informations contextuelles dans les modèles de PM.

Cependant, il n'est pas toujours facile d'identifier les différents éléments de contexte influençant les PM. [Rosemann *et al.*, 2008] et [Ploesser *et al.*, 2010] proposent l'utilisation des objectifs des PM pour identifier le contexte du PM et les besoins d'adaptation. Ils subsument que chaque objectif d'un PM permet de capturer des besoins d'adaptation ou simplement de répondre à la question "est-ce que tel élément de contexte est pertinent pour le processus ?". Ainsi, un métamodèle de PM orienté objectif est utilisé dans [Rosemann *et al.*, 2008] et il a été amélioré dans [Ploesser *et al.*, 2010].

Contexte dans les langages de modélisation des PM

La modélisation d'un PM permet de décrire son fonctionnement en définissant l'ensemble des activités à exécuter et leur ordre d'exécution. Deux grandes catégories de langages de modélisation de PM existent : les langages impératifs et les langages déclaratifs.

Langages impératifs. Ce sont des langages qui définissent, en même temps que les activités à exécuter, un ordre précis qui doit être respecté en suivant des flux de contrôle explicites. Plusieurs langages impératifs sont connus dans la littérature comme BPMN, le diagramme d'activité d'UML ou encore Yet Another Workflow language (YAWL) (Yet Another Workflow language).Ces langages permettent une modélisation de haut niveau. D'autres langages impératifs sont conçus pour l'exécution des PM tels que Business Process Execution Language (BPEL) (Business Process Execution Language) et XML Process Definition Language (XPDL) (XML Process Definition Language).

Les langages impératifs permettent en général de représenter le contexte immédiat du processus qui est constitué des activités, des données, des participants et des évènements qui peuvent influencer l'enchaînement du processus. Ces langages sont généralement associés à des métamodèles qui capturent les différents constructeurs des PM. Parmi ces métamodèles, nous citons le métamodèle de l'OMG Business Process Definition Metamodel (BPDM) . Un autre métamodèle est proposé par [Ben Cheikh *et al.*, 2010d], il regroupe les différents composants d'un PM dans cinq vues : la vue intentionnelle, la vue organisationnelle, la vue fonctionnelle, la vue comportementale et la vue interactionnelle.

Les langages et les modèles impératifs de PM ne permettent pas de supporter directement des informations externes au contexte immédiat. Ceci explique la rigidité de ces processus qui ne peuvent pas s'adapter à des informations externes. Cependant, les règles métier combinées avec ces langages permettent d'ajouter un espace de flexibilité. Mais au niveau exécution, ces règles sont souvent mélangées au code ce qui complique la maintenance et les évolutions.

Langages déclaratifs. Les langages déclaratifs se concentrent sur "ce qui devrait être fait" plutôt que sur "quoi faire". Pour cela, des contraintes sont utilisées pour décrire le fonctionnement et l'enchaînement du processus. La seule exigence est de respecter ces contraintes, ainsi plusieurs possibilités d'exécution peuvent surgir. En effet, tous les chemins qui ne violent pas les contraintes sont autorisés, ce qui multiplie et diversifie les structures des processus à l'exécution. La structure du processus ne peut donc être déterminée qu'à l'exécution. Dans cette perspective, les langages déclaratifs sont considérés comme un moyen pour rester abstrait au niveau de la modélisation et éviter ainsi d'énumérer explicitement les différents scénarios possibles. Parmi les langages déclaratifs, nous citons : ConDec,

PENELOPE, BPtrigger, EPC (Event Processing Chain), FLOWER ou encore EM-BrA^{2}CE (qui utilise le vocabulaire standard de l'OMG Semantics of Business Vocabulary and Business Rules (SBVR)).

Une des approches utilisées pour modéliser la logique déclarative des processus est le paradigme à base de règles métier (voir section 1.3.1.0) impliquant un ensemble de règles et un moteur d'inférence pour l'exécution des activités. Ces règles permettent de raisonner sur des données de l'environnement métier ou ambiant exprimées sous la forme de conditions contextuelles.

La nature variée des langages impératifs et des langages déclaratifs mettent en évidence plusieurs différences dans leur pouvoir d'expression ou la manière de les concevoir et de les utiliser par la suite. Tandis que la définition des flux de contrôle est explicite dans les langages impératifs, elle est implicite pour les langages déclaratifs. Par ailleurs, l'approche déclarative permet l'expression d'évènements complexes et la réaction à des situations complexes, ce qui n'est pas le cas des langages impératifs. Une comparaison plus approfondie de ces deux types de langages peut être consultée dans [Goedertier et Vanthienen, 2007] et [Schonenberg *et al.*, 2008] .

Approches d'adaptation des PM

Plusieurs travaux de recherche se sont concentrés sur la proposition de solutions pour l'adaptation des PM ou la définition de PM sensibles au contexte. Selon [Smanchat *et al.*, 2008], il existe trois niveaux pour introduire l'adaptation dans les PM :

- Niveau abstrait : permet l'adaptation du processus au niveau modèle en déterminant les détails de l'implémentation selon le contexte au moment de l'exécution.
- Le niveau concret : permet d'introduire l'adaptation au niveau workflow qui correspond à une définition exécutable du processus.
- Le niveau instance : permet d'adapter une instance d'un workflow alors que la définition du workflow reste inchangée.

Les travaux de recherche qui s'inscrivent dans ce cadre proposent tous d'appliquer des changements à une définition d'un processus. Dans l'approche "CAWE" (Context-Aware Workflow Execution) [Ardissono *et al.*, 2007] le processus contient une ou plusieurs activités abstraites qui peuvent être remplacées par une ou plusieurs implémentations concrètes prédéfinies en vue d'adapter le service à l'utilisateur. Par ailleurs, [Narendra *et al.*, 2005] propose l'utilisation de points de variation représentant des parties du processus où une sélection de choix est nécessaire pour l'exécution. Cette approche prend en compte le contexte d'une ressource pour redémarrer l'exécution d'une tâche si la ressource n'est plus disponible. Une approche de modélisation de la variabilité appliquée aux PM a également été proposée dans [Ben Cheikh *et al.*, 2009b].

Une autre solution qui va dans le même sens est proposée par [Modafferi *et al.*, 2005] . Elle consiste à utiliser des régions sensibles au contexte (Context Sensitive Region) avec des points de migration pour aller d'un processus prédéfini à une implémentation spécifique à un contexte.

Dans [Kumar et Yao, 2009] les auteurs utilisent des règles métier en parallèle avec des processus impératifs en vue d'augmenter la flexibilité de ces derniers. Ces règles utilisent une même taxonomie que les processus afin de produire des actions qui modifient les constructeurs des processus (ajouter ou sauter une activité, modifier des rôles, modifier des flux ou des données, etc) en fonction de certaines conditions.

Dans la thèse de Boukhebouze [Boukhebouze, 2010], une approche orientée règles est proposée pour modéliser certains aspects du processus qui n'est défini que partiellement. L'application des règles

se fait au niveau instance du processus pour spécifier certains comportements. L'auteur propose une extension du formalisme ECA appelée Evènement- Condition - Action - Post condition - post Evènement (ECAPE) pour décrire explicitement les évènements provoqués par l'exécution de la partie action. Pour modéliser ces processus, un nouveau modèle ECAPE-M, utilisant des séquences de règles, est proposé.

Toutes ces approches présentent des solutions différentes, mais qui se ressemblent par le principe de définir des régions pour appliquer une certaine adaptation au contexte quelle que soit sa nature. Ces approches varient dans l'usage des langages impératifs et déclaratifs. Nous sommes convaincus que l'adaptation au niveau métier souffre d'un manque de standards et d'approches solides qui soient applicables sur tous les systèmes ubiquitaires et qui soient supportés par des outils performants.

1.3.2 Adaptation des interactions Homme-Machine

La sensibilité au contexte peut être utilisée pour adapter des interfaces homme-machine. Ainsi, la manipulation et la gestion du contexte intéressent les chercheurs en IHM pour construire des interfaces plastiques, c'est-à-dire des interfaces adaptées au contexte (par adaptabilité ou adaptativité). De nos jours, la flexibilité et la plasticité des IHM deviennent une nécessité avec l'émergence de nouvelles technologies d'affichage et d'interaction qui peuvent s'intégrer dans l'environnement ambiant.
Pour comprendre comment cette adaptation est perçue et conçue dans cette discipline, nous rappelons que l'IHM repose sur un ensemble de modèles permettant de formaliser les étapes de conception des interfaces. Ces modèles sont :

- *le modèle de tâches :* une représentation de haut niveau des activités (tâches) réalisées par l'utilisateur pour atteindre un but en interagissant avec une application.
- *le modèle de domaine :* un modèle indépendant de l'application, mais permettant de décrire le domaine de l'application. Il représente les objets du domaine impliqués dans l'application et les liens entre eux.
- *le modèle de dialogue :* il représente les entrées et les sorties en informations, la structure des objets d'interaction et le comportement dynamique. Il permet d'abstraire les détails du style d'interaction [Traetteberg et Krogstie, 2008].
- *le modèle d'interface concret :* il permet de décrire les interactions abstraites en détail en se référant à un style particulier d'interaction ou à une plateforme particulière [Traetteberg et Krogstie, 2008].

Ces différents modèles utilisés pour la construction d'une IHM ont été formalisés et décrits avec un métamodèle proposé dans [Sottet *et al.*, 2008]. Ce métamodèle montre l'organisation de ces différents modèles, leurs liens et les mapping qui peuvent exister pour passer d'un modèle à un autre. Pour construire des interfaces, les concepteurs des IHM partent en général des modèles métier représentant les fonctionnalités d'un système ou d'une application pour extraire les modèles de tâches et les modèles de dialogue [Guerrero García *et al.*, 2009]. Cependant, la sensibilité au contexte nécessite de nouvelles techniques que nous présentons dans la suite.

Vision du contexte

Selon la vision de Weiser [Weiser, 1991], les objets de la vie quotidienne deviennent des supports d'interaction. Ainsi, les dispositifs d'interaction se diversifient par leur forme et leur finalité : les ordinateurs, les TV, les téléphones, les PDA, etc. Ces dispositifs supportent des fonctionnalités de plus en plus nombreuses et utilisent des techniques intelligentes d'interaction. La plasticité des interfaces nécessite la découverte dynamique des plateformes d'interaction et de leurs caractéristiques.

La miniaturisation des dispositifs d'interaction facilite la mobilité des supports d'interaction et ainsi la possibilité d'interagir avec le système à partir de lieux différents. Par conséquent, le système doit reconnaître ces différents lieux et adapter ses paramètres en fonction des conditions contextuelles (en réunion, en voiture, chez soi). De plus, aujourd'hui, il est possible d'utiliser des systèmes interactifs pour une communication plus intelligente en déduisant des commandes avec des perceptions gestuelles, corporelles ou encore faciales.

Ainsi, les IHM sont fortement influencées par les nouvelles technologies selon trois axes : les méthodes d'acquisition du contexte, les données contextuelles plus novatrices et les modes de communication avec l'utilisateur. Dans le framework Caméléon [Calvary *et al.*, 2003], le contexte a été considéré comme la composition d'un triplet « Utilisateur, Plateforme, Environnement ».

Contexte dans les métamodèles de présentation

De même que pour les autres disciplines, la métamodélisation du contexte pour l'adaptation des interfaces permet de produire des modèles de contexte uniformes et de faciliter l'identification des éléments de contexte. Cependant, cette dernière n'a pas bénéficié d'un grand intérêt, ce qui explique le nombre réduit de métamodèles de contexte dans cette discipline. Ci-dessous nous présentons une description de certains métamodèles intégrant la sensibilité au contexte avec des constructeurs IHM.

Métamodèle orienté rôle [Rey, 2005] Ce métamodèle montre une modélisation du monde réel qui met en évidence des entités (êtres vivants ou objets inertes) intervenant dans l'application. L'utilisateur est une entité réalisant une activité qui est un ensemble de tâches courantes et de tâches de fond. Les entités sont reliées entre elles et peuvent jouer des rôles. Le réseau de contexte est un ensemble de contextes et le contexte est un ensemble de situations où une situation représente des liens entre des entités jouant certains rôles.

Métamodèle orienté plateforme [Vanderdonckt *et al.*, 2008] Ce métamodèle utilise la notion de modèle de contexte pour désigner un ensemble de contextes utilisé comme modèle d'interface. Le contexte est une information appartenant au triplet « Utilisateur, Plateforme, Environnement ». La plateforme est un ensemble de plateformes logicielles situées sur une surface faisant partie de l'environnement. Une plateforme logicielle est une surface d'interaction possédant une certaine forme et fournissant des services gérés par des évènements, des conditions et des messages et réalisés par des rôles.

Nous constatons que dans ces métamodèles le lien entre contexte et interaction est représenté différemment. Le premier métamodèle fait apparaître la notion de tâche et d'activité de l'usager alors que le second montre le lien entre éléments d'interaction et contexte. Le premier métamodèle met en évidence la notion d'entité alors que le second fait une classification des informations contextuelles.

Approches d'adaptation d'interfaces

Pour améliorer la qualité des interactions homme-machine, le contexte est exploité à la fois dans les étapes amont du processus de conception d'un système tout comme à l'exécution. Parmi les premiers travaux qui proposent un cadre formel pour le développement des interfaces plastiques et adaptées au contexte, le framework Caméléon [Calvary *et al.*, 2003] est composé de trois parties :

- un ensemble de modèles ontologiques représentant les modèles conceptuels descriptifs de l'application (modèle de domaine, modèle de tâche, modèle de contexte et modèle d'adaptation). Ces modèles permettent de produire des modèles prédictifs qui servent de point d'entrée au processus de conception.

- un processus de développement permettant de spécifier les étapes de conception et la navigation entre ces étapes : modèle de concepts et de tâches, interface abstraite, interface concrète et interface finale.
- un processus d'exécution montrant la transformation des interfaces utilisateurs pour cibler un nouveau contexte au moment de l'exécution (runtime).

Ce framework ne permet pas de fournir une démarche précise et rigide, mais plutôt un cadre formel souple pour définir des démarches.

Une approche différente est proposée dans [Sousa *et al.*, 2007] qui supporte la conception de méthodes d'ingénierie des interfaces utilisateur en utilisant USIXML dans une approche dirigée par les modèles. Dans cette approche, il est possible de créer des démarches de conception sous forme de processus BPMN en respectant les objectifs d'utilisabilité définis en amont.

Enfin, la démarche proposée dans [Hariri *et al.*, 2009] respecte trois niveaux : niveau abstrait, niveau concret et niveau final. Elle est basée sur l'usage des patrons qui permettent l'intégration de l'adaptation à la démarche. Chaque patron est responsable d'une action sur l'interface qui est déclenchée en fonction des changements de contexte. Ces actions peuvent être d'ordre métier (adapter le modèle de tâche) ou d'ordre présentation (adapter l'affichage).

Le nombre réduit d'approches témoigne du manque de démarches de conception d'interfaces plastiques qui décrivent de façon exhaustive le passage des modèles conceptuels au code en intégrant la sensibilité au contexte.

1.3.3 Adaptation pour le contrôle d'accès

La protection des données et des services dans les SI constitue une exigence importante à prendre en compte. Le contrôle d'accès permet de définir les droits de consultation, d'usage et de modification des ressources du SI. Le contrôle d'accès peut être appliqué à plusieurs couches du SI : pour sécuriser l'accès aux PM, pour sécuriser l'accès à des services et pour sécuriser l'accès aux données.

Différents modèles ont été proposés pour gérer le contrôle d'accès. Les premiers modèles étaient des modèles discrétionnaires Discretionary Access Control (DAC) visant à accorder des permissions prédéfinies à chaque sujet et chaque objet. Les deuxièmes types de modèles concernent le contrôle d'accès mandataire Mandatory Access Control (MAC) qui ne permet pas la migration des accès aux flux de données d'un niveau moins sensible à un niveau plus sensible. Une autre vision orientée tâche Task Based Access Control (TBAC) a marqué ce domaine. Elle consiste à accorder à l'utilisateur un ensemble de permissions dépendantes de la tâche en cours. Si la tâche est activée alors il peut profiter de ses permissions. Finalement, le modèle de contrôle d'accès qui a eu le plus de succès et qui a été largement adopté est le modèle à base de rôles Role Based Access Control (RBAC). Dans ce modèle, le concept de rôle est utilisé comme un intermédiaire entre les acteurs et les permissions. Les permissions sont accordées à des rôles activés par les acteurs durant une session. Des contraintes peuvent être appliquées également quand il s'agit d'assigner des rôles, des permissions et des sessions. La gestion des rôles (identification, hiérarchisation, clustering, fouille) représente un nouveau courant de recherche étroitement lié à RBAC.

Vision du contexte

Avec les besoins croissants d'ubiquité et de mobilité, les utilisateurs accèdent depuis n'importe quel endroit et n'importe quand, à des données de sources diverses et hétérogènes. Ainsi, le contrôle d'accès doit prendre en compte le contexte de l'utilisateur pour gérer les accès et préserver la confidentialité des données. Nous présentons dans la suite les approches permettant de répondre à ce besoin, mais, tout d'abord, nous nous focalisons sur la vision du contexte dans ces approches. Plusieurs so-

lutions ont été proposées dans la littérature pour accorder des permissions d'accès selon le contexte des utilisateurs. Dans ces solutions, des perceptions différentes du contexte sont apparues. Dans [Filho et Martin, 2009], l'information contextuelle peut être : spatiale, temporelle, spatio-temporelle, sociale et d'exécution (dispositif, environnement informatique, bande passante). Le contexte manipulé peut concerner le propriétaire de l'information à protéger, le demandeur de l'information et le contexte de la ressource demandée. Une ontologie représentant cette structure de contexte est également fournie où un contexte d'accès est associé avec un contexte contenant les dimensions précitées.

Le contexte est considéré comme une association (localisé à, utilise) entre une entité (une personne ou un lieu) et un élément de contexte (activité, statut et localisation). Les contraintes temporelles peuvent constituer aussi une dimension importante du contexte à considérer pour définir des accès (durée de l'activité, moment de consultation, etc.) comme dans les approches [Mossakowski *et al.*, 2003] et [Joshi *et al.*, 2005]. D'autres approches [Adaikkalavan et Chakravarthy, 2006] proposent la prise en compte des évènements pour gérer les accès sachant que l'évènement est porteur d'une information contextuelle fortement dynamique. C'est la nature de l'application qui permettra de définir quelles données contextuelles doivent être considérées pour accorder des permissions d'accès.

Approches de contrôle d'accès contextuel

L'usage des données contextuelles dans le contrôle d'accès nécessite la considération d'un modèle de contrôle d'accès pour lui appliquer des techniques d'assignation contextuelle de permission. Étant donné le succès reconnu des approches RBAC, plusieurs solutions ont été proposées pour enrichir RBAC dans le but de supporter des contraintes contextuelles. Comme vu précédemment, RBAC supporte la définition de contraintes pour l'assignation de rôles, de sessions et de permissions. Ces contraintes ont évolué pour intégrer des données contextuelles. Dans cette perspective, plusieurs approches se partagent le même principe d'appliquer des règles ou des contraintes contextuelles au modèle RBAC. Cependant, la forme de ces règles et leur résultat diffèrent d'une approche à une autre. Dans [Wilikens *et al.*, 2002], des règles d'autorisation multi-niveaux utilisent des évènements, une hiérarchie de rôles et des attributs des utilisateurs pour accorder dynamiquement des rôles et des permissions concernant les trois niveaux : fonctions, ensembles de fonctions et données.

L'approche RBAC étendue dans [Kumar *et al.*, 2002] est appelée CS-RBAC (Context Sensitive -RBAC). Elle utilise des filtres de contexte utilisant des données sur le contexte de l'utilisateur, du rôle et de l'objet pour assigner des rôles ou accorder des permissions. Le contexte d'un rôle est ici une combinaison du contexte de l'utilisateur et de l'objet.

Une approche similaire xoRBAC est proposée dans [Strembeck et Neumann, 2004] et consiste à appliquer des règles pour accorder des permissions, mais elle va jusqu'à la proposition de démarches pour la spécification des contraintes contextuelles ou pour l'acquisition du contexte. Dans [Kim *et al.*, 2005] une matrice SCM (State Checking Matrix) est utilisée pour évaluer les conditions contextuelles pour assigner des permissions.

L'approche de [Kulkarni et Tripathi, 2008] appelée CA-RBAC (Context Aware RBAC) considère deux types d'objets : partagés et privés. L'accès aux rôles, aux permissions et aux objets publics fait intervenir des règles supportant des conditions contextuelles. D'autres contraintes contextuelles sont appliquées pour l'accès aux ressources où il est possible d'avoir accès à des opérations sans avoir accès aux ressources correspondantes.

Une autre approche consiste à utiliser directement des règles pour accorder des permissions à des personnes avec ou sans passer par le concept de rôle. Il s'agit des modèles de contrôle d'accès RuleBAC (Rule Based Access Control) [Axel Kern, 2005] et Attribute RBAC [Al-Kahtani et Sandhu, 2002] qui utilisent les attributs des utilisateurs (leurs données personnelles) pour évaluer des règles afin d'autoriser l'accès ou non. Ces règles sont de la forme Condition Action. Les attributs des utilisateurs

peuvent contenir des données contextuelles, ce qui permet un accès sensible au contexte.

Enfin, la vision de COSMOS [Bellavista *et al.*, 2003] est différente des autres et met le contexte au centre de l'approche. Le contexte est composé de trois parties : une vue désirée du contexte, une vue autorisée par l'utilisateur et, à l'intersection de ces deux vues, une vue activée (pendant le processus de contrôle d'accès). Le système utilise un ensemble de métadonnées contenant le profil de l'utilisateur, les vues et les paramètres de confidentialité fixés par l'utilisateur pour évaluer des politiques d'autorisation et d'obligation.

Toutes ces approches intègrent l'usage de règles pour accorder l'accès ou non. Ces règles peuvent être appelées contraintes, politiques ou filtres. Un système ubiquitaire soucieux de la sécurité doit avoir une politique de contrôle d'accès contextuelle et solide surtout pour des applications d'assistance personnelle (voyage, transport, finance, médical) et pour les SI de domaines sensibles (médicaux, bancaires, administratifs) qui intègrent de plus en plus les besoins ubiquitaires.

1.3.4 Sensibilité au contexte pour la gestion des données

La sensibilité au contexte concerne à ce niveau la livraison d'informations adaptées au contexte de l'utilisateur. En effet, aujourd'hui, l'accès à l'information utile n'est pas toujours évident à cause de la grande quantité des informations et de l'hétérogénéité de leurs sources. Ainsi, dans plusieurs SI le problème de l'accès à l'information doit bénéficier d'un ensemble de techniques permettant d'optimiser le temps, l'effort et le coût.

Plusieurs domaines sont concernés par ce besoin d'adaptation de l'information tels que la recherche d'information, la fouille de données, le e-commerce et les services web. Dans tous ces domaines, il est nécessaire de proposer des résultats pertinents et utiles à l'utilisateur en un temps court, ce qui constitue une mesure d'utilisabilité et de performance.

Les applications qui communiquent avec des SI pour raisonner sur certaines données et fournir des services à l'utilisateur s'intéressent à injecter de l'adaptation dans les données manipulées. C'est par exemple le cas des applications d'assistance comme les applications de suivi médical ou les assistants de voyage ou de transport.
Ces applications sont de deux types [Tchienehom, 2006] :

- des applications reposant sur une approche "service au comptoir" ou pull qui consistent à renvoyer des informations répondant à une demande explicite d'un individu.
- des applications reposant sur une approche "service à domicile" ou push qui consistent à renvoyer automatiquement à un individu des informations qui pourraient l'intéresser, sans qu'il en fasse explicitement la demande.

De manière générale, l'adaptation du contenu nécessite l'usage d'un modèle de contexte servant à raisonner sur les informations collectées pour déduire la situation contextuelle. A partir de la base de données, il est possible de sélectionner le contenu en adéquation avec le contexte et de le délivrer aux parties intéressées. Les techniques de collecte du contexte et d'adaptation du contenu caractéristiques de ce domaine sont décrites ci-dessous.

Vision du contexte

La personnalisation est la forme d'adaptation utilisée à ce niveau. Elle concerne d'abord la collecte des données sur le contexte de l'utilisateur groupées sous la notion de profil et qui permettent d'adapter l'information en fonction de ce profil. La collecte du contexte peut se faire de façon explicite, avec des capteurs ou fournie par l'utilisateur, ou de façon implicite en étudiant les actions de l'utilisateur pour déduire ses préférences ou ses intérêts particuliers.

Ainsi, le défi majeur que rencontrent ces applications est de construire le profil de l'utilisateur. Chaque application de ce type possède un ou des modèle(s) de profils. Les données que peuvent contenir ces modèles peuvent avoir des origines diverses. Aujourd'hui, plusieurs techniques sont utilisées pour capturer ces données :

- les historiques : ils sont utilisés pour sauvegarder les actions de l'utilisateur telles que les traces de navigation et les requêtes qu'il aurait exprimées. Plusieurs approches utilisent des caches pour sauvegarder l'historique qui sera utilisé pour déduire les préférences et les habitudes de l'utilisateur. La gestion et la mise à jour des historiques sont réalisées par des algorithmes spécifiques [Perich *et al.*, 2004].
- l'auto-évaluation : l'utilisateur est invité à évaluer certaines opérations ou certaines données proposées par l'application pour exprimer son niveau de satisfaction. Ceci permet de déduire les préférences et les intérêts de l'utilisateur. D'autres techniques capturent le contexte de la personne et utilise cette évaluation pour déduire les intérêts dépendants du contexte. Ainsi, le contexte de la personne peut être enrichi davantage.
- la saisie explicite du profil : l'utilisateur saisit par lui même son profil, y compris ses préférences. Il peut spécifier des préférences qui dépendent du contexte comme dans le cas du paramétrage d'un téléphone.

Une autre classification des profils est fournie par [Chevalier *et al.*, 2007] qui fait la distinction entre le profil long terme et le profil court terme (la durée d'une session) et entre le profil positif et le profil négatif (des informations qui n'intéressent pas l'utilisateur).

Approches de personnalisation des informations

Les approches de personnalisation des informations utilisent différentes techniques décrites ci-dessous. Une analyse plus approfondie de ces techniques est fournie dans [Anand et Mobasher, 2005].

1. *Le filtrage basé contenu :* il consiste à trier le contenu intéressant avec des méthodes permettant de mesurer l'intérêt de l'utilisateur. Ensuite, des techniques classiques de recherche d'informations sont appliquées pour récupérer le contenu pertinent en adéquation avec les intérêts. Des systèmes de recommandation comme Web-Watcher [Mladenic, 1996] et Letizia [Lieberman, 1995] ont utilisé cette technique de filtrage.
2. *Le raisonnement basé règles :* il consiste à utiliser des règles pour construire des modèles de préférences et le contenu à délivrer comme action de la règle. Ainsi des règles permettent d'exprimer, pour un contexte déterminé, une caractérisation du contenu à délivrer. [Perich *et al.*, 2004] propose une approche basée règles.
3. *La co-occurrence pour le filtrage :* elle consiste à détecter les patrons qui se répètent dans l'historique d'un utilisateur particulier ou des utilisateurs en général pour déduire des habitudes. Les patrons permettent de prévoir les comportements de l'utilisateur si l'application est mono-utilisateur ou de prévoir l'attitude de nouveaux utilisateurs si l'application est multi-utilisateurs. Un exemple est fourni dans [Xu *et al.*, 2008].
4. *Le filtrage collaboratif :* il consiste à collecter les opinions des utilisateurs et à mesurer les taux de satisfaction pour recommander un certain contenu. Cette approche est très utilisée dans les systèmes de recommandation sur le web et dans le domaine du e-commerce [Schafer *et al.*, 1999].
5. *La souscription sensible au contexte :* c'est une forme de livraison de l'information en prenant en compte le contexte de l'usager de façon paramétrable par l'utilisateur même. En effet, l'utilisateur définit un ensemble de souhaits de souscription à certaines informations dans certains contextes prédéfinis. Le changement de ces informations déclenche une notification de

l'utilisateur respectant les conditions sur le contexte. Un exemple de système est fourni dans [Giannakopoulos et Palpanas, 2009].

Il est toutefois possible d'avoir des approches hybrides qui combinent deux techniques ou plus à la fois.

1.3.5 Synthèse

La définition de modèles flexibles diffère d'une couche du SI à une autre. Ceci peut s'expliquer par les objectifs différents de l'adaptation et la nature de l'information contextuelle utile dans chaque couche. Un système ubiquitaire et sensible au contexte peut comporter plusieurs besoins d'adaptation (appartenant à différentes couches et dépendant de la nature même de l'application), tel est le cas de l'approche proposée par [Chaari *et al.*, 2007] pour injecter de l'adaptation des services, du contenu et de la présentation au contexte dans les applications ubiquitaires. Ainsi, la comparaison des besoins et des approches d'adaptation dans ces couches s'avère utile pour développer des applications ubiquitaires. Le tableau 1.2 présente une comparaison des modèles concernés par l'adaptation, l'information contextuelle que ces modèles utilisent, la manière de les acquérir et les approches d'introduction de l'adaptation (identifiées jusqu'à présent en littérature), dans les quatre couches du SI (métier, présentation, contrôle d'accès et gestion des données).

1.4 Synthèse

Les systèmes ubiquitaires doivent répondre au défi d'adaptation et de personnalisation pour garantir à l'utilisateur un service performant et répondant à ses besoins tout en respectant la sécurité et la confidentialité de ses données personnelles. Ce défi est réalisable avec la sensibilité au contexte qui consiste à capturer le contexte ambiant, permettant ainsi de le reconnaître et de l'utiliser pour raisonner de façon appropriée et réagir à certaines situations. Ainsi les systèmes ubiquitaires doivent supporter des techniques et des méthodes qui soutiennent le bon déroulement du cycle de vie du contexte.

Ce chapitre met l'accent sur ce défi d'adaptation des SI ubiquitaires qui intéresse un grand nombre de disciplines. Il s'intéresse en particulier aux deux phases en aval du cycle de vie du contexte qui correspondent à la modélisation et à l'usage du contexte. Dans cette perspective et pour la clarification des différents vocabulaires utilisés dans cette tendance de recherche, ce chapitre donne une présentation des différents concepts utilisés et présente notre prise de position par rapport à ces concepts et aux liens sémantiques qui peuvent les lier. Cette étude nous permet d'aboutir à la conclusion suivante :

> « la flexibilité est un besoin général des systèmes informatiques et peut avoir différentes formes telles que l'adaptabilité, l'adaptativité, la personnalisation et la sensibilité au contexte qui est une technique supportant les trois premières dans une vision ubiquitaire ».

Pour prendre en compte la sensibilité au contexte il est essentiel d'avoir un modèle de contexte pour permettre la reconnaissance des informations provenant de sources variées et hétérogènes. Ces modèles de contexte peuvent être des modèles objets, des modèles ontologiques ou autres. Cependant, ces modèles sont insuffisants : il est nécessaire de dispose d'un métamodèle générique permettant d'instancier plusieurs modèles de contexte interopérables et en harmonie. Les approches de la littérature qui proposent ce type de métamodèles sont récentes et justifiées par le fait que ce besoin devient plus prégnant. Ce chapitre présente ces métamodèles et fait une comparaison selon des critères importants pour la complétude et la généricité du métamodèle. Nous remarquons qu'en général, le support de l'acquisition du contexte et de son usage, s'il est abordé, n'est pas bien fondé.

TABLE 1.2 – Comparaison entre les besoins d'adaptation dans les couches du SI

Couche d'adaptation	Modèle concerné ou langage	Contexte utile	Collecte du contexte	Approche d'adaptation
Métier	langages de PM (ou de workflow) déclaratifs ou impératifs	Toute information contextuelle peut être utile (contexte, immédiat, interne, externe et environnemental)	Explicite (fournie ou détectée par capteur)	– Injection de règles métier contextuelles ou autre format de règles – Usage de la variabilité et points de variations – Définition de plusieurs versions de fragments de PM
Présentation	Modèle de tâches, interfaces abstraites, concrètes ou finales	Environnement + Utilisateur (expressions gestuelles et faciales) + Plateforme	Explicite	– Définition de plusieurs versions de modèles (avec transformation de modèles) – Usage de patrons contextuels
Contrôle d'accès	modèle de rôles RBAC	Toute information contextuelle de la ressource, du demandeur et du propriétaire (y compris l'environnement spatio-temporel)	Explicite	– Enrichissement de RBAC avec des règles – Application de règles sans paradigme RBAC (règles de formats variés)
Gestion de données	Push ou Pull	Contexte du destinataire de l'information (y compris ses préférences et intérêts)	Explicite ou implicite (sauvegarde des actions de l'utilisateur)	– Filtrage basé contenu – Usage des règles – Filtrage par co-occurrence de patrons – Filtrage collaboratif – Souscription sensible au contexte

L'usage du contexte concerne principalement la prise en compte du contexte pour un des objectifs suivants : l'adaptation des métiers, l'adaptation de l'affichage et des interfaces hommes-machines, le contrôle des accès sensibles au contexte et la personnalisation des informations et du contenu délivrés. L'usage du contexte est très varié selon la diversification des objectifs, ce qui explique les différentes techniques de raisonnement et de manipulation du contexte. Or plusieurs systèmes ubiquitaires combinent deux ou plusieurs besoins d'adaptation différents à la fois. Ceci nécessite l'usage d'une approche solide supportant les différentes étapes du cycle de vie du contexte. Nous verrons dans le chapitre suivant dans quelle mesure les approches classiques de développement des SI sont capables de prévoir cette prise en compte du contexte et s'il existe des approches de conception des SI ubiquitaires. Nous montrerons la place du contexte dans ces approches et dans quelle mesure les autres besoins des SI sont pris en compte.

Chapitre 2

L'ingénierie des SI ubiquitaires

La construction des applications ubiquitaires a mis en évidence plusieurs défis qui doivent être considérés dans les SI qui les supportent. Ces défis représentent aussi les critères spécifiques de l'ubiquité qui réalisent, s'ils sont satisfaits, des systèmes ubiquitaires réussis et utilisables. Ces critères sont les suivants :

- **la mobilité :** représente la possibilité de changer la localisation d'un ou plusieurs acteurs ou systèmes interagissant avec l'application, y compris l'application même qui peut être installée sur un dispositif mobile. La mobilité induit, dans ce sens, un changement de l'environnement spatial d'une entité qui peut influencer le comportement et la structure de l'application. Elle est à l'origine d'autres critères, qui sont la portabilité et la dynamicité. En effet, une application mobile doit être portable, c'est-à-dire doit opérer dans des conditions particulières et restreintes de mémoire, de puissance de processeur et d'interfaces. La dynamicité, dans le cadre de la mobilité, signifie la réaction aux changements de l'environnement et la détection des évènements ambiants. Elle met en évidence des problématiques de sensibilité à la localisation en relation avec l'axe temporel.
- **la distribution :** représente la possibilité de faire communiquer des systèmes géographiquement éloignés ce qui nécessite des architectures flexibles et dynamiques et la garantie d'une connectivité fiable et sécurisée. La distribution fait surgir des besoins comme l'hétérogénéité et la scalabilité. En effet, les systèmes reliés représentent des caractéristiques de traitements (langages, systèmes d'exploitation et formats de données) très hétérogènes qui doivent être pris en compte dans un système ubiquitaire. Concernant la scalabilité, elle doit supporter un nombre potentiel d'utilisateurs en prévoyant les architectures et les capacités de traitement nécessaires.
- **la sécurité :** représente un besoin primordial des applications ubiquitaires qui collectent et manipulent des données personnelles des usagers. Ainsi, dans un système ubiquitaire, la confidentialité de ces données doit être garantie.
- **la sensibilité au contexte :** représente la capacité de collecter des données concernant le contexte de l'application ou des acteurs avec lesquels elle interagit et les prendre en compte pour fournir ses services. Ce critère a été traité exhaustivement dans le chapitre précédent.

Ces critères ont été considérés (en totalité ou en partie) dans les travaux existants sur les systèmes ubiquitaires. Les approches orientées événements se sont intéressées particulièrement à une combinaison de ces critères pour fournir des bases conceptuelles et architecturales aux systèmes ubiquitaires qui sont la mobilité, la dynamicité, la distribution, l'hétérogénéité, la scalabilité et la sensibilité au contexte (acquisition du contexte). Ainsi, les approches basées évènements représentent un cadre approprié pour concevoir des applications ubiquitaires (voir la section 2.1).

Mais l'usage de ces approches s'avère insuffisant pour fournir des guides méthodologiques pour le développement des SI ubiquitaires. Ce qui nous mène vers les méthodes d'ingénierie des SI pour étudier leur adéquation pour le développement de ces systèmes et la prise en compte des critères ubiquitaires dans leurs démarches (voir section 2.2).

Dans la dernière section de ce chapitre (section 2.3), nous étudions les démarches de développement de systèmes ubiquitaires et faisons une comparaison selon des critères de performance pour évaluer leur aptitude à faciliter et couvrir tout le processus de développement.

2.1 Approches orientées évènements

L'évolution des systèmes informatiques d'aujourd'hui est fortement influencée par les effets de la mise en réseau et de la connectivité. Par conséquent, un intérêt croissant est accordé aux systèmes distribués qui doivent surmonter des défis comme la scalabilité et l'hétérogénéité. A ce stade, la notion d'intergiciel (middleware) a été introduite pour faciliter la communication entre les parties hétérogènes d'un système distribué. Ces intergiciels sont utilisés aussi bien pour gérer des communications de type Client/Serveur qui supportent des interactions de type Requête/Réponse, que pour gérer des communications Publish/Subscribe à base de flux d'évènements.

Dans les systèmes basés évènements, les composants communiquent en produisant et recevant des notifications d'évènements. Ces composants sont totalement découplés et aucun composant ne connaît l'autre. Par conséquent, les approches orientées évènements résolvent les problèmes d'hétérogénéité et de scalabilité.

D'après la vision du diagramme d'états d'UML, où un évènement est une transition entre deux états, un évènement est une occurrence intéressante temporellement qui exprime le changement d'état d'une entité. Dans le livre de Luckham "The power of events" [Luckham, 2002], un évènement est l'enregistrement d'une activité, alors que dans [Chakravarthy *et al.*, 1994] « un évènement est une occurrence atomique (se produit entièrement ou pas du tout) ». Toutes ces définitions montrent la nature variée d'un évènement, qui explique les usages divers de ce concept dans les systèmes informatiques. En effet, les évènements dépassent le cadre d'un simple moyen de communication entre systèmes pour devenir un concept de base au sein des composants mêmes de ces systèmes. Aujourd'hui, les évènements sont utilisés dans la couche métier du système, pour interagir avec l'utilisateur et dans les bases de données.

Dans cette section, nous présentons l'adéquation des approches orientées évènements pour la conception de SI ubiquitaires. Ensuite, nous focalisons sur les caractéristiques de ces approches : les modèles d'évènements, les concepts de base et les architectures qui les supportent.

2.1.1 Lien entre les systèmes ubiquitaires et les approches basées évènements

Les systèmes ubiquitaires doivent supporter les composants mobiles ce qui permet d'aboutir à une topologie réseau dynamique. Pour de telles topologies, il n'est pas possible de conserver des structures statiques avec des serveurs centralisés et des modes de communication synchrones. Il est donc pertinent, pour les systèmes mobiles, d'adopter une approche orientée évènements.

Des plus, les systèmes ubiquitaires sont en étroite connexion avec leur environnement. Ils doivent interagir avec des capteurs et des dispositifs dans une structure dynamique et évolutive. La communication par des flux d'évènements entre les capteurs et le système supporte l'adoption d'une approche orientée évènements. De plus cette approche permet de produire une vue de haut niveau des données de capteurs.

Une des caractéristiques requises des applications ubiquitaires concerne la sensibilité et la réactivité au contexte. Or, ceci n'est possible qu'avec une approche permettant la détection des évènements

survenant dans l'environnement d'exécution afin de construire une vue du contexte et de réagir à ces évènements de façon efficace et en temps réel. Tous les avantages précités de cette approche font d'elle un choix de référence pour les systèmes ubiquitaires.

Mises à part les caractéristiques mêmes des SI ubiquitaires, de nombreux domaines d'application existent, où la gestion des évènements est d'une importance cruciale tels que les marchés financiers, le commerce, la sécurité et les services mobiles. Dans ces domaines, il est important de répondre en temps réel (ou dans un temps très peu différé) aux évènements pour gérer des situations exceptionnelles ou indiquer des opportunités ou des risques.

Aujourd'hui, un système ubiquitaire peut adopter une approche orientée évènements dans plusieurs couches informatiques. Ces couches représentent en réalité des domaines informatiques différents où les techniques basées évènements ont évolué indépendamment pour soulever les mêmes défis, à savoir la scalabilité, la dynamicité, l'évolution et la réactivité. Ces domaines sont présentés ci-dessous.

Les bases de données actives Historiquement, elles représentent l'un des premiers usages des évènements où un évènement sert à capturer un changement d'intérêt pour la base de données. Ces événements représentent des déclencheurs de la BD (trigger) tels que des actions de l'utilisateur ou des évènements externes. Des règles ECA sont ensuite utilisées pour déduire des modifications à apporter à la base de données.

Le génie logiciel Dans une architecture logicielle, les évènements présentent un moyen de communication entre les composants logiciels. [Luckham, 2002] présente la relation étroite entre les applications d'entreprise et le traitement des évènements. Ce concept est utilisé, aussi, dans les interfaces graphiques qui utilisent des paradigmes à base d'évènements comme le patron de conception “Observer" ou l'approche Model View Controller (MVC). Par conséquent, les évènements font partie de tous les modèles récents de composants et de frameworks.

La gestion des PM (BPM) Les systèmes de workflow sont souvent construits autour de bases de données centralisées, mais les environnements distribués d'exécution ont montré l'importance de l'usage des évènements. Ces derniers sont échangés entre composants et rapportent les états des activités distribuées. Ils sont utilisés pour des buts comme l'automatisation de la gestion des exceptions et la réactivité aux changements internes ou externes. En effet, la structure procédurale des PM ne peut pas être généralisée dans le cas des systèmes ubiquitaires où il n'est pas possible de prévoir un enchaînement fixe des activités. Par conséquent, un nouveau paradigme “Event driven Business Process Management (EdBPM)" est apparu pour désigner la branche de recherche qui étudie les techniques et les bases conceptuelles pour promouvoir l'usage des évènements dans les PM.

La branche Business Activity Monitoring (BAM) pour le pilotage et l'amélioration des PM connaît, à son tour, un intérêt accru pour les approches à base d'évènements. En fait, les évènements permettent de capturer les états des activités d'un processus en vue d'améliorer leur fonctionnement. Selon [Ammon *et al.*, 2007], un travail collaboratif entre le modélisateur des PM et le modélisateur des évènements doit permettre l'identification des vues BAM et les évènements utilisés par chaque vue. Si nécessaire, ces évènements doivent être agrégés et corrélés pour obtenir des évènements métier de haut niveau.

La gestion de flux de données Cette gestion est réalisée par des systèmes permettant de traiter des flux de données et de répondre à des requêtes continues sur ces flux comme Borealis [5], Aurora [4] ou STREAM [18]. Les sources de données, quelle que soit leur nature (capteurs, services ou systèmes), produisent des évènements de façon continue qui sont évalués avec des requêtes continues afin de

délivrer le résultat au consommateur. Les approches orientées évènements permettent de garantir le découplage entre les producteurs et les consommateurs de données.

Les réseaux de capteurs Les réseaux de capteurs orientés évènements sont basés sur des échanges de flux d'évènements pour le paramétrage des capteurs et la livraison de leurs données. Les capteurs produisent des évènements qui doivent être nettoyés, filtrés et traités. Les techniques de traitement des évènements permettent d'appliquer un pré-filtrage à ces évènements, ce qui permet une sélection des évènements à transmettre. Ainsi, il est possible de réduire la surcharge du réseau et d'optimiser les bandes passantes [Muhl *et al.*, 2006] de façon préventive.

Les approches orientées services Les évènements sont utilisés pour rendre compte de certains changements qui peuvent influencer la structure à base de services d'un système. Ainsi, la dynamicité de l'environnement est répercutée sur la composition et la coordination des services qui devient dynamique à son tour. Une approche supportant cette vision est décrite dans [Kong *et al.*, 2009]. Elle utilise des règles de la forme Web Service - Event Condition Action (WS-ECA) qui permettent dans la partie action d'invoquer des services ou de produire des évènements. L'approche [Levina et Stantchev, 2009] propose une architecture combinant Event Driven Architecture (EDA) et Service Oriented Architecture (SOA) pour permettre le traitement des évènements par des services distribués.

L'usage des approches orientées évènements peut concerner toutes les disciplines informatiques citées ci-dessus. Un système ubiquitaire est généralement concerné par des technologies ubiquitaires touchant toutes ces disciplines. Ce qui fait de l'approche orientée évènements une base idéale supportant les besoins ubiquitaires dans toutes les perspectives du système. Le cas d'un système ubiquitaire supportant une approche par évènements dans plusieurs couches du système a été présenté dans [Zang *et al.*, 2008], où une architecture permet de relier plusieurs composants appartenant à des couches différentes du système à un système de traitement des évènements centralisé.

2.1.2 Gestion des évènements

D'après [Paschke et Vincent, 2009], le cycle de vie d'un évènement passe par les phases suivantes : production, définition, sélection, agrégation, manipulation et consommation. Un schéma explicatif de ce cycle de vie proposé par [Paschke et Vincent, 2009] est représenté par la figure 2.1. Ce cycle de vie rentre dans le cadre d'une architecture de référence utilisée dans un système de traitement des évènements proposé par [Paschke et Vincent, 2009]. Ce cycle de vie permet d'identifier deux grands groupes de phases entre la production et la consommation de l'évènement : la modélisation (représentation) et le traitement des évènements.

Représentation des évènements

Le manque de standard supportant les approches orientées évènements explique les représentations variées des évènements d'un système à un autre. Par exemple, dans [Nagargadde *et al.*, 2005], un évènement est représenté par trois dimensions : le temps, l'espace et le label. Il est convenu dans tous les travaux que le temps est un paramètre caractéristique et essentiel pour tout évènement. Il peut être un point dans le temps, on parle d'évènement instantané, ou un intervalle pour représenter les évènements qui s'étalent sur une durée de temps. L'espace représente la localisation d'un évènement (de même il est possible d'utiliser des régions pour indiquer l'espace) et permet de déduire des relations entre évènements. Le label représente le type de l'évènement. D'après [Nagargadde *et al.*, 2005],

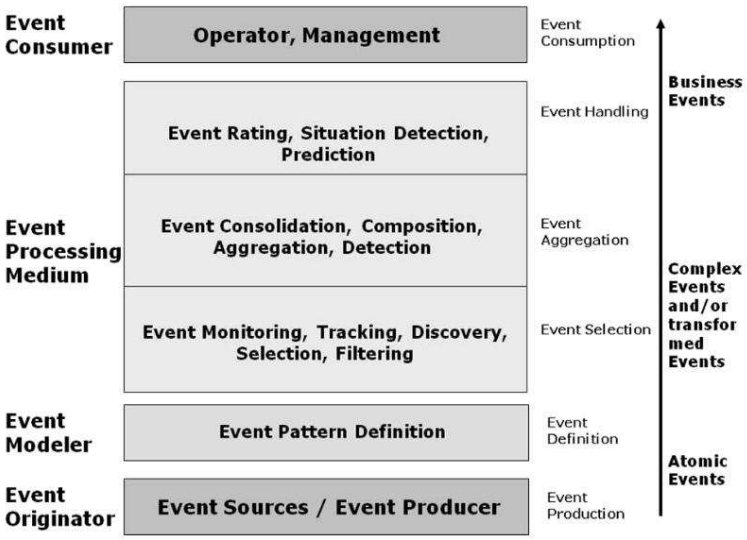


FIGURE 2.1 – Cycle de vie d'un évènement et architecture de référence [Paschke et Vincent, 2009]

chaque domaine est caractérisé par une ensemble de labels, par exemple les types d'évènements *Retard, Avance, Accident* et *Travaux* caractérisent le domaine des transports publics.

Selon [Luckham, 2002], un évènement est caractérisé par trois aspects : sa forme, sa signification et sa relativité. La forme d'un évènement est un objet avec des attributs particuliers qui représentent des données de l'évènement parmi lesquelles son timestamp. Dans certains travaux la forme d'un évènement est dite schéma ou type d'évènements. La signification d'un évènement est une donnée abstraite qui représente l'activité à l'origine de l'évènement (un évènement est l'enregistrement d'une activité selon [Luckham, 2002]). La relativité indique les relations entre les évènements qui peuvent être le temps, la causalité et l'agrégation. Une relation temporelle entre évènements représente l'ordonnancement de ces évènements l'un par rapport à l'autre. La causalité est une relation de dépendance entre évènements, où un évènement A cause un évènement B. L'agrégation est une relation d'abstraction pour désigner un évènement qui est composé d'autres évènements.

Chaque système propose sa propre vision des évènements et ses propres modèles. Ces modèles servent essentiellement à typer les évènements, associer des attributs aux évènements et définir les liens possibles entre évènements. Un exemple typique de ces modèles est représenté par la figure 2.2. Il s'agit d'un métamodèle d'évènements structurant des objets évènements utilisés dans le système de gestion des évènements SARI [Rozsnyai *et al.*, 2007]. Ce métamodèle résume la représentation des évènements dans le système SARI, et il est comparé, dans [Rozsnyai *et al.*, 2007], à d'autres modèles d'évènements utilisés dans les systèmes Esper [11], Hermes [Pietzuch et Bacon, 2002], Borealis [5], RuleCore [16] et AMIT [Adi et Etzion, 2004]. Cette comparaison met en valeur les différences entre les représentations des évènements et les techniques (langages) d'expression des objets évènements dans chaque système.

Le métamodèle d'évènements de la figure 2.3, proposé par [Zang *et al.*, 2008], représente non seulement une vision des évènements, mais aussi leur place dans un SI d'entreprise. Il s'agit d'un métamodèle brut sans précision de toutes les multiplicités. En effet, dans [Zang *et al.*, 2008], la gestion des évènements est centralisée pour tous les besoins du SI (bases de données, services et workflows). Dans ce métamodèle, un évènement peut être primitif ou composite, possède un ensemble de propriétés (attributs) et il est dérivé des bases de données, des services, des activités d'un processus ou des capteurs Radio Frequency IDentification (RFID). La notion de contexte dans ce métamodèle

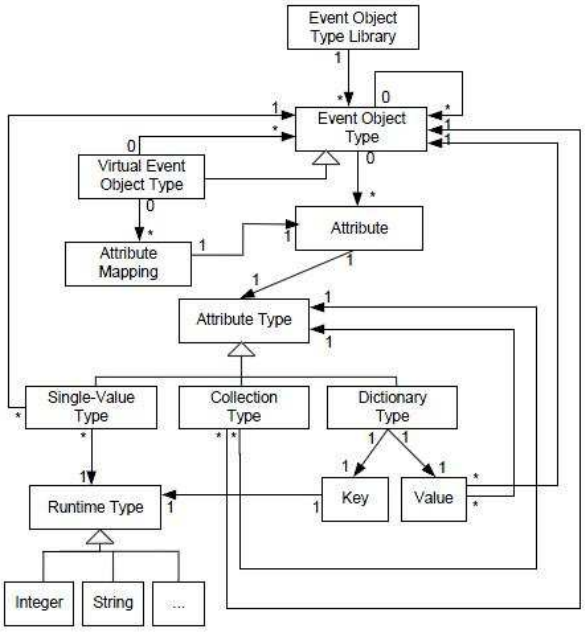


FIGURE 2.2 – Métamodèle pour la représentation des évènements du système SARI [Rozsnyai *et al.*, 2007]

fait référence à toute information utile pour abstraire les évènements de bas niveau et obtenir des évènements de haut niveau telle que la hiérarchie d'abstraction de certaines dimensions comme les produits ou le temps.

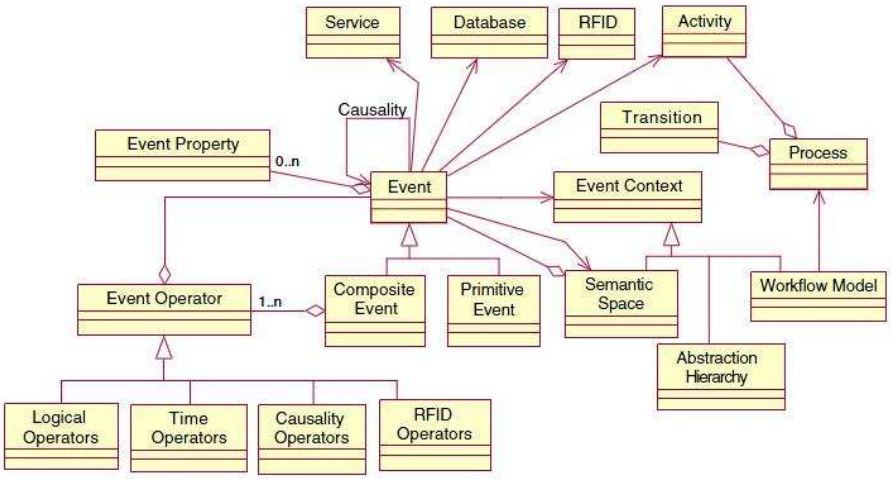


FIGURE 2.3 – Métamodèle brut d'évènements [Zang *et al.*, 2008]

Le manque de standards supportant les approches orientées évènements explique la multitude des représentations des évènements. Ceci est néanmoins un point positif puisque ces approches ne possèdent pas de modèles rigides et peuvent s'appliquer à des domaines informatiques variés.

Traitements des évènements complexes (Complex Event processing (CEP))

Le traitement des évènements complexes est un nouveau paradigme qui a été utilisé dans [Luckham, 2002] pour désigner l'ensemble des techniques utilisées pour produire des évènements complexes. Un évènement complexe est une agrégation (composition) d'autres événements primitifs, temporels ou complexes.

Proche de la notion d'évènement complexe, la notion de patron d'évènements est un modèle combinant un ensemble d'évènements. Il décrit, précisément, non seulement les évènements mais aussi des dépendances causales, des contraintes temporelles, des paramètres de données et du contexte [Luckham, 2002]. Un patron d'évènements utilise des opérateurs logiques et temporels pour relier les évènements, tout en associant des conditions sur les attributs de ces évènements. L'application des patrons d'évènements permet de produire des évènements complexes. Un patron est satisfait si les évènements se produisent dans l'ordre prédéfini et si leurs attributs vérifient les conditions du patron ce qui permet de détecter des situations particulières et intéressantes. Ces patrons sont spécifiés avec des langages Event Processing Language (EPL) basés sur des algèbres d'évènements et qui sont exécutés par des moteurs d'évènements. Deux catégories différentes de langages EPL peuvent être identifiées : les langages à base de règles ECA qui sont utilisés dans RuleCore [16] et AMIT [Adi et Etzion, 2004] et des langages proches de SQL comme ceux utilisés dans Esper [11] et Aleri [2]. Dans la figure 2.4, un exemple de patron d'évènements est exprimé en Continuous Computation Language (CCL), le langage utilisé par Aleri [2]. Dans le cadre d'une application de transport, ce patron permet de produire une alerte de retard qui est le résultat de deux évènements de retard successifs dont la valeur de retard dépasse 3 mn.

```
INSERT INTO
retardAlert
SELECT
retardf a, retardf b
MATCHING
[ a , b]
ON
a.vehicle = b.vehicle
WHERE
a.time > 3 AND b.time > a.time
```

FIGURE 2.4 – Exemple de règle exprimée avec EPL

Les règles de traitement réagissent à certaines situations pour détecter un patron d'évènements et générer d'autres évènements de plus haut niveau envoyés vers d'autres composants du système. Les règles peuvent contenir d'autres propriétés non fonctionnelles comme le mode de couplage, le mode de consommation et les priorités [Chakravarthy et Jiang, 2009]. Ces règles utilisent les techniques suivantes [Ben Cheikh *et al.*, 2010c] :

- **la composition (ou l'agrégation) :** elle consiste à dériver un évènement en utilisant un ou plusieurs évènements dans le cadre d'une dimension temporelle. La sémantique des dimensions temporelles utilise souvent les opérateurs temporels définis par [Allen, 1991]. Les opérations de composition sont la conjonction (AND) et la disjonction (OU).

- **la séquence :** elle consiste à déduire un évènement suite à la succession d'autres évènements. D'après la définition de la séquence, un évènement e_1 avec un temps d'occurrence t_1 succède à un évènement e_2 avec un temps d'occurrence t_2 si $t_1 > t_2$.
- **le filtrage :** il est utilisé pour vérifier la satisfaction d'une condition par un ensemble d'évènements. Si la condition est vérifiée par un évènement alors il est généré en sortie. Ce type de règle est un laissez-passer accordé aux évènements. Les règles de filtrage peuvent être utilisées pour personnaliser la livraison des évènements en utilisant les paramètres du profil comme conditions contextuelles. Par exemple, un évènement de perturbation n'est envoyé au voyageur que si dans ses itinéraires, ce voyageur utilise des moyens de transport concernés par la perturbation.
- **la corrélation :** elle a suscité beaucoup d'intérêt puisqu'elle permet de mieux gérer de grands ensembles d'évènements en identifiant les évènements concernés par une même situation. Elle permet de chercher dans un nuage d'évènements les évènements reliés à un évènement d'entrée. Elle compare les attributs des évènements à ceux de l'entrée pour identifier les évènements qui partagent les mêmes valeurs d'attributs. La figure 2.5 représente un exemple de corrélation entre deux évènements. La situation où deux trains utilisant la même voie et qui sont proches (attribut localisation) dont l'un est en retard et l'autre en avance représente un risque d'accident. Pour comparer les attributs, un espace sémantique et des opérateurs de comparaison doivent être utilisés pour identifier les attributs équivalents. [Aasman, 2008] propose une ontologie d'évènements avec un langage de requêtes adapté pour raisonner sur des attributs temporels, spatiaux et sociaux.
- **la causalité :** elle permet de déduire à partir d'une situation les évènements qui en découlent. Plusieurs sortes de causalité peuvent être identifiées :
  - *transformation des évènements :* par modification des évènements d'entrée. Les transformations sont appliquées aux attributs des évènements en les enrichissant ou en les réduisant.
  - *déduction des évènements :* par utilisation des relations de causalité. Les évènements générés et les évènements d'entrée sont de types différents, ils sont simplement reliés de cause à conséquence par certaines conditions contextuelles. Un évènement signalant des conditions de trafic difficiles permet de déduire des retards sur les lignes de bus.
  - *prédiction d'évènements :* c'est un cas particulier de la déduction d'évènements puisque les évènements déduits sont des évènements virtuels qui ne se sont pas encore produits. Les données météo sont des évènements virtuels à partir desquels plusieurs règles sont utilisées pour prévoir des évènements futurs.

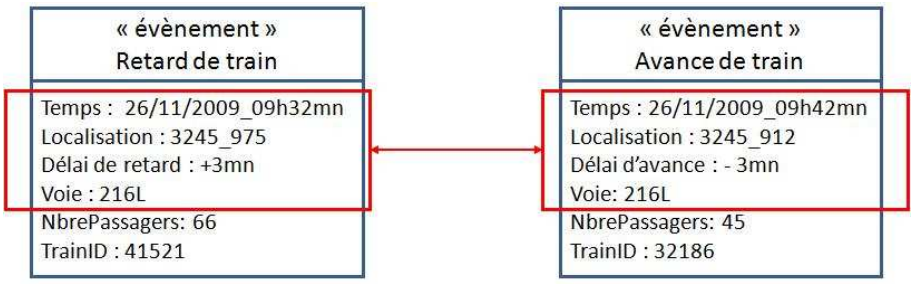


FIGURE 2.5 – Exemple de relation de corrélation entre deux évènements

Toutes les techniques citées ci-dessus peuvent être regroupées ensemble pour former une chaîne globale de traitement des évènements formant des règles de traitement des évènements. Les unités de traitement des évènements prennent en charge l'exécution de ces règles et la livraison des évènements produits aux ressources qui leur sont liées selon une architecture dirigée par les évènements. Les

topologies et les structures de ces architectures sont étudiées dans la section suivante.

2.1.3 Architecture dirigée par les évènements

L'architecture dirigée par les évènements (EDA) est le cadre exécutable englobant toute approche orientée évènements. Le principe de cette architecture est que chaque évènement interne ou externe, quand il se produit, est envoyé immédiatement à toutes les parties intéressées (humaines ou automatisées) [Michelson, 2006]. Par nature, une architecture EDA est faiblement couplée et très distribuée. Le producteur d'un évènement ne connait pas les parties intéressées ni les traitements que va subir l'évènement. Ainsi, cette architecture est le choix idéal pour des flux d'informations et des tâches asynchrones. Mais, cette architecture rencontre encore certains verrous reliés principalement à l'usage d'un standard permettant l'interopérabilité entre les composants hétérogènes.

Infrastructure des architectures

Une architecture dirigée par les évènements est basée sur un ensemble de composants qui constitue ce qu'on appelle une infrastructure CEP. Cette infrastructure est responsable de l'interfaçage avec les couches techniques du système et fournit une plateforme pour l'implémentation du CEP. Un exemple d'architecture fournie par [Michlmayr *et al.*, 2008] est présenté dans la figure 2.6. Les composants de cette architecture sont décrits ci-dessous.

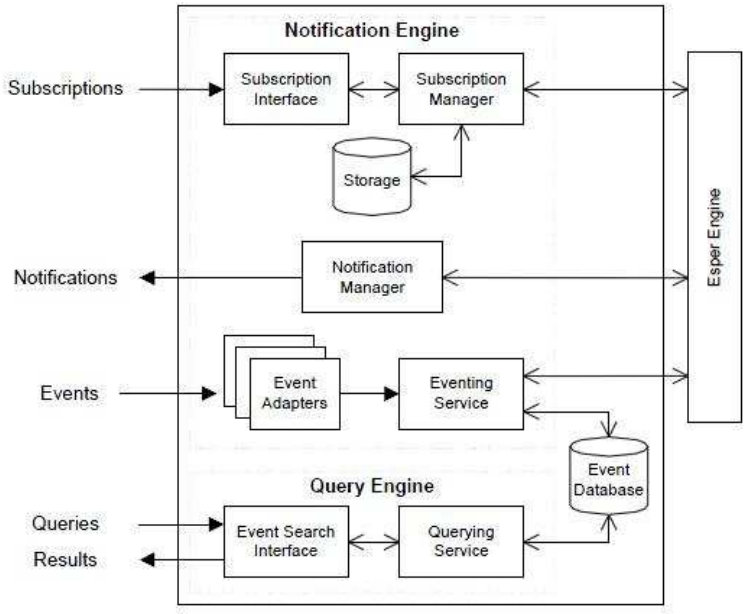


FIGURE 2.6 – Exemple d'architecture orientée évènements [Michlmayr *et al.*, 2008]

Adaptateur Un adaptateur permet de gérer diverses sources d'évènements et de convertir les évènements reçus en des formats de données compréhensibles par l'infrastructure CEP pour faciliter son interprétation. Ensuite, il délivre l'évènement au composant concerné. Plusieurs types d'adaptateurs existent tels que les adaptateur XML, les adaptateurs de bases de données et les adaptateurs d'alarmes.

Service de notification Un service de notification, dans une architecture EDA, est basé sur l'usage des communications Publish/Subscribe. Ce mécanisme permet de découpler les producteurs et les consommateurs des évènements des messages. Des produits standards appliquant ce mécanisme existent tels que CORBA Notification Service ou Java Message Service (JMS).

Moteur de règles Un moteur de règles ou un moteur d'évènements est responsable de l'exécution des règles de traitement des évènements. Une règle prend en entrée un patron d'évènements et produit un ensemble d'évènements en sortie en appliquant des opérations de traitement (transformation, corrélation, composition, etc.). Plusieurs moteurs d'évènements existent dans des contextes industriels ou académiques comme Esper [11], Cayuga [7], Hermes [Pietzuch, 2004], IBM WebSphere Business Event [12] et TIBCO Rendezvous [15].

Dépôt de métadonnées Le dépôt permet de sauvegarder les concepts et les données du système tels que les ontologies d'évènements, la configuration des adaptateurs, la configuration des services et les données des règles. Ce dépôt est utilisé pour la configuration et la maintenance [Cilia *et al.*, 2001].

Réseau de traitement des évènements : Event Processing Network (EPN)

Un réseau de traitement des évènements (EPN) est un framework conceptuel constitué de quatre composants : des producteurs d'évènements, des consommateurs d'évènements, des agents de traitement des évènements (EPA) et des composants de connexion appelés canaux d'évènements (Event channel) [Sharon et Etzion, 2008]. Un EPN permet de décrire les échanges d'évènements entre producteurs et consommateurs d'évènements à travers des agents qui traitent ces évènements en appliquant des opérations comme la transformation, l'agrégation et la corrélation. Un aperçu de ces traitements d'évènements est donné dans [Lakshmanan *et al.*, 2009] qui propose une classification des EPA en fonction des opérations qu'ils effectuent (voir figure 2.7). Selon [Sharon et Etzion, 2008], ces agents procèdent aux trois tâches suivantes :

- détection de patrons : pour sélectionner les évènements à traiter,
- traitement : en appliquant des opérations sur ces évènements,
- émission : pour décider comment et où envoyer les évènements dérivés.

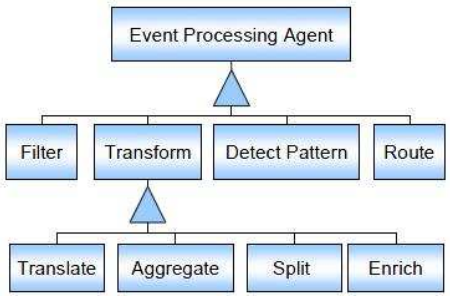


FIGURE 2.7 – Classification des agents EPA [Lakshmanan *et al.*, 2009]

Les réseaux EPN permettent une réutilisation des agents permettant ainsi une modification et une évolution plus facile des systèmes. De plus ces réseaux peuvent être décomposés dynamiquement en assemblant plusieurs EPN. Cette approche permet aussi de faciliter la gestion des évènements complexes en utilisant des étapes successives simples au lieu de procéder avec des patrons complexes.

Une approche de stratification des réseaux EPN est proposée par [Lakshmanan *et al.*, 2009], où il est possible d'organiser des agents, partageant des caractéristiques communes, en des strates comme

le montre la figure 2.8. Cette approche a l'avantage de permettre à des agents de travailler en parallèle et d'organiser des systèmes distribués complexes optimisant ainsi le temps d'exécution et améliorant la scalabilité de l'application.

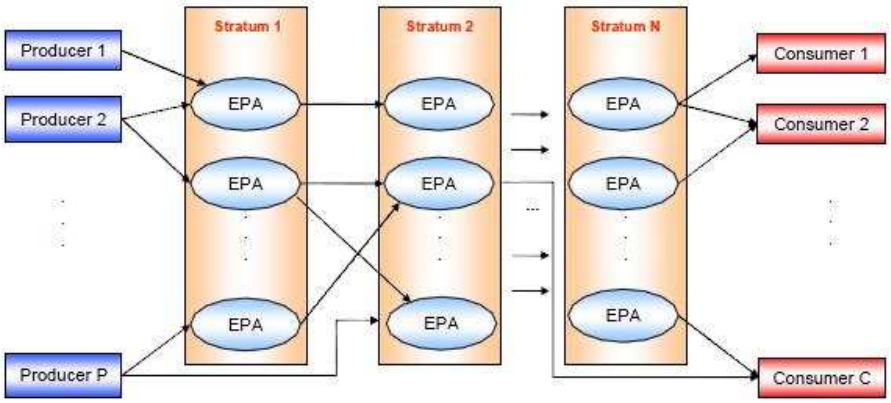


FIGURE 2.8 – Réseau EPN stratifié [Lakshmanan *et al.*, 2009]

2.1.4 Synthèse

Les approches orientées évènements bénéficient d'un intérêt accordé par plusieurs domaines de recherche en vue d'intégrer les évènements dans différents composants d'un SI. Ceci est dû à un besoin croissant de communication asynchrone, scalable et fortement distribuée. La communication via des flux d'évènements garantit un faible couplage entre les composants distribués et une flexibilité pour supporter la dynamicité de leur topologie. Cette section présente un aperçu sur ces approches en décrivant les usages possibles des systèmes à base d'évènements, les modèles utilisés pour les évènements (concept central), les approches de traitement des évènements et les architectures ainsi que l'infrastructure d'un système à base d'évènements. La littérature est très riche en langages et outils portant sur ces approches. Cependant, d'une part ce domaine manque de modèles et métamodèles standards, et d'autre part, l'ingénierie de ces systèmes n'a pas bénéficié du même intérêt. Ce dernier point fait l'objet de la suite de cette section.

Dans ce qui suit, nous nous intéressons à deux types de démarches d'ingénierie. Notre étude des démarches de développement de SI traditionnels a pour but d'identifier les capacités de ces démarches à supporter les besoins ubiquitaires. L'existence d'approches de développement spécifiques aux SI ubiquitaires est étudiée pour comparer ces approches et dégager les besoins ubiquitaires supportés par chacune.

2.2 Démarches unifiées pour l'ingénierie des SI

La complexité croissante des SI a suscité un intérêt croissant pour les méthodes de développement. Les processus unifiés font partie des méthodes les plus récentes. Aujourd'hui, ils représentent un référentiel pour les projets de développement de SI et de logiciels et une trame commune des meilleures pratiques de développement. Ces démarches sont basées sur un cycle de vie itératif, incrémental et piloté par des cas d'utilisation. Ces démarches sont génériques, c'est-à-dire applicables à tout type d'entreprise et tout domaine d'application. Plusieurs démarches unifiées ont vu le jour et sont répandues et largement utilisées. L'objectif de cette section est de décrire trois de ces démarches et de discuter de leur aptitude à supporter les critères ubiquitaires.

2.2.1 Rational Unified Process RUP

Le RUP est un processus commercial, développé et maintenu par Rational Software (maintenant dans le cadre d'IBM). Le processus RUP représente l'une des plus célèbres implémentations de l'approche Processus Unifié. D'après [Kruchten, 2003], il est utilisé par une dizaine de milliers d'entreprises.

UML est le langage graphique utilisé par le RUP pour exprimer les différents modèles. La richesse d'UML en terme de diagrammes influe sur le RUP qui permet d'exprimer différentes vues :

- une vue logique pour représenter des données avec des diagrammes de classes,
- une vue dynamique pour représenter des processus avec des diagrammes d'états-transitions, des diagrammes d'activités et des diagrammes de séquences,
- une vue utilisateur qui permet de guider l'analyse des besoins avec des diagrammes de cas d'utilisation,
- une vue composants qui permet d'exprimer des modules à déployer avec le diagramme de composants,
- une vue déploiement qui permet de projeter les composants sur le matériel avec le diagramme de déploiement.

Le RUP permet de présenter le cycle de développement d'un processus en fonction de phases entreprises par un ensemble d'acteurs (analystes, développeurs, testeurs, managers) et qui évoluent au cours des différentes itérations. La figure 2.9 représente l'architecture générale du RUP caractérisée par deux dimensions :

- l'axe horizontal qui représente le temps et montre le déroulement du cycle de vie du processus ; cette première dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases d'itérations et de jalons.
- l'axe vertical qui représente les principaux enchaînements d'activités qui sont regroupées selon leur nature dans des phases. Cette seconde dimension rend compte de l'aspect statique du processus qui s'exprime en terme de composants, de processus, d'activités, d'enchaînements, d'artefacts et de contributeurs. Ces phases permettent le développement de toutes les couches logiques du système.

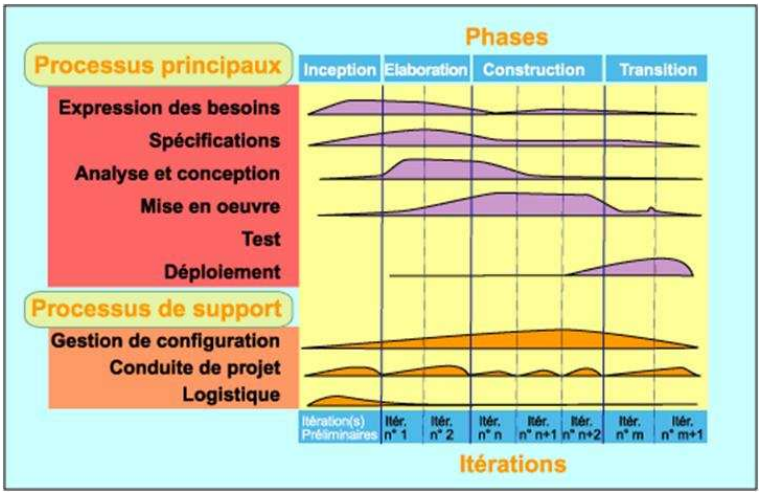


FIGURE 2.9 – Représentation du processus de développement du RUP [Kruchten, 2003]

2.2.2 2TUP : 2 Track Unified Process

Comme son nom l'indique, le processus 2TUP [Rocques et Vallée, 2002] comporte deux branches de développement qui évoluent en parallèle : un développement fonctionnel et un développement technique pour répondre aux contraintes techniques et fonctionnelles qui sont imposées par le projet. A l'issue de l'évolution de ces deux branches, un axe central permet la fusion de leurs résultats pour établir la conception et la réalisation du système. Ceci conduit à un processus en « Y », comme le montre la figure 2.10.

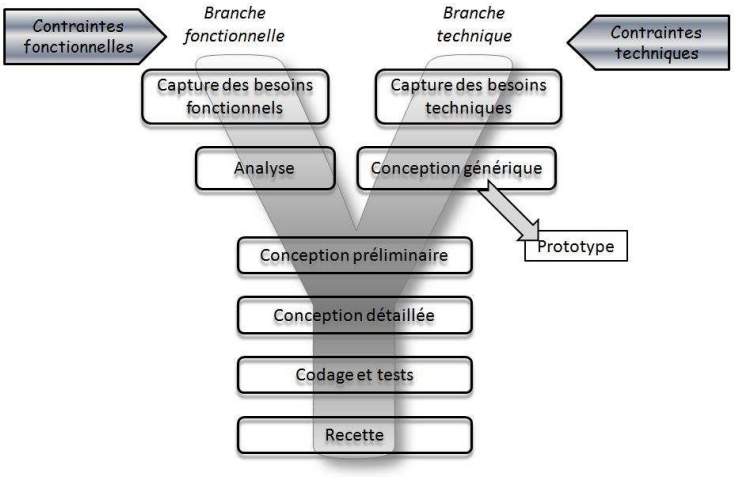


FIGURE 2.10 – Processus de développement en Y du 2TUP [Rocques et Vallée, 2002]

Ce processus est organisé selon trois branches :

- **la branche fonctionnelle** permet la capture des besoins fonctionnels et produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs. De son côté, la maîtrise d'œuvre consolide les spécifications et en vérifie la cohérence et l'exhaustivité. L'analyse consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en terme de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.
- **la branche technique** permet dans un premier temps de capturer les besoins techniques, qui recensent toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des prérequis d'architecture technique. Dans un second temps, la conception générique permet de définir les composants nécessaires à la construction de l'architecture technique. Cette conception est la moins dépendante possible des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour un même système. La réussite de cette branche est vérifiée par la réalisation d'un prototype pour tester sa validité.
- **la branche centrale** comporte les phases suivantes :
  - *la conception préliminaire* est une étape délicate qui intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer,
  - *la conception détaillée* étudie ensuite comment réaliser chaque composant,
  - *l'étape de codage* produit ces composants et teste en parallèle les unités de code réalisées,

– *l'étape de recette* permet la validation des fonctions du système développé.

L'usage d'un processus en Y permet de produire des modèles réutilisables puisque les modèles fonctionnels et techniques sont indépendants. Il est ainsi plus facile d'appliquer des changements sur une branche sans affecter les résultats de l'autre. UML représente le langage de base pour tous les modèles du 2TUP.

2.2.3 Symphony : une méthode d'ingénierie des SI orientée objets métier

Symphony est une méthode de développement centrée sur les objets métier, originellement développée au sein de la société Umanis [20] en collaboration avec l'équipe SIGMA. Des projets ambitieux (refonte du système d'information du service après vente de l'enseigne Conforama [Hassine, 2005] et migration d'une partie de l'intranet du système d'information hospitalier du CHU de Grenoble [Jausseran, 2005]) ont notablement permis de mettre en application les évolutions théoriques de la méthode. L'extension de Symphony pour le développement des systèmes de réalité mixte [Godet-Bar, 2009] a permis une considération plus approfondie des aspects interactionnels et de conception des IHM.

Symphony s'est imposée devant d'autres méthodes telles que Catalysis [D'Souza et Wills, 1998], RUP [Kruchten, 2003] , Select Perspective [SelectPerspective, 2006] et 2TUP [Rocques et Vallée, 2002] dans une comparaison décrite dans [Hassine *et al.*, 2005], grâce aux caractéristiques suivantes :

– elle intègre une approche fonctionnelle centrée sur les objets métier, dans une perspective de réutilisation et de réutilisabilité,
– les objets métier sont identifiés de manière systématique lors de la spécification des besoins,
– la traçabilité des choix est garantie tout au long du processus de développement,
– elle intègre un modèle d'objets métier tripartite original : les Objets Métier.

Le cycle de développement de Symphony est caractérisé par un processus de développement « en Y », dont les principes et l'organisation empruntent aux méthodes Merise 2 et RUP.

La figure 2.11 illustre le cycle de vie en Y tel qu'il a été décrit à l'issue des travaux de Godet-Bar [Godet-Bar, 2009].

Symphony sépare l'étude des besoins fonctionnels de celle des besoins techniques et ce dès le début du cycle de développement. Cette approche permet non seulement une meilleure analyse des problèmes et des risques, mais aussi une meilleure réutilisation de l'existant. Menées en parallèle, ces deux activités se déroulent néanmoins en étroite collaboration afin d'en assurer la cohérence. Le cycle de vie de la méthode Symphony est ainsi organisé comme suit :

– **l'étude préalable**, dont les objectifs sont de réaliser une modélisation des processus métier existant sous forme de cas d'utilisation et d'identifier les acteurs impliqués dans l'utilisation du produit final.
– **la branche fonctionnelle (gauche)** qui correspond à la traditionnelle analyse du métier, ainsi qu'à la modélisation des besoins utilisateurs, indépendamment des caractères techniques de l'application. La version étendue de Symphony proposée par [Godet-Bar, 2009] intègre dans cette branche l'analyse des besoins interactionnels et l'élaboration de modèles interactionnels. Cette branche est caractérisée par une collaboration étroite entre le spécialiste Génie Logiciel (GL) et le spécialiste IHM.
– **la branche technique (droite)** qui traite des aspects architecturaux de l'application, aussi bien en terme logiciels que matériels. Cette branche prend en compte également les contraintes non fonctionnelles telles que la sécurité, l'équilibrage de charge, la distribution ...
– **la branche centrale** qui intègre les modèles d'analyse de la branche gauche et l'architecture applicative définie dans la branche droite dans un modèle de conception. Ce dernier permet de

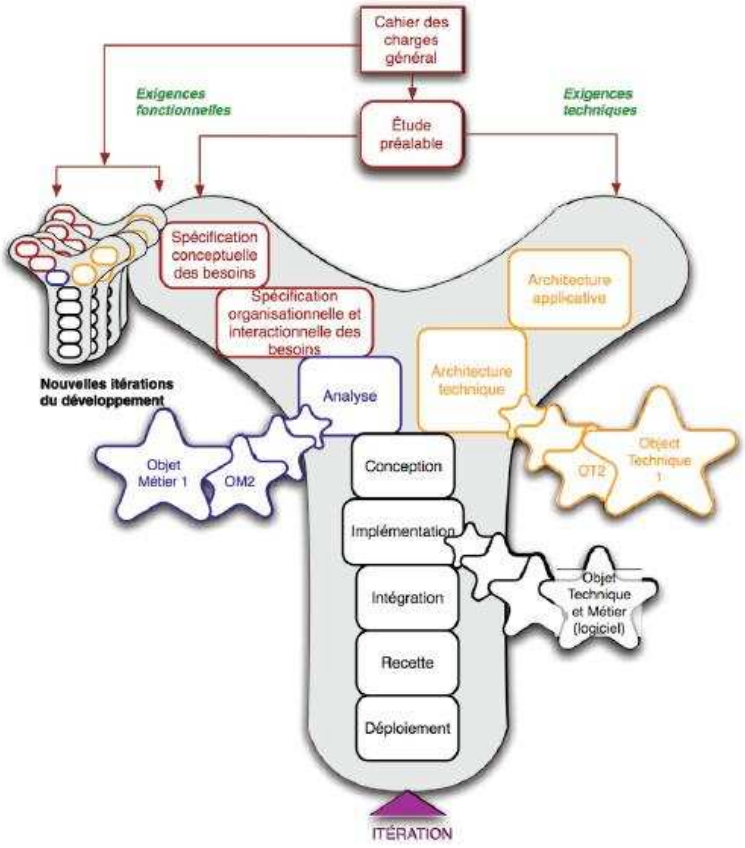


FIGURE 2.11 – Nouveau cycle de développement Symphony [Godet-Bar, 2009]

raffiner les composants jusqu'au niveau du code d'implémentation.

D'un point de vue général, Symphony utilise exclusivement le langage UML pour réaliser les différents artefacts du développement. Ces artefacts sont conceptuellement présentés sous forme de paquetages tripartites nommés les Objets Métier, et le système est vu comme un assemblage d'Objets Métier (représentant des processus, des entités ou des données de référence).

2.2.4 Synthèse : prise en compte des besoins ubiquitaires dans les démarches unifiées

Les démarches présentées dans cette section fournissent un cadre solide et bien fondé suscitant la confiance des entreprises. Elles sont orientées composants ce qui permet de produire un modèle d'implantation sous forme d'un ensemble de composants. L'avantage de ces approches est d'obtenir un système avec des parties indépendantes les unes des autres ce qui facilite les changements et la réutilisation. Ceci présente un avantage en faveur des systèmes ubiquitaires à faible couplage et fortement distribués. Cependant, les démarches décrites dans cette section sont génériques et dédiées à tout type de projet ou de domaine d'application, ce qui les rend moins appropriées à un système spécifique, en particulier à un système ubiquitaire. En focalisant sur les différents besoins ubiquitaires : mobilité, distribution, sécurité et sensibilité au contexte, les démarches unifiées présentent des lacunes pour prendre en charge ces besoins dans les différentes phases de développement. Ces lacunes sont présentées ci-dessous.

En effet, dans ces démarches, les phases de spécification des besoins font la différence entre un

besoin fonctionnel et un besoin non fonctionnel qui ne sont pas explicitement reliés à des besoins d'adaptation, de réutilisation et de flexibilité associés aux besoins de sensibilité au contexte. Concernant le 2TUP, la notion de contexte utilisée dans la phase d' « étude préliminaire »est différente de la notion de contexte utilisée dans les systèmes ubiquitaires car elle représente l'ensemble des acteurs interagissant avec le système et les messages échangés entre eux.

La conception métier, réduite dans RUP à une phase et dans 2TUP et Symphony à la branche fonctionnelle, ne prend en compte ni les besoins d'adaptation des PM ni les mesures d'injection de flexibilité dans les modèles produits. Il en est de même pour l'analyse des interactions et la conception des interfaces qui sont figées et ne permettent pas de produire des interfaces flexibles.

L'accès aux données et les techniques de contrôle d'accès, essentiels pour la sécurité de SI ubiquitaires, ne sont pas considérés dans ces démarches, ce qui présente une lacune lors de la conception de systèmes comprenant des fonctionnalités comme la personnalisation de l'information et le contrôle d'accès sensible au contexte.

Les interactions avec l'environnement, fortement distribué et mobile, se basent essentiellement sur l'échange d'évènements, qui ne sont pas prévus dans ces processus de développement, sauf dans Symphony qui utilise les évènements externes pour décomposer les PM. En effet, dans un processus global, les PM qui les composent sont déclenchés par des évènements externes permettant de différencier un processus de l'autre.

Les phases de développement technique permettent de réaliser des architectures, matérielles, logicielles et applicatives et prennent en compte les besoins de communication. Cependant, une architecture à flux d'évènements manque à ces phases pour réaliser un système distribué et mobile. Une telle architecture permet de prévoir les composants de détection d'évènements, de routage de flux et de gestion des évènements.

L'application des démarches unifiées pour le développement de SI ubiquitaires n'est donc pas appropriée. La section suivante présente des approches spécifiques de conception de systèmes ubiquitaires qui représentent des efforts d'identification des spécifications ubiquitaires et des propositions de processus de développement d'une partie ou de la totalité du système.

2.3 Démarches de développement des applications ubiquitaires

Le besoin de concevoir des applications ubiquitaires a engendré l'apparition de modèles et de systèmes dans plusieurs disciplines concernées par des besoins ubiquitaires. Néanmoins, un intérêt moins important a été accordé aux démarches de développement de SI ubiquitaires. Parmi les toutes premières démarches, celle proposée dans [Zambrano *et al.*, 2004] utilise la programmation orientée aspects pour découper les besoins en un ensemble d'aspects et construire un système ubiquitaire modulaire. Dans ce type de système, le découplage est total entre les fonctionnalités et les besoins d'adaptation au contexte. Mais cette approche ne propose pas une démarche globale de conception, mais seulement une manière de concevoir.

D'autres approches se sont intéressées à l'intégration d'un besoin ubiquitaire précis lors de la conception d'une application. Par exemple, [Augustin *et al.*, 2006] fournit un environnement pour la construction des applications ubiquitaires en utilisant une plateforme de développement et un intergiciel pour l'exécution. Cette proposition est technique et ne permet pas de capturer les besoins de conception. Elle part d'un ensemble d'unités de code qui présentent un comportement ubiquitaire. Ensuite des morceaux alternatifs de code doivent être définis avec les situations contextuelles correspondantes. L'environnement proposé se charge de la sélection du code approprié pour chaque situation en temps d'exécution. Cette proposition ne peut ainsi être utilisée que pour de petites applications ubiquitaires avec des besoins d'adaptation très clairs.

Enfin, l'injection de l'adaptation dans des applications ubiquitaires a fait l'objet de la proposition de [Chaari *et al.*, 2007] qui propose un framework prenant en charge une adaptation des services, du contenu et de la présentation. Cette adaptation est réalisée avec un système contenant des modules de gestion et de stockage du contexte et des modules d'adaptation.

Néanmoins, ces approches ne présentent pas de vraies démarches de développement mais plutôt des consignes et des guides pour la conception d'applications ubiquitaires ou de fonctionnalités ubiquitaires à attacher à des systèmes préexistants.

A contrario, nous présentons dans cette section des démarches de développement de systèmes ubiquitaires. Cet ensemble de démarches est considéré, jusqu'à présent, comme le plus complet possible. Nous remarquons l'apparition d'un nombre plus important de travaux traitant ce sujet, durant les deux dernières années. Nous décrivons chaque démarche de façon indépendante, ensuite nous présentons une comparaison pour vérifier la satisfaction des caractéristiques ubiquitaires supportées par ces démarches.

2.3.1 Méthode pour l'ingénierie des applications ubiquitaires [Henricksen et Indulska, 2006]

La méthode d'ingénierie proposée dans [Henricksen et Indulska, 2006] repose sur un ensemble de concepts et de langages pour la modélisation et la gestion du contexte. Elle utilise, entre autres, une architecture d'une infrastructure logicielle générique à adapter selon les projets. La démarche est illustrée par la figure 2.12 qui présente un enchaînement d'activités de développement concernant : l'analyse (A), la conception (D), l'implémentation et la programmation (P), la personnalisation de l'infrastructure (I) et les tests (T). L'analyse permet d'identifier dans un premier temps les exigences et les fonctionnalités de l'application (A1). Ensuite, il est recommandé d'identifier les types d'information contextuelle exigés par ces fonctionnalités en modélisant le contexte avec CML et d'exprimer les situations auxquelles doit répondre le système (A2). Finalement, l'étape (A3) doit raffiner les choix de l'utilisateur et identifier les déclencheurs (évènements) pour les comportements d'adaptation. L'analyse est suivie de deux branches qui s'effectuent en parallèle : l'une concerne la conception et l'implémentation (étape classique de mise en œuvre et codage), et l'autre concerne l'architecture technique.

L'infrastructure de gestion du contexte proposée dans cet article doit être reprise et adaptée au projet pour obtenir une architecture en couches permettant la détection, le traitement et l'usage du contexte (I1). Ensuite l'étape (I2) permet de construire des ensembles de données contextuelles, de préférences et d'évènements pour réaliser les tests (T2). Le succès de ces tests permet de passer à l'étape (I3) pour construire les interfaces de communication avec les utilisateurs et réaliser, de nouveau, des tests (T3).

Cette démarche n'est pas itérative, mais elle permet des allers/retours entre les différentes étapes. Elle assure ainsi une bonne modélisation du contexte et les mécanismes qui supportent sa gestion. Cependant, peu d'intérêt a été accordé aux fonctionnalités du système et leur lien avec le contexte pour guider différents types d'adaptation. De plus, l'identification des informations contextuelles n'est pas guidée et ne garantit pas une définition complète des besoins. Ainsi, cette approche est assez générale et ne permet pas de prévoir toutes les étapes nécessaires pour le développement de ces systèmes.

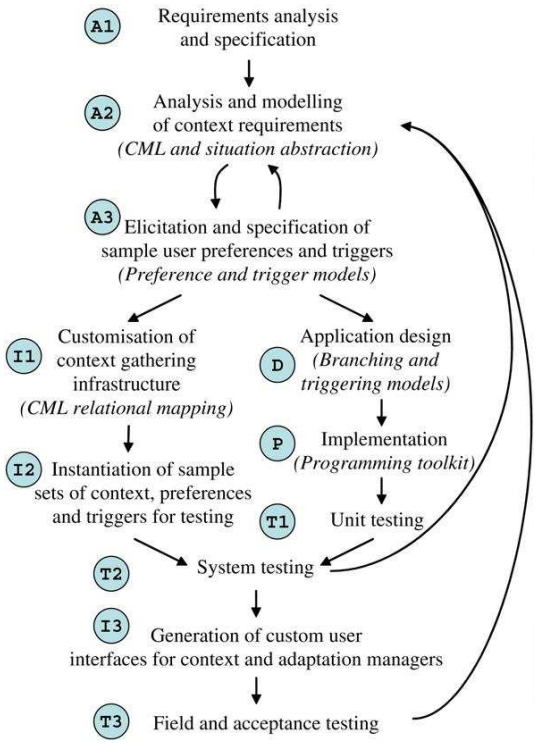


FIGURE 2.12 – Processus de développement d'applications ubiquitaires [Henricksen et Indulska, 2006]

2.3.2 Démarche IDM pour les applications sensibles au contexte [Ayed *et al.*, 2007]

L'article [Ayed *et al.*, 2007] propose une démarche dirigée par les modèles pour le développement des systèmes sensibles au contexte. Cette approche est basée sur l'usage d'un profil UML pour la modélisation du contexte et des comportements adaptables de l'application. L'approche IDM permet d'obtenir un modèle de l'application indépendant de la plateforme et ensuite de le transformer pour l'implémenter avec différentes plateformes. Cette démarche est basée sur un enchaînement de phases présenté par la figure 2.13.

Les deux premières phases permettent de produire le modèle de contexte et les comportements variables de l'application (qui dépendent du contexte) en instanciant des profils UML. L'adaptation supportée par cette approche peut être structurelle (variation de la structure de certaines classes), architecturale (existence d'objets optionnels) ou comportementale (variation des diagrammes dynamiques de l'application). Ces deux phases permettent de fournir des modèles indépendants de la plateforme et sont suivies par deux phases d'identification de besoins spécifiques à la plateforme qui permettent la définition des mécanismes de collecte de l'information contextuelle et des mécanismes d'adaptation. La collecte se fait par des composants comme les capteurs qui peuvent être physiques (matériel pour détecter des données physiques), virtuels (pour dériver les données de contexte d'autres applications logicielles) ou logiques (combinaison de capteurs physiques et virtuels).

Les mécanismes d'adaptation peuvent être basés sur la programmation orientée aspects et les approches orientées composants. Dans la quatrième phase, pour chaque classe présentant un comportement ou une structure variable, il faut définir le mécanisme d'adaptation à appliquer.

Ces différentes phases sont ensuite suivies par une phase IDM qui doit spécifier la plateforme cible et transformer les modèles Platform Independent Model (PIM) en des modèles spécifiques à la

plateforme en utilisant les mécanismes identifiés dans les phases 3 et 4.

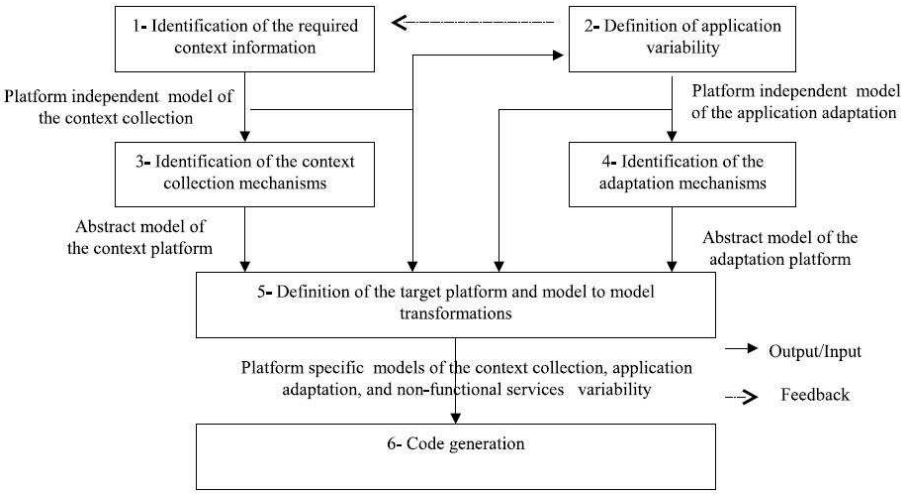


FIGURE 2.13 – Processus de développement d'applications sensibles au contexte [Ayed *et al.*, 2007]

Cette démarche est assez générique et permet la construction d'applications ubiquitaires adaptables. Cependant, elle n'est pas complète puisqu'elle ne prend pas en compte les étapes de conception d'applications classiques comme l'analyse fonctionnelle et technique de l'application et l'intégration des besoins d'adaptation dans les modèles produits. De plus, les phases de définition des besoins contextuels et des besoins d'adaptation ne bénéficient pas de guides et de consignes pour faciliter le travail de spécification et assurer une définition exhaustive des informations contextuelles. Cette approche reste, par conséquent, incomplète et ne prend pas en compte les niveaux de complexité que suscitent les besoins ubiquitaires.

2.3.3 Démarche de conception des applications à base de services web [Kapitsaki *et al.*, 2009]

Dans [Kapitsaki *et al.*, 2009], une méthode de développement des application web sensibles au contexte est proposée. Elle est basée sur une approche IDM utilisant UML comme langage de modélisation. Un ensemble de profils UML (métamodèles) est introduit pour la description des services web, la description des informations du contexte et la définition des propriétés de présentation. Cette démarche de développement permet l'exploitation de services web existants agissant comme des composants réutilisables pour la création d'applications web composites. Une séparation entre l'application et la sensibilité au contexte est réalisée via l'usage de service web sensibles au contexte et encapsulant l'application. Cette séparation est reflétée dans le processus de développement lors des phases d'analyse et de conception. La démarche est ainsi constituée des phases suivantes :

- importation des modèles de services existants en notation UML. Cette phase consiste à réutiliser des services existants pour le développement de l'application.
- conception de l'application dans un environnement de modélisation. Dans cette phase, il s'agit de modéliser l'application (avec les modèles importés), les dépendances entre les informations du contexte et les services métier et les opérations d'invocation des services.
- transformation pour une plateforme spécifique. Cette phase consiste à transformer les modèles obtenus en code source et à générer un ensemble de fichiers de configuration pour l'exécution de l'application.

Cette démarche fournit des guides et des profils UML pour la composition de services et la réalisation de la couche interface pour communiquer avec l'utilisateur. Cependant, le champ d'application de cette démarche est restreint puisqu'elle est dédiée seulement aux systèmes ubiquitaires qui concernent les applications sensibles au contexte composées par des services web. De plus, les aspects techniques de l'application tels que les architectures et les modes de communication ne sont pas évoqués dans la démarche.

2.3.4 Démarche IDM utilisant les ontologies [Serral *et al.*, 2010]

La démarche proposée par [Serral *et al.*, 2010] utilise une approche IDM où le contexte est la partie la plus intéressante. Cette démarche est basée sur un ensemble de concepts utilisés dans les différentes phases de développement. Elle est centrée sur l'usage d'un langage spécifique au domaine appelé Pervasive system Modeling Language (PervML) qui permet une description abstraite des fonctionnalités du système ainsi qu'une expression des besoins contextuels. Les modèles obtenus sont transformés en ontologies pour être implémentés dans le système. L'adaptation au contexte est basée entièrement sur les spécifications OWL. Un framework est utilisé avec un moteur de transformation pour traduire les modèles PervML en code Java.

De façon plus détaillée, la démarche de [Serral *et al.*, 2010] contient une phase de développement et une phase de déploiement. La phase de développement est illustrée par la figure 2.14 et contient les étapes suivantes :

- **modélisation conceptuelle :** le langage PervML est utilisé pour modéliser différents aspects du système : les fonctionnalités, les informations du contexte et les services disponibles pour chaque utilisateur.
- **génération du code :** deux transformations de modèles sont appliquées pour traduire les modèles PervML en code Java et en ontologies OWL. La première transformation consiste à implémenter les fonctionnalités supportant les services du système dans le framework ce qui permet d'aboutir à une architecture commune pour tous les systèmes développés avec cette méthode. La deuxième transformation permet de traduire les informations du contexte en des spécifications OWL mises à jour lors de l'exécution. L'usage des ontologies facilite le raisonnement sur le contexte et l'adaptation du système.
- **implémentation des drivers :** Les drivers utilisés pour la gestion et la sélection des dispositifs doivent être implémentés manuellement puisqu'ils dépendent de questions technologiques. Il est parfois utile d'utiliser un répertoire de drivers et d'invoquer le driver adéquat à chaque fois qu'un dispositif est connecté.

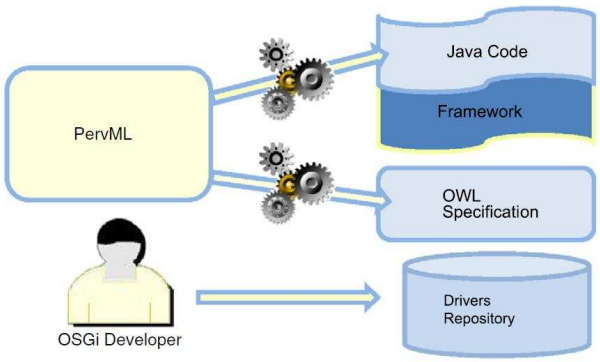


FIGURE 2.14 – Phase de développement centrée langage PervML [Serral *et al.*, 2010]

Dans la phase de déploiement un serveur OSGi est utilisé pour exécuter le système en déployant le code de l'application obtenu à l'issue de la phase de développement. Tout d'abord, le système doit être configuré en implémentant manuellement les drivers dans le code Java. Ensuite, les fichiers Java doivent être compilés et organisés dans des bundles. Les spécifications OWL doivent être copiées dans le même emplacement que le serveur Open Services Gateway initiative (OSGi). Finalement, les bundles installés sont démarrés.

Cette démarche est basée sur un ensemble riche de modèles et prend en compte aussi bien les aspects contextuels que les aspects fonctionnels et les aspects techniques. Cependant, les étapes conceptuelles ne sont pas guidées et les difficultés rencontrées lors de la modélisation du contexte ne sont pas évoquées ou considérées dans des métamodèles de haut niveau pour faciliter la tâche des concepteurs. De plus, l'architecture des systèmes cibles est assez rigide puisque le développeur doit utiliser un intergiciel OSGi et respecter une architecture prédéfinie.

2.3.5 Démarche dirigée par les modèles pour la création de services ubiquitaires [Achilleos *et al.*, 2010]

La méthode proposée dans [Achilleos *et al.*, 2010] utilise une approche IDM pour fournir un framework de modélisation supportant un processus de développement de services ubiquitaires. Le processus de développement est divisé en un ensemble de quatre séquences qui correspondent à une approche IDM classique (comme définie par l'OMG) à savoir : la définition d'un langage spécifique au domaine, la définition et la validation du modèle de domaine, la transformation de modèle à modèle et la transformation de modèle à code. Chacune de ces phases contient un ensemble d'étapes comme illustré dans la figure 2.15.

La première phase consiste à définir un langage spécifique au domaine en identifiant et transformant la sémantique du domaine en un ensemble de constructeurs formant un métamodèle (ensemble d'éléments, de relations et de propriétés du domaine). Ce métamodèle doit satisfaire un ensemble de contraintes dérivées des règles du domaine. Ce métamodèle est ensuite utilisé dans la deuxième phase pour produire des modèles cohérents de l'application. Cette méthode fournit un framework générique pour définir les langages du domaine et construire les modèles de l'application. Ce framework permet de mettre en œuvre les contraintes ce qui garantit la définition de modèles valides (cohérents et complets).

La troisième phase consiste à appliquer des transformations « modèle à modèle » pour transformer les modèles obtenus avec le langage spécifique au domaine en des modèles spécifiques à la plateforme. La dernière phase concernant des transformations « modèle à code » permet de générer le code de l'application à partir des modèles spécifiques à la plateforme. Il s'agit de la phase responsable de l'implémentation.

Cette démarche est générique et a été appliquée dans [Achilleos *et al.*, 2010] pour la création d'applications sensibles au contexte. Elle a permis la création d'un métamodèle générique de contexte et la définition d'un processus de modélisation et d'implémentation du contexte. Par contre, elle ne fournit pas de guides et de consignes pour la modélisation du contexte et l'analyse des besoins et elle se contente de fournir un métamodèle pour modéliser le contexte.

Un deuxième aspect rend l'application de cette démarche difficile et complexe. En effet, une application ubiquitaire comporte, en plus des besoins contextuels, des aspects fonctionnels, interactionnels et techniques. Le développement de ces aspects n'est pas abordé dans cette méthode et la réalisation d'un système ubiquitaire complet n'est pas évoquée. De plus, le développement de chaque aspect de l'application séparément et son implémentation en code ne garantit pas la cohérence de l'application entière. Ainsi, cette approche, bien que générique, ne permet pas un développement complet d'une application ubiquitaire et ne prévoit pas l'assemblage des différents résultats.

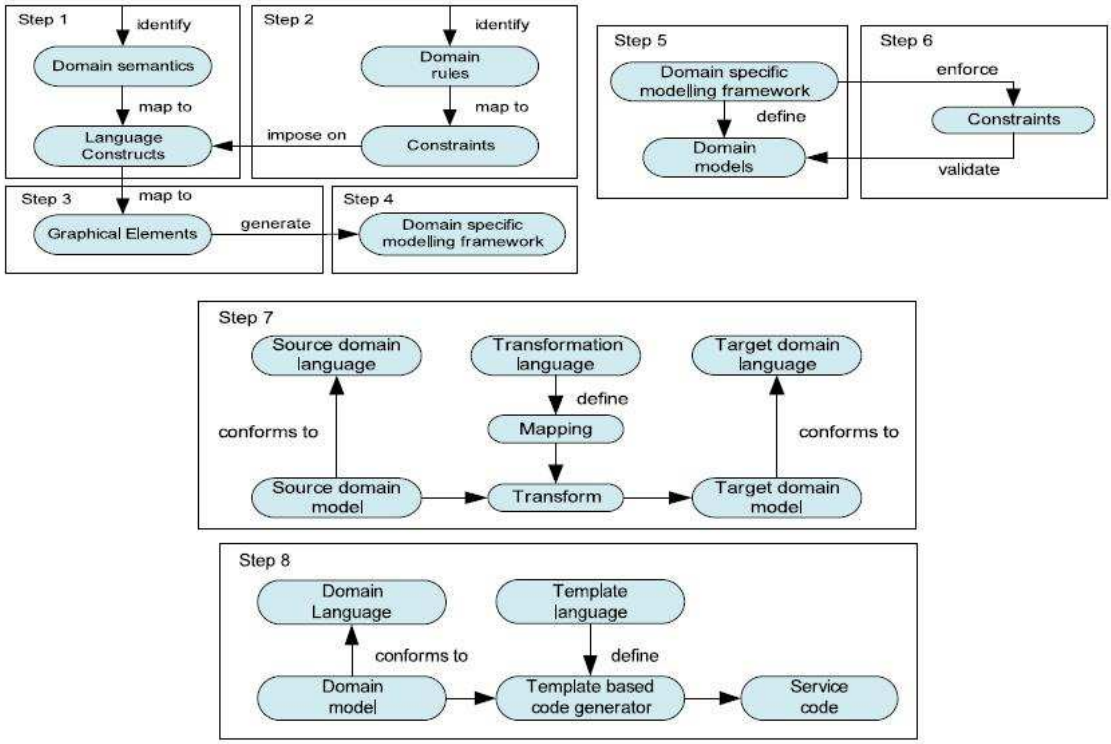


FIGURE 2.15 – La démarche IDM pour la création de services ubiquitaires [Achilleos *et al.*, 2010]

2.3.6 Démarche respectant le cycle de vie du contexte [Vieira *et al.*, 2010]

L'approche proposée dans [Vieira *et al.*, 2010] s'intéresse au développement de systèmes sensibles au contexte (CSS : Context Sensitive System) en se basant sur la logique d'un cycle de vie de contexte à trois phases : spécification, gestion et usage du contexte. La spécification et l'usage du contexte sont dépendants du domaine alors que la gestion du contexte est indépendante du domaine. Ainsi, dans le processus de développement proposé, la conception du gestionnaire de contexte est indépendante des aspects métier de l'application. Cette logique définit l'architecture générique proposée pour un système sensible au contexte et qui contient un bloc central de gestion du contexte (y compris des modules pour l'acquisition, le traitement, le stockage et la distribution) et plusieurs liens avec les sources du contexte en entrée et avec les consommateurs du contexte en sortie. Cette vision du contexte fournit un processus de développement à trois phases, présenté par la figure 2.16, respectant la notation et la terminologie proposées par le métamodèle Software Process Engineering Metamodel (SPEM) de l'OMG. Trois rôles sont considérés dans ce processus : le concepteur du système (responsable de l'architecture du système), l'analyste du système (responsable de l'identification des besoins des utilisateurs et des exigences métier) et le concepteur du contexte (responsable de l'identification des besoins contextuels et de la conception des solutions pour réaliser la sensibilité au contexte).

La première phase du processus (voir figure 2.16) est la spécification du contexte qui consiste à utiliser les exigences métier pour identifier les exigences contextuelles par la collaboration entre le concepteur du contexte et l'analyste du système. Elle permet de produire à partir des modèles de cas d'utilisation l'ensemble des focus du contexte (le concept "focus" a été expliqué dans le métamodèle de contexte de [Vieira *et al.*, 2010] dans le chapitre précédent), puis il faut identifier les variations du comportement pour s'adapter au contexte. Après, il faut utiliser le métamodèle de contexte pour identifier les éléments et les entités du contexte. Finalement, la pertinence des éléments de contexte

doit être évaluée en vérifiant la compréhension de ces concepts à la fois par les utilisateurs et les concepteurs.

La deuxième phase concerne la conception du gestionnaire du contexte au cours de laquelle le concepteur du contexte doit identifier la façon d'acquérir, gérer et livrer les éléments du contexte. Pour l'acquisition du contexte, le concepteur doit définir les sources du contexte et les besoins d'implémentation du module d'acquisition. La gestion du contexte est spécifiée en définissant les règles contextuelles, les spécifications du module de traitement et une version à jour du comportement vis-à-vis du contexte. La conception du module de livraison doit définir les éléments responsables de la livraison des éléments du contexte aux consommateurs.

La troisième et dernière phase a pour objectif de supporter l'adaptation du comportement et d'enrichir la sensibilité à l'information contextuelle. Cette phase doit, tout d'abord, concevoir le modèle de comportement qui consiste à faire le lien entre le focus, les éléments de contexte et les variantes de comportement. Ensuite, il faut concevoir l'adaptation au contexte qui consiste à définir les spécifications du module d'adaptation. Dans ce cadre, il faut concevoir la présentation du contexte qui concerne les éléments de contexte gérés et leur livraison aux agents concernés.

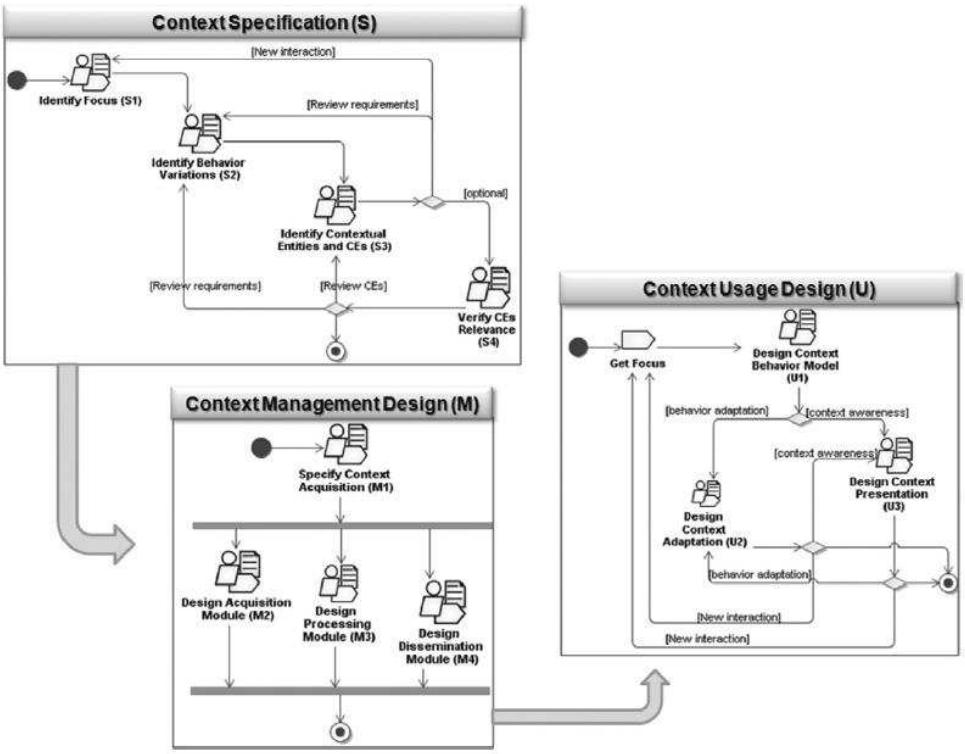


FIGURE 2.16 – Démarche pilotée par le cycle de vie du contexte [Vieira *et al.*, 2010]

Cette démarche de conception se base sur un ensemble de concepts pertinents tels qu'une vision originale du contexte, un métamodèle générique de contexte, une architecture à trois compartiments et une logique de raisonnement sur le cycle de vie du contexte. De plus, le rôle du concepteur du contexte est très utile pour mener le travail de conception d'une application sensible au contexte. Cependant, cette démarche est trop rigide puisque la conservation du cycle de vie de contexte pour la conception n'est pas judicieux. En effet, l'usage du contexte est en relation directe avec la spécification du contexte puisque ce sont les besoins d'usage qui permettent d'identifier les informations de contexte : l'ordre des phases n'est donc pas approprié. De plus, on remarque, dans cette démarche, une interférence

entre les aspects conceptuels et architecturaux, ce qui risque de produire des modules spécifiques et figés et ne permet pas de changements de la spécification et de l'adaptation au contexte. Finalement, la démarche n'évoque pas les aspects métier, interactionnels et données de l'application et ne permet pas de concevoir l'intégration du contexte dans ces types de modèles, ni de définir les spécificités d'adaptation pour chacune de ces couches du système.

2.3.7 Environnement pour la conception et la gestion du contexte [Cipriani *et al.*, 2011]

L'article [Cipriani *et al.*, 2011] propose un environnement logiciel pour la conception et la gestion du contexte utilisé par une application sensible au contexte. Cet environnement, appelé Nexus, est commun à plusieurs applications à la fois et permet de fournir les modèles et les schémas de contexte nécessaires à ces applications pour réaliser l'adaptation au contexte. Concernant les exigences spécifiques au domaine de ces applications, la plateforme Nexus permet l'intégration de données de contexte spécifiques au domaine par extension des schémas de données existants. Les données de contexte sont sauvegardées sur des serveurs avec lesquels il est possible de communiquer via une interface permettant l'expression de requêtes et la sélection de données de contexte. Un éditeur Nexus fournit une interface graphique permettant la modélisation des schémas, la conception des requêtes, la modélisation des données et les tests.

Cette approche ne fournit pas une démarche d'ingénierie, mais plutôt un environnement facilitant la conception et la gestion du contexte selon un scénario d'usage présenté par la figure 2.17. L'expert Nexus est responsable du développement et de la maintenance de la plateforme et de l'éditeur. L'expert du domaine intègre son propre modèle de contexte et peut demander l'extension du schéma pour l'adapter au domaine. Ensuite, il peut exprimer des requêtes et tester les données du contexte. L'application développée est en communication directe avec la plateforme pour toute consultation et sélection du contexte.

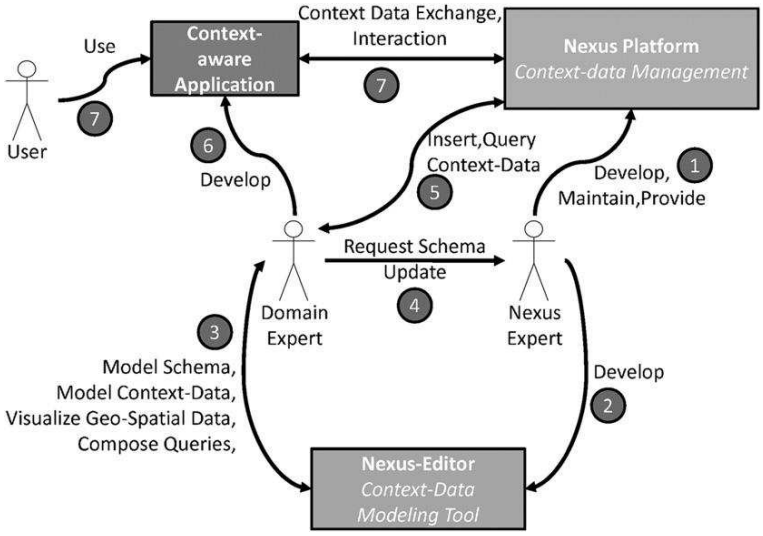


FIGURE 2.17 – Scénario d'usage de l'environnement et de l'éditeur Nexus [Cipriani *et al.*, 2011]

Cette approche peut être utilisée dans des contextes bien précis, où il y a un besoin de partager le contexte entre plusieurs applications avec des visions proches du contexte. Cependant, les données contextuelles deviennent ainsi publiques, ce qui risque de poser un problème de sécurité et de confidentialité. De plus, cette approche n'évoque pas les problématiques d'acquisition du contexte, ni les

topologies de communication entre systèmes sous-jacents, ce qui amène un problème de conception technique non supportée par cette approche.

De plus, les problématiques de modélisation et de conception du contexte sont évoquées de façon superficielle et ne se basent pas sur des modèles conceptuels facilitant la tâche des concepteurs. Finalement, cette approche est totalement indépendante des différentes problématiques de conception d'applications ubiquitaires, surtout par rapport à l'usage du contexte (l'adaptation).

2.3.8 Comparaison entre différentes démarches de développement de SI ubiquitaires

Dans cette dernière partie du chapitre, nous avons présenté différents travaux proposant des démarches de développement de systèmes ubiquitaires. Nous comparons maintenant ces démarches (voir le tableau C.1) selon les critères suivants :

- **le champ d'application :** les spécificités des applications ciblées par la démarche de développement.
- **l'itération :** la possibilité de refaire des séquences du processus en vue de reconsidérer des résultats de développement. L'itération est obligatoire puisque la réalisation parfaite des objectifs du client n'est pas possible. Elle permet d'éviter le déploiement d'un système avant qu'il ne soit conforme aux exigences.
- **les concepts de bases :** l'ensemble des métamodèles, modèles et langages sur lesquels se base la démarche de développement. Ils sont indispensables pour faciliter la tâche des concepteurs et des développeurs et faire valoir les principes de la démarche.
- **la généricité :** la capacité de la démarche à s'appliquer à toute sorte de systèmes ubiquitaires.
- **la complétude :** la capacité de la démarche à prendre en compte tous les aspects de développement d'un système.
- **la dynamicité :** la prise en compte des aspects d'adaptation et de réaction aux changements externes (usage des évènements, des règles et des déclencheurs).
- **l'ingénierie des besoins :** la considération des approches d'ingénierie des besoins pour la définition des exigences ubiquitaires (adaptation, besoins contextuels ou réactivité).
- **le guidage de la modélisation du contexte :** la définition de recommandations et de consignes pour modéliser le contexte (l'usage d'un métamodèle de contexte n'est pas concerné par ce critère).

Toutes ces démarches accordent un intérêt particulier à la sensibilité au contexte considérée comme caractère dominant et unique des systèmes ubiquitaires. Cependant, plusieurs autres critères sont liés à ce type de systèmes tels que la distribution, la mobilité et la disponibilité qui ne sont pas traités dans ces approches. De plus, la plupart des approches se concentrent uniquement sur l'analyse et la conception du contexte en négligeant les aspects fonctionnels et techniques de développement des systèmes, ce qui rend ces approches incomplètes, à l'exception de [Henricksen et Indulska, 2006] et [Serral *et al.*, 2010] qui évoquent ces aspects sans les traiter complètement.

De plus, la plupart des approches proposent des métamodèles de contexte, mais ne fournissent pas de guides ni de consignes pour l'usage de ces métamodèles en vue de faciliter la tâche des concepteurs. Enfin, l'usage des méthodes d'ingénierie des besoins pour définir les besoins ubiquitaires est absent dans toutes les démarches étudiées.

TABLE 2.1 – Comparaison entre démarches de développement de SI ubiquitaires

Démarche	Champs d'application	Itération	Concepts de base	Généricité	Complétude	Dynamicité	Ingénierie des besoins	Modélisation guidée
[Henricksen et Indulska, 2006]	Systèmes sensibles au contexte	Boucles de phases	– Architecture générique – Langage CML – Modèle de préférences – Modèle de règles	oui	oui	oui	non	non
[Ayed *et al.*, 2007]	Systèmes sensibles au contexte	non	– Métamodèle de contexte – Métamodèle d'adaptation – Métamodèle d'acquisition	oui	non	oui	non	non
[Kapitsaki *et al.*, 2009]	Applications web composites et sensibles au contexte	non	– Métamodèle de contexte et services web – Métamodèle de présentation	non	non	oui	non	oui
[Serral *et al.*, 2010]	Systèmes sensibles au contexte	non	– Modèle de services – Modèle structurel – Modèle d'interaction – Modèle utilisateur – Architecture avec OSGi	oui	oui	oui	non	non
[Achilleos *et al.*, 2010]	Services ubiquitaires	non	– Métamodèle de contexte	oui	non	non	non	non
[Vieira *et al.*, 2010]	Systèmes sensibles au contexte	non	– Métamodèle de contexte – Architecture de référence	oui	non	oui	non	oui
[Cipriani *et al.*, 2011]	Applications sensibles au contexte utilisant un système commun de gestion de contexte	non	– Plateforme et éditeur Nexus – Modèle de données – Langage de requêtes	non	non	non	non	non

2.4 Synthèse

Ce chapitre a présenté un aperçu des approches de développement de systèmes ubiquitaires. Nous avons focalisé en particulier sur les approches orientées évènements qui fournissent des bases conceptuelles et architecturales facilitant la construction de systèmes répondant aux différents critères d'ubiquité à savoir la mobilité, la distribution, la dynamicité et la sensibilité au contexte. Les approches évènementielles prennent leur force des avancées technologiques qui supportent les nouvelles capacités caractérisant les systèmes ubiquitaires. Cependant, jusqu'à présent, une prise de recul par rapport à ces capacités, la manière de les concevoir et la prise en compte des exigences des utilisateurs n'a pas été réellement considérée par les travaux de recherche. C'est dans cette perspective que ce chapitre a considéré, en premier temps, les méthodes d'ingénierie traditionnelles de SI et leur adéquation pour le développement de systèmes ubiquitaires. Nous avons montré que ces méthodes, faites pour être génériques, ne permettent pas de supporter les différents critères ubiquitaires.

Dans un deuxième temps, ce chapitre s'est intéressé à un certain nombre de démarches plus spécifiques qui sont apparues durant les cinq dernières années supportant le développement des applications ubiquitaires. Neanmoins, ces démarches présentent des lacunes dues d'une part à la non couverture de tous les aspects ubiquitaires et d'autre part à la non couverture de tous les aspects fonctionnels et techniques de l'application.

L'objectif de ce travail de thèse est de réconcilier les différentes facettes de développement d'un SI ubiquitaire pour définir une méthode bénéficiant des approches orientées évènements pour construire des systèmes complets couvrant tous les critères ubiquitaires. Cette méthode nommée E-CARe (Engineering Context-Aware and Reactive systems) possède une vision originale des systèmes ubiquitaires supportée par un ensemble de concepts de base définis dans le chapitre suivant.

Deuxième partie

Contributions

Chapitre 3

E-CARe : concepts de base

L'état de l'art abordé dans les deux chapitres qui précèdent, montre que les applications ubiquitaires présentent de nouveaux défis logiciels dûs à l'impact des nouvelles technologies. Ces défis sont résumés en ce que nous appelons des besoins ubiquitaires ou encore des caractéristiques ubiquitaires et qui sont : la sensibilité au contexte, la réactivité au contexte, la dynamicité, la distribution et la mobilité. Plusieurs autres caractéristiques peuvent découler de ces derniers telles que la scalabilité, l'interopérabilité et la sécurité. Pourtant, au vu de l'étude de l'état de l'art du chapitre précédent, peu nombreux sont les travaux qui proposent une méthode de développement d'applications ubiquitaires considérant une ou plusieurs de ces caractéristiques.

Pour développer une application ubiquitaire, il ne faut pas se contenter des démarches traditionnelles d'ingénierie, non seulement à cause des caractéristiques ubiquitaires qui entrent en jeu dans le processus de développement, mais aussi à cause d'une nouvelle vision des SI. En effet, les usages ubiquitaires mettent en évidence une nouvelle philosophie où on passe de SI centrés entreprise à des SI centrés utilisateur. L'objectif est de produire des services utilisables et utiles qui profitent au maximum des technologies de communication. Ces services doivent être innovants et intelligents facilitant la tâche à l'utilisateur :

- en minimisant l'intervention de l'utilisateur,
- en cherchant de façon autonome l'information de sources diverses et distribuées (capteurs, autres services, systèmes, etc.),
- en fournissant juste l'information et le service utiles et personnalisés à l'utilisateur.

On peut conclure de ces propriétés et capacités attendues des SI ubiquitaires qu'un besoin imminent d'une nouvelle vision surgit et doit être mis en œuvre par des approches supportant ces propriétés. Nous proposons dans ce chapitre (section 3.1), un framework générique utilisé comme cadre structurant pour les applications ubiquitaires. Ce framework est composé de cinq modules et supporte une architecture innovante grâce à l'introduction d'un module pour le contexte. Les différents composants de ces modules communiquent par échanges d'évènements (voir section 3.2), ce qui présente des avantages comme l'indépendance des bases de données, la circulation optimisée et rapide de l'information et la dynamicité. L'architecture orientée évènements supporte des caractéristiques comme la distribution, la scalabilité et la mobilité.

Concernant le reste des caractéristiques ubiquitaires liées à la sensibilité et à la réactivité au contexte, ce chapitre propose, dans la section 3.3, deux métamodèles permettant la modélisation structurelle et évènementielle du contexte. Ces métamodèles ont été validés par des expérimentations auprès de chercheurs intéressés par les applications ubiquitaires.

Par ailleurs, de telles applications ne doivent pas être développées loin des exigences et des attentes des usagers. En effet, les fonctionnalités ubiquitaires représentent un défi concernant la confidentialité

et les paramètres psychologiques qui peuvent influencer la prise de position de l'utilisateur qui risque de refuser certaines fonctionnalités ubiquitaires. Ces paramètres peuvent être le refus de l'information apportée et manipulée, le manque de confiance dans les services personnalisés et le besoin de réaliser certaines tâches par soi-même (estime de soi). On peut en conclure que l'ingénierie des besoins est primordiale avant tout travail de développement. Dans ce cadre, nous proposons une approche intentionnelle pour supporter la définition des besoins ubiquitaires dans la section 3.4.

3.1 Le framework E-CARe

Un framework est défini par Appleton [Appleton, 2000] comme « une architecture réutilisable qui fournit la structure logique et le comportement d'une famille d'abstractions logicielles, dans un contexte qui spécifie leur collaboration et leur utilisation à l'intérieur d'un domaine donné ». Ainsi, il sert à simplifier le développement des applications et à faciliter leur maintenance. Pour qu'il s'adapte à tous les domaines d'application, le framework doit être le plus simple et évolutif possible. Ainsi, la structuration générale de haut niveau est nécessaire pour simplifier la réalisation d'architectures détaillées respectant ces structures et spécifiques à divers types d'applications.

3.1.1 Présentation du framework

La méthode E-CARe s'appuie sur un framework générique qui permet de présenter une structure abstraite et de haut niveau de l'architecture d'un SI ubiquitaire. Il s'agit d'un ensemble de vues logiques considérant les critères ubiquitaires d'un système et leur position par rapport aux autres modules du SI. Ce framework, présenté par la figure 3.1, contient cinq modules structurant le SI ubiquitaire en cinq vues séparées mais reliées par des échanges d'informations. Le nom et l'ordonnancement des modules découlent des différences entre les besoins d'adaptation et la circulation de l'information contextuelle d'un module à une autre.

Trois modules sont superposés : les modules Présentation, Métier et Gestion des données. Ces modules sont ordonnés selon une architecture 3-tiers qui sépare les aspects présentation, métier et données qui ne partagent pas les mêmes composants logiciels, mais qui sont en liaison par un modèle d'échange. Le module Communication est latéral aux autres puisqu'il permet de gérer les accès et les interactions des parties externes avec les autres modules du système. L'accès peut intéresser les données du système, les métiers et les processus ou encore les interfaces quand il s'agit d'une interaction utilisateur-machine.

En comparant ce framework à une architecture N-tiers, le module Contexte constitue l'originalité de ce framework. Ce module est essentiel pour concevoir des systèmes flexibles et adaptables et prendre en compte les besoins contextuels. Il n'est pas obligatoirement physique, puisqu'il ne contient, réellement, des composants logiciels que dans le cas de l'usage d'un système de gestion et de collecte de contextes séparé des autres modules du système. Dans le cas inverse, l'acquisition, la gestion et l'usage du contexte sont pris en charge par un des modules classiques (présentation, métier, données et communication), ce qui fait du module contexte un module logique permettant de faire le lien entre le système et son environnement.

Entre deux modules voisins, un échange à double sens est possible et passe par le module contexte. Ce passage n'est pas obligatoire mais met en évidence l'adaptation possible de ces échanges par rapport au contexte. Les échanges ascendants d'informations passent par le module contexte ce qui implique la possibilité d'utiliser des informations et des procédures adaptées au contexte destinées à l'utilisateur final. Cependant, les échanges descendants d'informations ne prennent pas en compte le contexte puisqu'ils reflètent des actions et des données à reproduire et à sauvegarder dans les infrastructures du système.

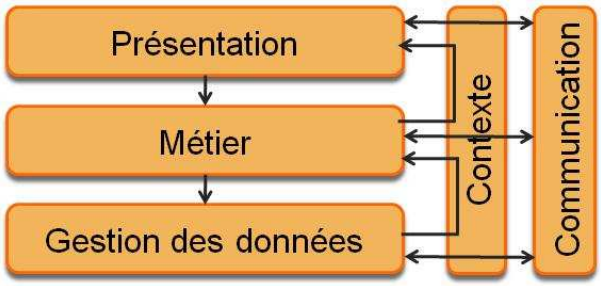


FIGURE 3.1 – Le Framework E-CARe et ses cinq modules

3.1.2 Module Gestion des données

Ce module permet de gérer les données et de les stocker. Tout accès personnalisé à ces données est pris en charge par ce module. Des composants appropriés pour la personnalisation et la gestion des requêtes adaptables au contexte sont utilisés dans le système de gestion des bases de données. Les accès à ce module proviennent soit de l'extérieur du système par des acteurs externes soit pour des besoins internes de l'application via le module métier. Ainsi, pour le premier cas, le module communication gère la connexion entre les parties externes et la module données de l'application en passant par le module contexte. Ce qui signifie que l'accès à ces données dépend du contexte et les données délivrées dépendent à leur tour du contexte de l'utilisateur. Par conséquent, l'adaptation au contexte est à double sens concernant :

- soit les droits d'accès pour désigner une autorisation d'usage du contexte pour des raisons de sécurité et de confidentialité des données du système.
- soit les informations délivrées qui peuvent présenter un contenu adaptable en fonction du profil et du contexte de l'utilisateur pour des raisons d'utilisabilité et d'optimisation du temps et des échanges.

Le deuxième cas d'accès à ce module concerne les métiers de l'application qui ont besoin de données pour leur exécution. L'accès à ces données ne dépend pas du contexte alors que les données délivrées aux métiers peuvent s'adapter au contexte. En effet, certains processus métier ont besoin d'accéder à des données en fonction de paramètres spacio-temporels, tels que les données du réseau de transport qui dépendent du jour de la semaine et d'une géographie bien définie.

3.1.3 Module Métier

Le module métier représente le cœur fonctionnel de tout système et toute application. Il est représenté par l'ensemble des PM et des Règle Métier (RM) définissant et gérant les comportements des différentes entités actives du système. Certaines règles métier nécessitent un raisonnement par rapport au contexte d'exécution ou par rapport au contexte d'un participant ou d'un rôle. Ce sont des règles métier d'adaptation qui permettent de changer le comportement d'un processus ou de déclencher des activités ou des processus.

Le module métier permet de faire le relais entre le module présentation et le module gestion des données. Il a besoin d'accéder aux données du module inférieur et de supporter les interfaces et la navigabilité dans le module supérieur. De plus, ce module est accessible par des parties externes qui sont en général des participants dans les processus métier. Aujourd'hui les besoins d'adaptation des processus métier deviennent une exigence pour certains domaines d'application comme le travail collaboratif et le travail à distance. De même que pour le module gestion des données, le lien entre le module communication et le module métier passe par le module contexte qui permet dans un sens

de contrôler les accès aux processus selon le contexte et dans le sens inverse d'autoriser l'accès à des PM personnalisés en fonction du contexte.

Techniquement, ce module contient le moteur de workflows qui permet l'exécution des PM et autres composants complémentaires pour le pilotage, la maintenance ou la régulation avec un moteur de règles. D'autres composants sont nécessaires pour assurer l'adaptation des PM. Ces composants dépendent de la stratégie d'adaptation adoptée.

3.1.4 Module Présentation

Le module présentation permet à l'utilisateur final d'accéder aux fonctionnalités du SI. Il fait le lien entre le noyau fonctionnel et l'usager. Il rencontre aujourd'hui des défis technologiques liés à l'informatique ubiquitaire et au fait qu'il intègre des interactions avec l'utilisateur final. En effet, la présentation doit être sensible au contexte à son tour pour fournir des interfaces esthétiques, pratiques et compréhensibles. De plus, vu la diversification des dispositifs et des supports de communication existants, les interfaces doivent s'adapter aux caractéristiques techniques de ces supports. L'usage d'un support de communication dépend à son tour du contexte (vidéo projecteur, dispositif mobile, ordinateur de bureau, PDA). Ce type d'adaptation met l'accent sur la plasticité des IHM, devenue un besoin prégnant dans toute application mobile.

Le module communication, pour des raisons de confidentialité et de sécurité, est responsable de gérer les accès aux interfaces qui dépendent du contexte et du profil de l'usager. De même les interfaces affichées pour des raisons d'utilisabilité et d'esthétique dépendent du contexte de l'utilisateur tel que ses préférences, son dispositif, sa localisation ou son environnement (luminosité, température).

Le lien entre le module présentation et le module métier est à double sens. Les métiers supportent les tâches utilisateur sur lesquels se base la composition des interfaces et la navigabilité entre interfaces, alors que le module présentation rapporte les actions et les entrées de l'utilisateur au module métier.

3.1.5 Module Communication

Les modes de communication entre une entité et une autre ou une application et une autre sont diversifiés et dépendent des choix architecturaux et de la nature de la communication. L'objectif de ce module du framework n'est pas d'imposer un mode de communication particulier mais de proposer un cadre logique pour spécifier les modes de communication appropriés tout en garantissant la prise en compte des critères de l'ubiquité. Les modes de communication peuvent être :

- le mode “Pull" consistant à livrer les informations à la demande de l'utilisateur qui exprime des requêtes pour accéder à une donnée ou un service.
- le mode “Push" permettant d'envoyer une information ou un service sans une demande explicite de l'utilisateur. Elle se base parfois sur des stratégies “Publish/subscribe" qui permettent à la partie intéressée de souscrire à un service ou une donnée et recevoir par la suite des informations concernant ces services ou ces données.

Les besoins d'adaptation recensés dans ce module concernent le respect des contraintes d'accès qui dépendent du contexte et qui s'expriment sous forme de règles d'accès ou de contraintes de souscription.

3.1.6 Module Contexte

Le module contexte est un module essentiel pour les applications ubiquitaires. Il permet l'acquisition, la spécification et la gestion des données contextuelles. Concernant l'usage du contexte, il

peut être pris en charge par ce module comme il peut être spécifique à chacun des autres modules présentant un besoin d'adaptation.

Dans le cas de plusieurs systèmes centralisés, le module contexte est commun à plusieurs applications et permet une sauvegarde centralisée des données de contexte. Cette approche ne garantit pas la confidentialité des données puisque la circulation des données personnelles d'une application à l'autre augmente le risque de perte et de piratage de données. Pour résoudre ce problème, notre approche associe à chaque application un module contexte gérant et sauvegardant de façon indépendante les données contextuelles. De plus, dans le cas des applications mobiles, cette approche réduit les besoins d'échanges d'informations qui sont coûteux en terme de dépenses et de temps.

L'adaptation au contexte dans une application ubiquitaire est possible dans tous les modules pour des objectifs différents et par des stratégies variées, ce qui a été montré ci-dessus. Cependant, elle n'est pas obligatoire pour tous les modules.

La spécification du contexte est la première phase du cycle de vie du contexte qui doit être prise en charge par ce module. Elle est essentielle avant tout usage du contexte. Elle consiste à reconnaître les données du contexte collectées, à les typer et à les qualifier de façon compréhensible. Ceci est possible par l'usage d'un modèle de contexte devenu une partie caractéristique de toute application ubiquitaire (cf. la section 3.3).

3.1.7 Synthèse

Cette section a permis de présenter une vision générale de la structure d'un SI ubiquitaire et de ses différents besoins dans une organisation à cinq modules appelée Framework E-CARe. Le contexte est représenté par un module logique jouant le rôle d'intermédiaire entre les différents modules. Ce framework permet de positionner les besoins ubiquitaires dans les SI et de définir des architectures appropriées. Ce framework mettent en œuvre les liens entre modules qui se basent sur une approche orientée évènements, présentée dans la section suivante.

3.2 Une approche orientée évènements

Une étude de l'état de l'art, dans le chapitre précédent, a montré que les approches orientées évènements représentent un choix approprié pour les systèmes ubiquitaires. Le développement de ces systèmes doit prendre en compte cette logique dans toutes les phases du processus pour construire des systèmes homogènes dont les évènements constituent une partie primordiale pour les composants de toutes les modules et pour leurs liens mutuels.

Cette approche permet de répondre à plusieurs besoins des SI ubiquitaires [Ben Cheikh *et al.*, 2010a]. La distribution est supportée par l'usage de plusieurs systèmes qui communiquent par des échanges d'évènements dans une architecture ad hoc. Les communications sont essentiellement asynchrones et les parties participantes peuvent être hétérogènes à faible couplage permettant ainsi la mise en place d'un système distribué.

De plus, les approches orientées évènements permettent d'instaurer une dynamicité dans le système, soit par la réponse rapide et réactive aux évènements ambiants, soit par la circulation optimisée des évènements entre les différents composants et acteurs du système. Chaque évènement est porteur d'une information utile pour une ou plusieurs parties du système. Au lieu de sauvegarder ces informations fortement dynamiques dans des bases de données, nous proposons d'utiliser une approche totalement découplée des bases de données où il est possible de faire circuler l'information portée par des flux d'évènements directement vers les consommateurs de l'information. Une donnée comme la localisation dans un système mobile ne peut être stockée pour après être consultée, car la mise à jour

est assez coûteuse en terme de temps et de coût de communication. Le transfert de cette donnée à la demande est donc une solution plus raisonnable et plus appropriée.

Cette section s'intéresse à la caractérisation des évènements utilisés dans les systèmes ubiquitaires, en particulier leurs usages et leurs liens avec les modules du système (qui correspondent aux modules du framework E-CARe). Elle permet de décrire notre vision de la dynamicité et de la réactivité au contexte à travers l'usage des règles d'adaptation. Cette approche possède des bases techniques qui représentent l'architecture et les topologies de communication adoptées dans la méthode E-CARe.

3.2.1 Les usages des évènements dans les SI ubiquitaires

Les évènements permettent à un système ubiquitaire d'être sensible aux différents changements qui se produisent à l'intérieur de l'application ou dans son environnement externe, dans l'objectif de réagir de façon appropriée dans un temps court ou presque réel. Dans ces systèmes, les évènements déclenchent des réactions qui sont classées en trois catégories qui correspondent à un comportement d'adaptation, à une mise à jour de données contextuelles ou à une notification destinée à des acteurs internes ou externes au système.

Un comportement d'adaptation déclenché par un évènement

Plusieurs évènements qui se produisent dans l'environnement d'une application ou à l'intérieur de l'application même permettent de provoquer des changements dans la structure et le comportement de l'application. Ceci concerne les quatre formes d'adaptation supportées par le framework qui sont l'adaptation du métier, de la présentation, des données et du contrôle d'accès.

Les évènements déclencheurs peuvent être des évènements simples ou des évènements composites respectant un patron d'évènements. Le raisonnement sur ces évènements pour la capture de leur occurrence se fait, en général, par des requêtes continues réalisées par un moteur d'évènements sur un ensemble de flux d'évènements pour valider la détection du patron d'évènements et le choix des réactions appropriées. Par exemple, la détection d'une haute luminosité dans l'environnement du dispositif mobile est réalisée par des capteurs de luminosité qui envoient l'information sous forme d'un évènement à l'application. L'analyse de cet évènement permet de régler les paramètres d'affichage. De même, un évènement métier exprimant l'échec d'une tâche de paiement par carte bancaire, trois fois successives, peut déclencher une modification des droits d'accès de l'utilisateur pour l'empêcher temporairement d'accéder à l'interface de paiement.

Une description du mécanisme basé sur des règles pour l'adaptation est présenté en détail dans la section 3.2.3.

Une mise à jour du contexte déclenchée par un évènement

Les évènements permettent de reconnaître ce qui se produit dans l'environnement de l'application grâce à l'usage de capteurs, de systèmes navigationnels ou aux interactions avec les utilisateurs. Ces données correspondent à des informations de contexte. Ainsi, ces évènements permettent la mise à jour des informations de contexte à chaque fois qu'un changement de l'information est constaté. Les bases de données de contexte, si elles existent, sont sensibles à ces évènements. Dans le cas d'un système totalement dynamique et indépendant des bases de données, les informations de contexte sont envoyées à la demande et c'est toujours la valeur la plus récente qui compte.

Un évènement pour la notification

Certains évènements sont destinés à être analysés et transférés à d'autres parties interagissant avec le système. Les systèmes d'alerte ou de notification représentent des exemples de systèmes où les évènements représentent le seul moyen de communication, et les évènements de notification représentent le premier service visé par les usagers. Ces systèmes jouent le rôle d'un intermédiaire pour détecter certaines données et les diffuser aux parties intéressées sous forme d'évènements de notification. Ils utilisent des modes de communication comme le Publish/Subscribe qui permet aux utilisateurs de souscrire à certains sujets d'intérêt et de recevoir toute notification concernant ces sujets. Par exemple, les applications de transport permettant de notifier des alertes de perturbation, les applications de voyage permettant de diffuser les bons plans et les systèmes d'alarme à domicile sont des exemples de ces applications. La fonctionnalité de notification n'est pas restreinte à ces systèmes mais peut être supportée par toute application ubiquitaire.

3.2.2 Les évènements dans le framework E-CARe

Notre vision orientée évènements considère que tout est évènement [Ben Cheikh *et al.*, 2010b]. Ainsi, toute information en circulation est considérée comme un évènement. Les informations échangées entre l'application et l'environnement (systèmes, acteurs et services) et les interactions à l'intérieur de l'application même représentent un échange d'évènements entre les différents composants et les différents modules.

Les évènements représentent un concept omniprésent dans tous les modules du système. Dans le module métier, les évènements permettent de déclencher des processus, de supporter des passages de décision, de détecter les états d'une action et de finir un processus. Pour le module présentation, les évènements permettent d'exprimer des actions de l'utilisateur ou des déclencheurs d'interfaces. Ils permettent de gérer la navigation entre les interfaces en déclenchant des changements d'interfaces suite à des tâches de l'utilisateur. Pour le module gestion des données, les évènements représentent un moyen de dialogue pour capturer le changement de certaines données et mettre à jour les bases de données concernées. Puisque notre approche est totalement dynamique, ce module ne produit pas ses propres évènements, mais elle est simplement consommatrice d'évènements.

Cette vision des évènements nous a mené à classifier les évènements en trois catégories : métier, utilisateur et externes.

Les évènements métier

Les évènements métier sont générés par le module métier et représentent les états des actions et des processus, les messages produits et les appels de règles métier. Ces évènements peuvent avoir des actions ubiquitaires comme le déclenchement d'une adaptation, la mise à jour de données contextuelles (surtout si les états d'un PM représentent une donnée contextuelle comme dans le cas du travail collaboratif à distance) ou la notification (usage des messages).

Les évènements utilisateur

Les évènements utilisateur concernent le module présentation et permettent de reporter les interactions avec l'utilisateur et leur contenu. Ces évènements sont utilisés par le module métier pour l'exécution des PM et la réification des tâches métier au module présentation. Ces évènements déclenchent des actions ubiquitaires comme les comportements d'adaptation, la mise à jour de données contextuelles (surtout celles concernant les données de profil de l'utilisateur) ou la notification (livraison des informations à l'utilisateur).

Les évènements externes

Les évènements externes représentent l'ensemble des évènements reçus par le système via des parties externes comme les capteurs, les dispositifs ambiants, les services externes et les autres systèmes qui ne sont pas des utilisateurs directs de l'application. Ces évènements déclenchent tout type d'action ubiquitaire : les comportements d'adaptation, la mise à jour de données contextuelles et la notification.

3.2.3 Nouveau paradigme de règles : les règles E-CARe

Une application ubiquitaire doit avoir un ensemble de besoins d'adaptation qui précisent quand et comment une adaptation doit être faite. Nous adoptons l'usage de règles pour l'expression de ces besoins d'adaptation. Notre vision est différente de celle des règles ECA utilisées dans les bases de données actives. Le nouveau paradigme de règles proposé est appelé « règle E-CARe », base de l'approche orientée évènements de la méthode E-CARe.

Présentation des règles E-CARe

Les règles E-CARe représentent des règles contenant trois parties : un évènement qui respecte un patron d'évènement, une condition qui porte sur des données du contexte et une action d'adaptation qui concerne n'importe quel type d'adaptation. Il est possible de représenter ces règles de la façon suivante : Patron d'évènement - Condition Contextuelle -Action d'Adaptation (Event Pattern - Context Condition - Adaptation Action). Ces règles sont déclenchées par des évènements simples ou complexes. L'usage des patrons d'évènements permet la détection d'évènements complexes contenant un ensemble d'évènements avec des conditions sur les attributs de ces évènements. La règle déclenchée doit évaluer une condition contextuelle (ou un ensemble de conditions), qui si elle est satisfaite, permet le choix de l'action d'adaptation appropriée.

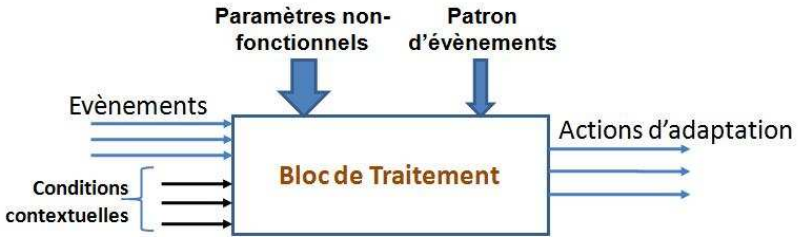


FIGURE 3.2 – Structure d'une règle E-CARe

La figure 3.2 présente la structure d'une règle E-CARe. Voici un exemple d'une règle E-CARe :
- Patron d'évènements : deux retards successifs de bus supérieurs à 5 mn chacun,
- Condition contextuelle : zone de trafic difficile ou heure de pointe ou manifestation prévue,
- Action d'adaptation : notification de perturbation pour la ligne de bus envoyée aux voyageurs intéressés.

Une règle E-CARe peut utiliser de plus des paramètres non fonctionnels qui imposent des contraintes sur les stratégies de traitement telles que le temps d'exécution de la règle ou le mode de consommation des évènements d'entrée.

Les types d'adaptation

Les règles E-CARe, représentant les règles d'adaptation, peuvent être classées en quatre catégories selon le type d'adaptation qu'elles supportent.

- **Les règles d'adaptation métier.** Ces règles génèrent un comportement d'adaptation qui intéresse le module métier et qui consiste à adapter un PM ou un ensemble de ses composants à une certaine situation. Par exemple, il est possible de modifier les rôles responsables de certaines tâches comme dans le cas d'une intervention nécessaire suite à un accident où l'équipe d'intervention est choisie selon la gravité de l'accident et les conditions climatiques. Il est possible de même d'adapter d'autres composants du PM comme l'enchaînement des activités, les données utilisées ou les messages générés.
- **Les règles d'adaptation de la présentation.** Ces règles permettent d'adapter les interfaces hommes-machine à certaines situations dépendantes de la plateforme utilisée, de l'utilisateur même ou de son environnement ambiant. L'adaptation peut concerner les caractéristiques techniques de l'interface (résolution, taille, luminosité), le contenu même de l'interface ou les modes d'interaction (visuelle, sonore, vibrante). Par exemple, les modes d'interaction dépendent des préférences de l'utilisateur, de sa localisation (en réunion, chez lui, en bus) ou aussi d'un éventuel handicap (malentendant, malvoyant).
- **Les règles d'adaptation d'informations.** Ces règles adaptent les données délivrées (par sélection ou adaptation du contenu) au destinataire. Ce type d'adaptation est très courant dans les systèmes de recherche d'information. Les règles E-CARe qui supportent ce type d'adaptation produisent des requêtes enrichies avec des conditions supplémentaires pour la prise en compte du contexte dans la recherche de l'information. Par exemple, le calcul d'un itinéraire du voyageur utilise une règle d'adaptation qui ajoute les préférences de l'utilisateur et ses restrictions de mobilité à la requête de l'itinéraire.
- **Les règles de contrôle d'accès.** Ces règles permettent de déduire les droits d'accès des utilisateurs en fonction de leur contexte. Par exemple, l'équipe d'intervention n'a pas les mêmes droits d'accès aux données de l'entreprise de transport en fonction de l'accident et du véhicule sur lesquels elle intervient. En analysant le contexte, ces règles permettent de produire des recommandations d'accès utilisées par la base de données concernée.

Métamodèle de règles

La spécification et la modélisation des règles nécessitent l'usage d'un métamodèle de règles qui permet de représenter chaque type de règles. Ce métamodèle est illustré dans la figure 3.3 représentant une règle comme un ensemble de trois constituants : patron d'évènements, condition et action. Les concepts de ce métamodèle sont :

- **Règle E-CARe** : qui est représentée par un nom et un identifiant. Chaque règle est reliée à l'ensemble des PM dans lesquels elle peut s'appliquer.
- **Patron d'évènements** : est une situation particulière combinant plusieurs évènements. Elle permet de déclencher la règle E-CARe.
- **Évènement** : une occurrence dans le temps qui indique la réalisation d'un fait intéressant pour l'application et qui permet en collaboration avec d'autres évènements de vérifier une situation appelée patron d'évènements.
- **Source** : qui peut être métier, utilisateur ou externe. Chaque évènement est relié à une source qui représente l'origine de l'évènement et permet de relier la règles aux différentes sources qui la déclenchent. De plus, les sources métier et utilisateurs facilitent le placement des règles dans les processus métier.
- **Condition de contexte** : qui doit être évaluée avant d'exécuter l'action de la règle. Elle permet la considération du contexte avant de décider d'effectuer des actions.
- **Élément de contexte** : est une donnée atomique du contexte. Une condition de contexte est exprimée en fonction d'un ensemble d'éléments de contexte.

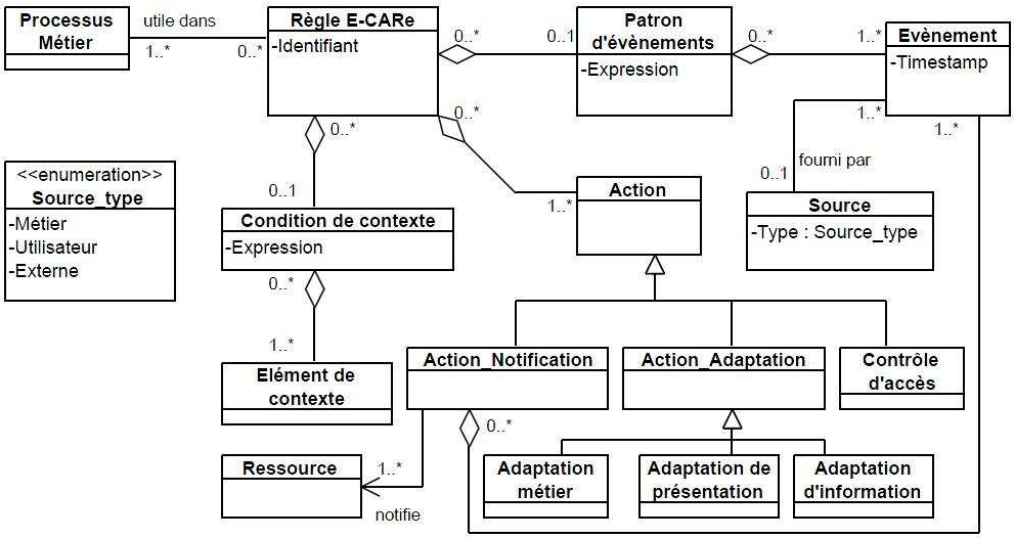


FIGURE 3.3 – Métamodèle de règles E-CARe

- **Action** : est la partie exécutable de la règle qui peut être :
  - *Notification* : permet de notifier une **ressource** externe d'un évènements,
  - *Contrôle d'accès* : permet d'autoriser ou d'interdire l'accès à des données du système en fonction de certaines conditions contextuelles,
  - *Adaptation* : permet d'adapter le système en fonction du contexte et peut être :
    - *Adaptation métier*,
    - *Adaptation de la présentation*,
    - *Adaptation d'informations*.

3.2.4 Architecture orientée évènements

L'approche orientée évènements de la méthode E-CARe s'appuie sur une architecture orientée évènements qui permet d'organiser les composants responsables de l'acquisition, de la gestion et de la diffusion des évènements.

Réseau de traitement des évènements

L'organisation d'un système en un ensemble de sous-systèmes faiblement couplés est possible par l'adoption de l'approche orientée évènements. La définition de ces sous-systèmes doit prendre en compte les besoins de gestion d'évènements qui doivent être cohérents tout en minimisant les échanges d'évènements. Ces systèmes doivent comporter des unités de traitement des évènements pour construire un réseau global de gestion des évènements EPN (Event Processing Network).

La figure 3.4 représente un exemple simplifié d'un réseau de traitement des évènements dans le domaine des transports publics. Les unités de traitement des évènements communiquent directement avec les fournisseurs d'évènements. Chaque unité est responsable de la détection et de la livraison des évènements aux différentes ressources.

Certaines unités sont implémentées dans le SI de transport alors que d'autres unités sont embarquées sur des dispositifs distribués communiquant avec le SI comme les applications mobiles des voyageurs, l'infrastructure des arrêts, et les systèmes embarqués sur les véhicules de transport. Les unités de traitement des données météo, de traitement des données trafic ou de surveillance du réseau

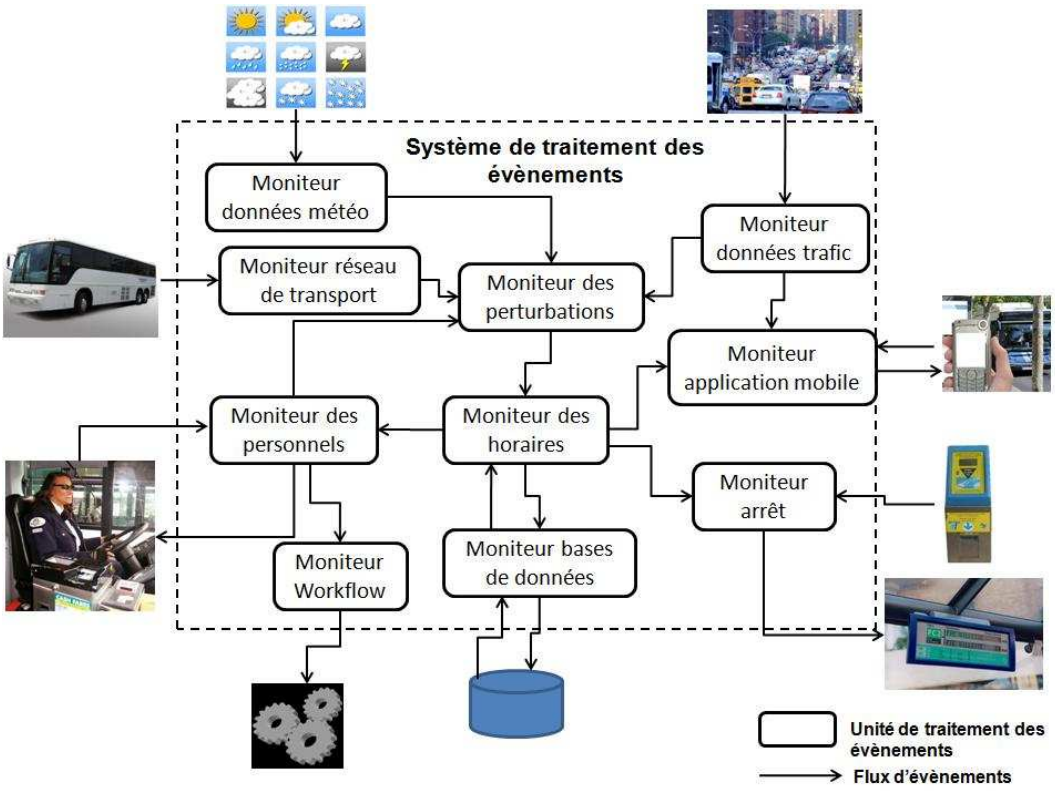


FIGURE 3.4 – Réseau de traitement d'évènements d'un SI de transport

de transport sont incluses dans le SI et communiquent avec des sources externes d'évènements tels que les services météorologiques, les services trafic et les capteurs RFID sur les rails des trams ou les coordonnées GPS des bus.

Pour la définition de ces réseaux EPN, nous préconisons d'associer une unité de traitement des évènements à chaque ressource ou catégorie de ressources permettant ainsi de mieux organiser le cycle de vie des évènements. Une mémoire embarquée dans l'unité permet de sauvegarder les profils des ressources concernées et d'agir directement pour la mise à jour du profil ou la consultation des données du profil. D'autres informations contextuelles intéressantes pour le raisonnement par règles peuvent être sauvegardées localement dans l'unité. Par conséquent, les échanges de données personnelles deviennent minimes, ce qui renforce la sécurité et la confidentialité.

Usage de l'annuaire comme mode de communication

Les systèmes distribués de traitement des évènements représentent de bonnes solutions qui correspondent aux besoins des SI ubiquitaires. Ils permettent de définir des unités de traitement, chacune embarquée dans son propre système. On parle souvent d'unité de traitement des évènements EPU (Event Processing Unit). Chaque unité manipule ses évènements et envoie des notifications aux autres unités. En revanche, il existe souvent un manque de cohérence entre les différentes unités de traitement. C'est dans cette perspective que nous proposons une architecture de système intégré de traitement des évènements qui permet de gérer la communication entre les unités qui échangent entre eux des flux d'évènements. Le système contient un annuaire similaire à l'annuaire UDDI dans l'approche SOA, qui décrit des règles d'accès. Chaque unité est décrite dans le système : ses fonctionnalités, ses produits (les évènements qu'elle génère) ainsi que ses besoins (en évènements). En

connaissant les besoins et les produits de chaque unité, le système peut autoriser des envois de flux dynamiques entre les unités. Des règles peuvent être définies pour accorder des droits d'accès des unités aux évènements produits par les autres unités qui sont les règles de contrôle d'accès.

3.2.5 Synthèse

L'approche orientée évènements de la méthode E-CARe fournit un ensemble d'outils pour caractériser les aspects dynamiques et distribués de l'application. En effet, les évènements dans une application ubiquitaire sont classés en trois catégories : métier, utilisateur et externes et permettent de produire des actions ubiquitaires comme la détection des données du contexte, la notification des parties externes ou encore le déclenchement d'une adaptation au contexte. Pour exprimer les comportements d'adaptation, la méthode E-CARe utilise une structure de règles permettant l'usage des patrons d'évènements pour l'identification des situations complexes et l'évaluation des conditions contextuelles pour déduire un comportement d'adaptation approprié. Ces règles E-CARe peuvent produire une adaptation métier, de présentation, de données ou de contrôle d'accès. Cette vision évènementielle d'un système ubiquitaire est renforcée par une infrastructure évènementielle permettant d'organiser le SI comme un ensemble de systèmes et de sous-systèmes communiquant par des échanges d'évènements, dans une architecture de réseau de traitement des évènements. L'usage d'un annuaire dans cette architecture facilite la découverte et la recherche des évènements par les unités interconnectées du réseau. Ainsi, l'architecture peut s'adapter aux besoins dynamiques et évolutifs de l'application ubiquitaire tout en garantissant la distribution et la scalabilité.

Cette approche orientée évènements est utilisée ci-après pour définir un ensemble de métamodèles permettant l'expression des besoins en contexte et en évènements d'une application ubiquitaire et qui permettent de mieux guider les développeurs dans la construction des modèles de l'application.

3.3 Métamodélisation du contexte

Les modèles de contexte produits doivent être conformes à un métamodèle facilitant l'interopérabilité entre systèmes. D'après l'étude de différents métamodèles génériques de contexte (cf. section 1.2.2), les propositions de la littérature souffrent soit d'un grand nombre de détails qui affectent leur généricité, soit de l'absence de certains concepts nécessaires pour caractériser le contexte d'une entité.

Ainsi dans cette section, nous proposons une vision originale du contexte qui fait intervenir la notion d'évènement comme un concept indispensable pour la construction de modèles de contexte. Cette vision a été mise en pratique dans deux métamodèles étroitement reliées pour modéliser deux vues différentes du contexte : le contexte structurel et le contexte événementielle. Il importe de souligner que ces métamodèles sont le fruit d'une expérimentation réalisée auprès de chercheurs sur deux métamodèles initiaux. Cette expérimentation a permis d'améliorer les métamodèles en matière de complétude, généricité et compréhension par les utilisateurs (voir chapitre 6) et obtenir les métamodèles présentés ci-dessous.

3.3.1 Métamodèle structurel

Le métamodèle structurel, présenté dans la figure 3.5, contient les métaclasses définies ci-dessous.

- **Entité** : toute partie interagissant avec l'application pour consommer un service ou une information ou encore fournir un service ou une information. Pour les systèmes personnalisés, l'entité centrale est l'utilisateur. En général, les entités considérées par l'application sont celles à qui on s'intéresse (ou à une information qui lui est reliée).

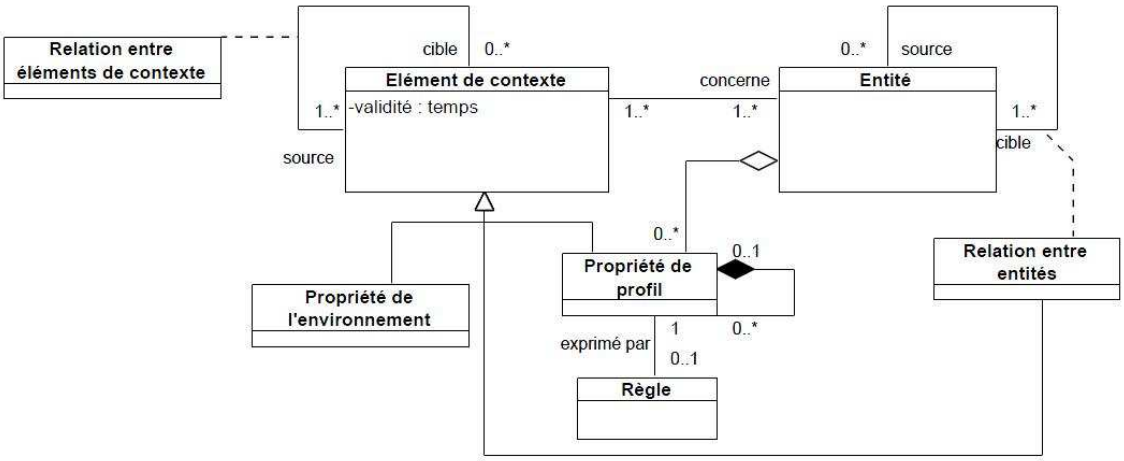


FIGURE 3.5 – Métamodèle structurel de contexte

- **Propriété de profil** : toute donnée caractéristique d'une seule entité. Ce sont des propriétés intrinsèques aux entités comme la localisation, l'âge, les préférences, etc.
- **Règle** : un type particulier de propriété de profil qui permet d'exprimer les préférences et les données d'une entité sous forme de règles telle que "le matin dès mon réveil, je prépare un café".
- **Relation entre entités** : toute relation caractéristique de deux entités ou plus à la fois telle que les relations d'amitié, les relations partie à ensemble (département d'une entreprise), ou les relations fonctionnelles (voyageur utilise dispositif mobile). Une telle relation permet de fournir une information utile sur le contexte d'une entité.
- **Propriété de l'environnement** : concerne des données globales caractérisant indirectement une entité. Ces propriétés sont reliées généralement au temps et à l'espace et permettent d'influencer le comportement de l'application (jours fériés, météo ou encore grèves).
- **Élément de contexte** : toute donnée atomique permettant de caractériser le contexte d'une entité, d'une application ou d'un système. Elle peut être une propriété de profil, une relation entre deux ou plusieurs entités ou une donnée de l'environnement.
- **Relation entre éléments de contexte** : une association entre deux éléments de contexte ou plus qui permet de mettre en évidence des liens comme la dérivation et l'abstraction.

Pour résumer, le contexte, qui n'apparaît pas explicitement dans ce métamodèle, est un ensemble d'éléments de contexte qui concernent soient l'environnement, soient certaines entités internes ou externes à l'application et leurs relations. Cette vision est générique et permet de couvrir tous les types de contexte en considérant qu'une entité peut être un utilisateur, un système, un dispositif, une application ou même un service.

Les relations entre ressources sont importantes pour caractériser le contexte. Contrairement à l'approche de [Zimmermann *et al.*, 2007], dans ce métamodèle, les relations entre ressources ne font pas partie du profil pour permettre au système de gérer plus facilement ces relations sans modifier les profils qui sont dans la plupart des cas détenus par des systèmes distribués couplés aux entités.

Concernant les relations entre éléments de contexte, elles servent à définir si certains éléments de contexte sont reliés par des relations de dérivation ou d'abstraction à des autres éléments. Ceci facilite dans plusieurs cas la manipulation et l'acquisition de certains éléments de contexte à partir d'autres.

3.3.2 Métamodèle évènementiel

Le métamodèle évènementiel est présenté dans la figure 3.6 et les classes sont définies ci-après.

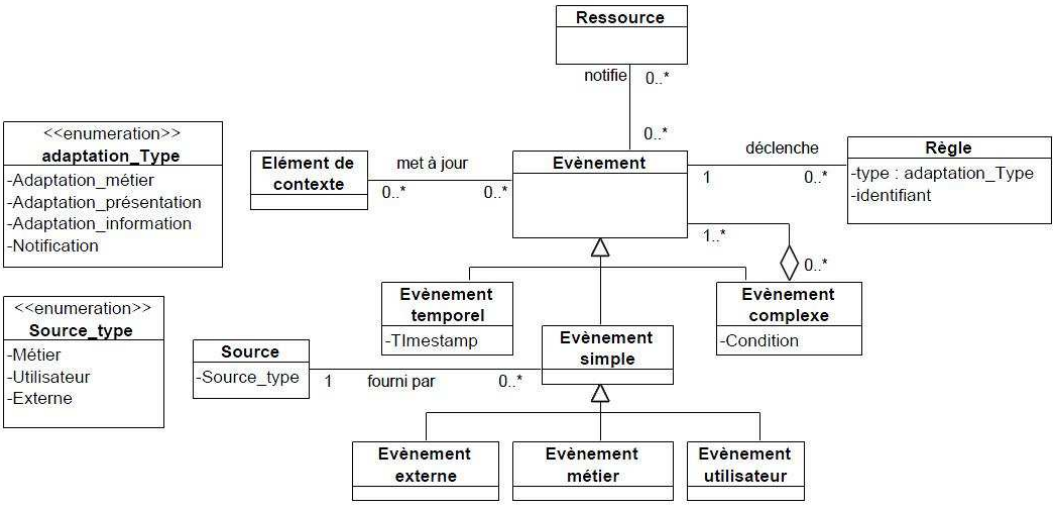


FIGURE 3.6 – Nouveau métamodèle évènementiel de contexte

- **Évènement** : groupe toutes les catégories d'évènements qui sont définies précédemment. Les évènements doivent être modélisés avec leurs actions. Un évènement peut être utilisé pour collecter des éléments de contexte et mettre à jour leurs valeurs. Ils peuvent déclencher une règle E-CARe comme ils peuvent notifier une ressource.
- **Source** : l'origine de l'évènement simple. Ce concept est gardé abstrait pour supporter tout type de sources (systèmes, acteurs, PM).
- **Évènement temporel** : un évènement fournissant une information liée au temps telle que l'heure ou la date ou un concept temporel (ce matin, le week-end, les vacances, etc.).
- **Évènement complexe** : une composition d'autres évènements avec des conditions sur les attributs de ces évènements. Un évènement complexe doit être conforme à un patron d'évènements définissant sa structure.
- **Évènement simple** : un évènement élémentaire produit par une source définie et qui n'a pas subi des traitement de composition. Il peut être :
  - *Évènement externe* : évènement fourni par un acteur externe (service, système ou utilisateur externe).
  - *Évènement métier* : évènement produit par un PM.
  - *Évènement utilisateur* : évènement produit par une interaction avec l'utilisateur.

Un évènement peut être temporel, complexe ou simple. Les évènements complexes sont produits dans le système à partir d'autres évènements alors que les évènements simples proviennent d'autres sources ce qui permet de les classer en évènements métier, utilisateur ou externes.

La métaclasse "Élément de contexte" est commune aux deux métamodèles permettant ainsi de les relier, mais de les représenter de façon séparée pour des raisons de développement décrites dans le chapitre qui suit.

3.3.3 Usage des métamodèles

Nous présentons ci dessous des "règles de bon usage" des métamodèles pour faciliter le travail d'instanciation.

- Règle 1 : commencer par le métamodèle structurel,
- Règle 2 : pour le métamodèle structurel, commencer par l'identification des entités interagissant avec l'application ou utilisées par l'application. Par exemple dans le cas d'une application centrée utilisateur, il faut garder l'utilisateur comme entité centrale, puis définir son environnement et ses relations avec son entourage.
- Règle 3 : pour le métamodèle structurel, utiliser les fonctionnalités d'adaptation pour déterminer quelle information contextuelle est utile pour qu'elle soit prise en compte.
- Règle 4 : nettoyer ces informations en définissant des informations élémentaires qui sont les éléments de contexte et choisir à quelles classes il appartiennent (propriété de profil, relation entre entités ou propriété de l'environnement).
- Règle 5 : définir les liens entre les éléments de contexte qui permettent de faciliter leur acquisition par déduction ou abstraction. Ces liens sont utilisés aussi pour abstraire les données de bas niveau et obtenir des éléments de contexte de haut niveau.
- Règle 6 : pour le métamodèle évènementiel, définir les évènements à l'origine de chaque élément de contexte (s'il n'est pas dérivé d'autres éléments) et de les classifier selon leur nature : temporel, simple ou complexe.
- Règle 7 : définir les sources des évènements simples et les classifier en des évènements métier, utilisateur ou externes.
- Règle 8 : nettoyer l'ensemble des évènements pour fusionner les évènements équivalents (provenant de la même source et fournissant les mêmes informations) et créer des évènements intermédiaires formant les évènements complexes.
- Règle 9 : définir des relations de causalité et d'abstraction entre évènements.

Une illustration de l'usage des métamodèles est décrite ci-dessous.

3.3.4 Exemple d'instanciation

L'exemple d'application choisi pour instancier les métamodèles est celui d'une application d'assistance d'une personne âgée à domicile. L'objectif de cette application est de surveiller le patient dans sa vie quotidienne et de reporter des situations d'alerte. Par exemple, il faut surveiller les durées des actions telles que dormir, regarder la télé, entrer dans la salle de bain ou aux toilettes. L'application communique avec un système de monitoring ambulatoire qui surveille le rythme cardiaque, la fréquence respiratoire et la température interne. L'application est installée sur le téléphone portable du patient qui contient des données entrées par le médecin sur ses maladies.

L'application permet en fonction de l'état de santé du patient, de la température ambiante (chaleur, froid, canicule) et des actions quotidiennes du patient (fréquence d'ouverture du frigo pour boire s'il fait chaud, ...) de détecter des situations d'alertes. Une personne avec un frigo vide pendant une certaine durée est une personne qui ne se nourrit pas bien. La figure 3.7 présente le modèle de contexte instanciant le métamodèle structurel.

En suivant les recommandations d'instanciation, on obtient le modèle évènementiel de contexte montré en figure 3.8. Dans ce modèle, l'ensemble des éléments de contexte identifiés dans le modèle structurel sont reliés aux évènements qui les produisent qui sont, dans le cas de l'application, des évènements externes provenant des dispositifs et des capteurs ambiants ou encore des données médicales entrées par le médecin.

3.3.5 Synthèse

Notre choix de l'approche orientée évènements nous a permis de réaliser deux métamodèles pour la modélisation structurelle et évènementielle du contexte. Ces métamodèles sont produits suite à

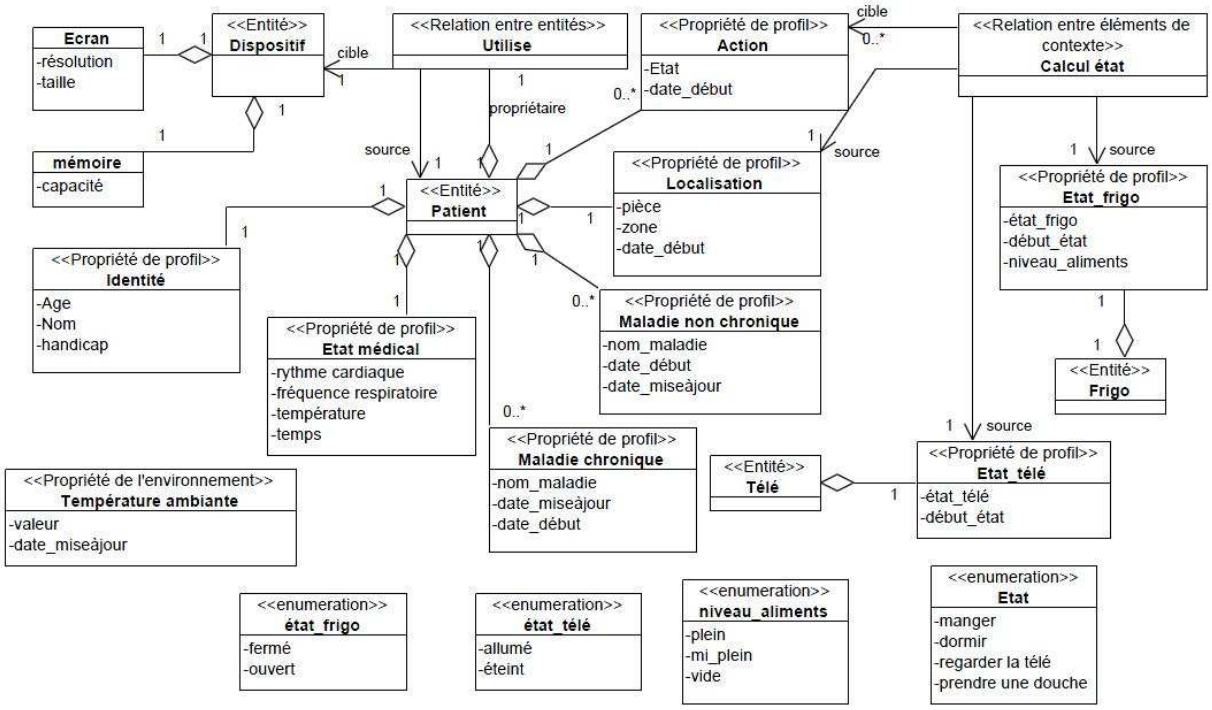


FIGURE 3.7 – Modèle de contexte structurel pour l'application d'assistance médicale

des expérimentations auprès de chercheurs de différentes disciplines informatiques intéressés par les applications ubiquitaires (voir chapitre 6). Cette section fournit une présentation de ces métamodèles et de leur usage pour exploiter au maximum la généricité et la complétude. Mais une identification complète des modèles contextuels et évènementiels doit partir, dans tout projet de développement, d'un ensemble de besoins et de contraintes imposés par les clients. C'est dans cette perspective que nous préconisons d'adopter une approche intentionnelle pour l'identification des besoins ubiquitaires présentée dans la section suivante.

3.4 Approche intentionnelle de E-CARe

Toute méthode de développement doit prendre en compte un processus d'ingénierie des besoins permettant l'identification et l'analyse des exigences des clients et leur usage pour le développement du système. Les méthodes d'ingénierie spécifiques aux systèmes ubiquitaires doivent être capables de répondre aux questions suivantes : quelles sont les attentes des utilisateurs ? quels éléments de l'environnement peuvent influencer l'exécution de l'application ? quelles sont les situations exigeant une adaptation au contexte ? quels sont les composants de l'application sensibles au contexte ? comment exploiter au maximum le contexte pour satisfaire les besoins des utilisateurs ?

Les questions qui précèdent permettent de voir que l'ingénierie des besoins doit supporter la lourde tâche de définir le contexte et ses répercussions sur l'application tout en exploitant au maximum les caractéristiques ubiquitaires pour réaliser les fonctionnalités du système. Ces aspects n'ont pas bénéficié jusqu'à présent de l'intérêt nécessaire pour garantir l'utilisabilité et l'utilité de l'application pour l'utilisateur final. Il est intéressant de constater que les travaux de recherche se focalisent plutôt sur les technologies ubiquitaires, leur intégration dans les domaines d'application et la réalisation de prototypes sans se soucier des besoins des utilisateurs.

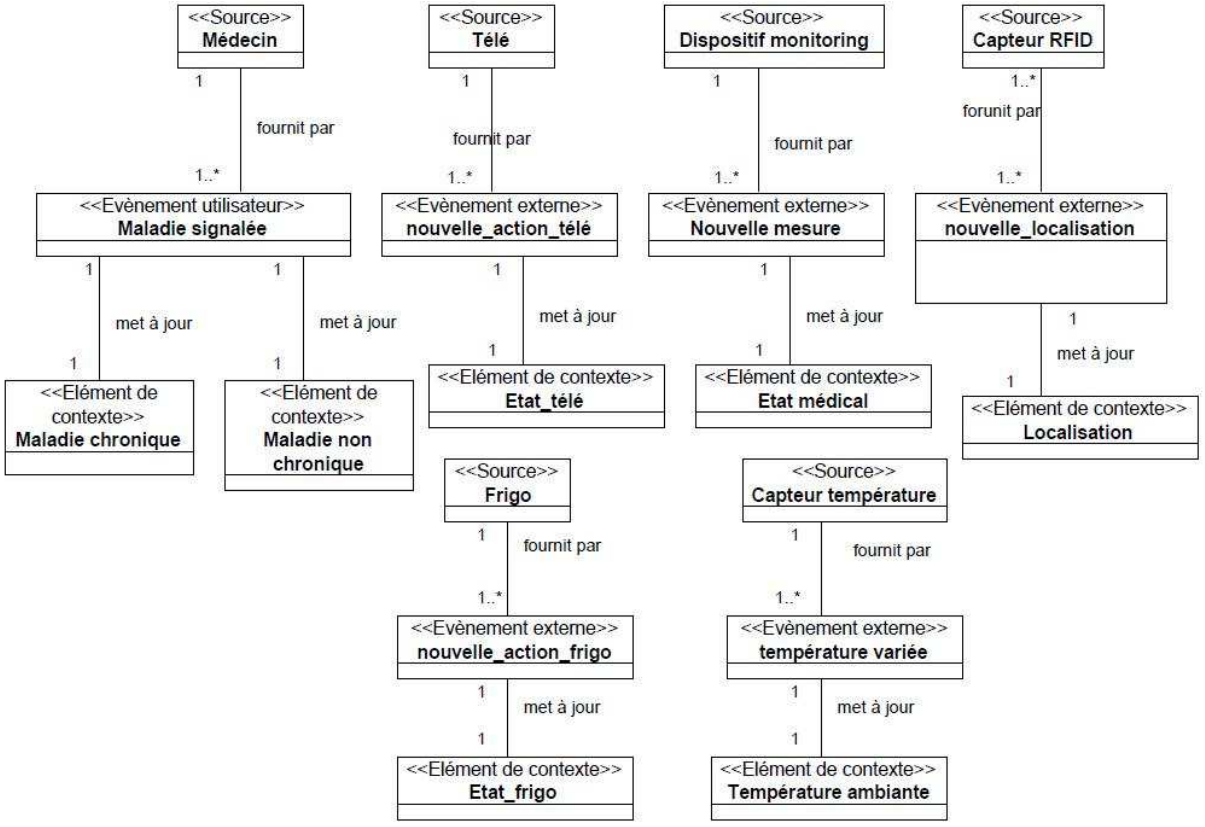


FIGURE 3.8 – Modèle évènementiel pour l'application d'assistance médicale

Parmi le nombre réduit de travaux ayant mis en valeur ce manque et essayé de proposer des approches d'ingénierie des besoins spécifiques aux applications ubiquitaires, nous citons [Kolos-Mazuryk *et al.*, 2005], [Muñoz *et al.*, 2006] et [Pichler, 2007]. Dans [Kolos-Mazuryk *et al.*, 2005], l'idée consiste à utiliser des approches existantes d'ingénierie des besoins, trier celles qui sont adéquates pour les besoins ubiquitaires et les tester pour identifier les besoins et les exigences des utilisateurs. Malheureusement, l'aboutissement et les résultats de ce travail n'ont pas été publiés jusqu'à présent.

L'approche proposée par [Muñoz *et al.*, 2006], consiste à utiliser un ensemble de modèles pour représenter des données du contexte : un modèle pour l'environnement physique, un modèle de tâches (arbre de tâches) et un modèle de comportement (diagramme de séquence UML). Il ne s'agit donc pas de "comment identifier ces données" mais plutôt de "comment les représenter", ce qui ne correspond pas à nos attentes des méthodes d'ingénierie des besoins.

Selon [Pichler, 2007], les chercheurs se sont intéressés aux technologies ubiquitaires sans considérer les avantages et la valeur ajoutée apportée à l'utilisateur, ni essayer de comprendre les exigences et les attentes des utilisateurs vis-à-vis de ces nouvelles technologies. Dans son travail de thèse [Pichler, 2007], Pichler propose une approche "X'Tream Processing" qui consiste à développer un prototype de l'application ubiquitaire et l'utiliser pour simuler son usage et recueillir les idées des utilisateurs sur leurs besoins et leurs attentes.

Ce manque constaté d'approches d'ingénierie des besoins pour les applications ubiquitaires, nous a poussé à créer notre propre méthode intentionnelle (basée sur les objectifs). Cette méthode doit permettre :

– l'identification des objectifs et des attentes des utilisateurs,

- le raffinement et l'analyse de ces objectifs,
- la dérivation des besoins ubiquitaires et des besoins d'adaptation,
- l'identification des données de contexte.

La modélisation du contexte n'est pas concernée puisque notre méthode doit supporter tout langage de modélisation. Donc le rôle de l'approche orientée objectifs s'arrête à l'identification des besoins ubiquitaires en adaptation au contexte. Cette approche est détaillée ci-dessous.

3.4.1 Adéquation des approches basées objectifs pour l'analyse des besoins ubiquitaires

Les approches orientées objectifs pour l'ingénierie des besoins (goal oriented requirement engineering) utilisent les objectifs pour toute action de gestion des besoins : identification, spécification, analyse ou encore modification des besoins. En effet, les objectifs représentent le niveau d'abstraction le plus haut qui permet de générer les besoins sans les connaître au préalable. Les caractéristiques de ces approches montrent leur adéquation pour l'analyse des besoins ubiquitaires et motivent ainsi notre choix.

- **Niveau d'abstraction.** Comme expliqué ci-dessus, les objectifs représentent la vue de plus haut niveau du système qui est compréhensible à la fois par les utilisateurs, les clients et les développeurs. Elle permet de supporter une identification flexible des besoins s'adaptant à tout type de système y compris les systèmes ubiquitaires.
- **Les objectifs comme point de départ.** L'ignorance des fonctionnalités de l'application n'empêche pas la définition des objectifs par les utilisateurs. En effet, parfois l'utilisateur arrive à exprimer le résultat voulu sans pouvoir identifier la fonctionnalité permettant d'y aboutir. Ainsi, les objectifs sont utilisés pour dériver toutes les fonctionnalités de l'application y compris les fonctionnalités ubiquitaires.
- **Imagination et inspiration.** La définition des objectifs permet aux utilisateurs d'imaginer des résultats et des services souhaités sans se soucier des fonctionnalités qui les supportent et de leur faisabilité. Cette imagination est extrêmement demandée dans le cas des applications ubiquitaires qui proposent de plus en plus des services divers, innovants et intelligents.
- **Intérêt accordé à l'exécution.** Les approches orientées objectifs prennent en compte le contexte d'exécution concernant principalement les tâches des acteurs et leurs liens avec les objectifs. Ceci peut être considéré dans le cas des applications ubiquitaires comme un environnement d'exécution faisant partie du modèle de contexte.
- **Le nombre de standards.** Les approches orientées objectifs bénéficient d'un nombre de standards et d'outils important qui peut être exploité pour le développement des applications ubiquitaires.

Concernant ce dernier point, nous avons étudié en particulier l'adéquation des standards Knowledge Acquisition in automated specification (KAOS) [Dardenne *et al.*, 1993] et I* [Yu, 1997] pour la spécification des besoins ubiquitaires. KAOS [Dardenne *et al.*, 1993] fournit des mécanismes robustes et formels pour le raffinement des objectifs et la vérification et validation des besoins. Les concepts principaux de KAOS sont les objectifs, les opérations, les tâches, les actions et les agents, ce qui permet de représenter l'environnement d'exécution et ses relations avec les objectifs. Cependant, l'intérêt pour les autres paramètres de l'environnement et pour les caractéristiques des agents ne sont pas supportés, ce qui ne permet pas de faire le lien avec le contexte ni les besoins d'adaptations.

La méthode I* est un framework pour la présentation et l'organisation des connaissances dans les premières phases de l'ingénierie des besoins. L'approche I* se distingue par un intérêt accordé à la modélisation au profit des techniques de raffinement et d'analyse des objectifs qui ne sont pas supportées. La richesse des modèles générés par I* représente son point fort permettant d'établir des

TABLE 3.1 – Comparaison entre KAOS et I*

Fonctionnalités souhaitées	KAOS	I*
Représentation des concepts	capacité restreinte, environnement non pris en compte	grande capacité de présentation, système dans son environnement
Modélisation des objectifs	modélisation informelle supportée par des modèles formels internes	modélisation basique et informelle
Manipulation des objectifs	méthode pour le raffinement et l'élaboration des objectifs	non supportée
Transition des objectifs aux besoins	supportée	non supportée

relations entre différents concepts : ressources, tâches, objectifs, acteurs et agents. Ainsi, l'expression du contexte et de ses liens avec les objectifs sont possibles.

Le tableau 3.1 compare ces deux approches selon nos besoins et nos attentes d'une méthode d'ingénierie des besoins dirigée par les objectifs. Nous avons choisi d'adopter KAOS vu ses capacités à manipuler et raffiner les objectifs et sa dotation d'un langage formel permettant la vérification et la validation des besoins. Les lacunes de présentation non riche des concepts peuvent être dépassées du moment que nous utilisons nos propres métamodèles qui permettent de couvrir tous les concepts du contexte.

Néanmoins, l'usage de KAOS n'est pas adapté aux besoins des applications ubiquitaires. C'est pourquoi nous proposons une nouvelle approche intentionnelle utilisant et enrichissant KAOS et permet de recenser les objectifs et les besoins ubiquitaires, ce que nous développons ci-dessous.

3.4.2 Nouvelle approche intentionnelle

Vu que KAOS permet de fournir les mécanismes nécessaires à l'identification et au raffinement des objectifs, nous intégrons cette méthode dans l'étape d'analyse des besoins. Ensuite, nous proposons une méthode permettant la dérivation des besoins ubiquitaires y compris les besoins d'adaptation. Ces besoins seront utilisés pour déduire les données du contexte nécessaires pour l'application. Cette démarche d'ingénierie des besoins est détaillée dans le chapitre qui suit. Dans cette section nous présentons les concepts supportant la démarche.

Notre approche intentionnelle repose sur deux principes : le lien entre fonctionnalités et objectifs et le lien entre objectifs et besoins ubiquitaires. Plusieurs travaux, non seulement portant sur l'ingénierie des besoins mais aussi sur la modélisation et la spécification des métiers, se sont intéressés au lien métier-intentionnel. Ainsi, dans [List et Korherr, 2006] et [Ben Cheikh, 2008] la vue intentionnelle positionne le processus dans son environnement en précisant ses objectifs et sa place dans l'ensemble des objectifs de l'entreprise. Chaque PM, quelles que soient sa nature et sa granularité, possède des objectifs supportant les stratégies de l'entreprise.

La définition détaillée de ces objectifs est exploitée dans notre approche pour dériver des besoins ubiquitaires, ce qui manque dans les approches présentées précédemment. Tandis que les autres approches se sont intéressées à la description de l'environnement pour modéliser le contexte et ensuite définir les besoins ubiquitaires (surtout l'adaptation), notre approche commence à l'inverse par extraire les besoins ubiquitaires dans le but de modéliser le contexte tout en considérant le modèle de domaine. Pour identifier ces besoins ubiquitaires à partir des objectifs, ces derniers doivent être raffinés et bien définis. Nous proposons la classification suivante des objectifs :

– objectifs fonctionnels (Hard goals) : expriment un besoin pour réaliser une fonctionnalité per-

mettant de passer d'un état du système à un autre. La taxonomie des verbes utilisée dans l'expression de l'objectif fonctionnel contient les verbes suivants : « accomplir, réaliser, fournir, créer, effectuer, faire, produire, etc. ». Une classification plus détaillée des objectifs fonctionnels et des différentes taxonomies de verbes est fournie dans [Priego Roche, 2011].

- objectifs non fonctionnels (Soft goals) : expriment des mesures qualitatives qui doivent être satisfaites par une fonctionnalité spécifique. Les mesures qualitatives concernent des propriétés non fonctionnelles comme la sécurité, le coût, la performance, la scalabilité, l'interopérabilité, etc. Ces objectifs peuvent contenir des objectifs ubiquitaires non fonctionnels qui concernent la sécurité, la mobilité et la distribution.
- objectifs CARe (Context Aware and Reactive goals) : représentent un besoin d'ajouter une caractéristique ubiquitaire à certaines fonctionnalités du système concernant principalement la sensibilité et la réactivité au contexte. Par exemple, les objectifs « Adapter le processus d'assistance aux empêchements du voyageur »et « Adapter le mode de livraison de l'information aux préférences et à la localisation du voyageur »sont des objectifs CARe. La taxonomie des verbes utilisée ici contient les verbes « adapter, prendre en compte, considérer, fournir, réagir, garantir, détecter, etc. ».

Chaque type d'objectif permet de dériver des besoins différents du système. Concernant les objectifs fonctionnels, ils peuvent être utilisés pour définir des règles métier qui enrichissent les PM. Les objectifs non fonctionnels sont utilisés pour imposer aux PM des contraintes non fonctionnelles exprimées sous forme de règles. Alors que les objectifs CARe permettent de dériver des règles exprimant besoins d'adaptation et de réactivités attachés aux PM concernés. Cette approche utilise KAOS pour la modélisation et le raffinement des objectifs en considérant la classification des objectifs présentée ci-dessus. Concernant le mécanisme de dérivation des règles, il est détaillé dans le chapitre de la démarche qui suit.

Notre approche intentionnelle est illustrée par un métamodèle intentionnel présentant les liens entre PM, objectifs et règles et utilisé dans la démarche E-CARe.

3.4.3 Métamodèle intentionnel

Le métamodèle de la figure 3.9 représente, la vision intentionnelle de la méthode E-CARe.

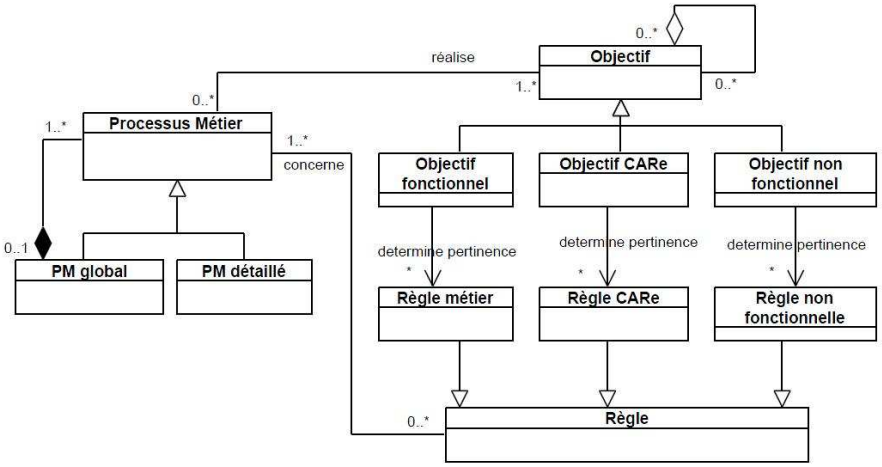


FIGURE 3.9 – Métamodèle intentionnel du SI ubiquitaire

- **Processus Métier :** un processus dont le comportement dynamique vise la réalisation d'un ou plusieurs objectifs.

- **Objectif** : la finalité et le résultat attendus d'un ou plusieurs processus qui regroupe tous les types d'objectifs.
- **Objectif fonctionnel** : une fonctionnalité attendue du PM.
- **Objectif CARe** : la prise en compte ou la réaction au contexte dans un PM.
- **Objectif non fonctionnel** : la prise en compte d'une propriété non fonctionnelle dans un PM.
- **Règle** : regroupe toutes les catégories de règles associées à un PM.
- **Règle métier** : une contrainte métier imposée à un PM.
- **Règle CARe** : un besoin d'adaptation ou de réactivité au contexte satisfait par un PM.
- **Règle non fonctionnelle** : une contrainte non fonctionnelle imposée au PM.

3.4.4 Exemple de vue intentionnelle dans un système de santé

L'application de santé détaillée dans la section 3.3.4 permet d'assister une personne âgée à son domicile. Les objectifs principaux de cette application sont de surveiller la personne et son environnement et réagir rapidement et de façon appropriée à toute situation critique. L'application de notre approche intentionnelle à cette application permet d'obtenir une hiérarchie d'objectifs respectant la classification des objectifs et présentée dans la figure 3.10. Cette hiérarchie est exprimée avec la notation KAOS en utilisant l'outil Objectiver [14]. L'application de santé contient un ensemble d'objectifs fonctionnels qui correspondent aux fonctionnalités souhaitées. Certains objectifs fonctionnels, sont attachés complétés avec des objectifs non fonctionnels et des objectifs CARe.

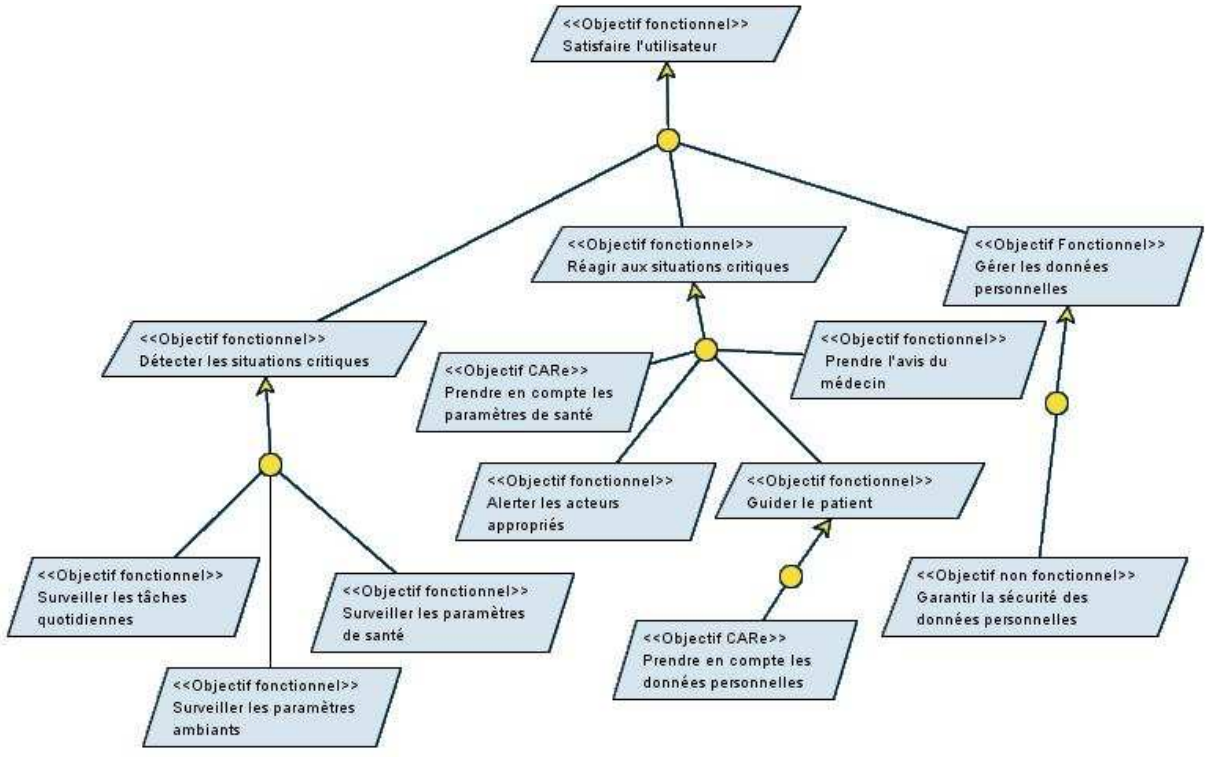


FIGURE 3.10 – Exemple de hiérarchie d'objectifs

Ensuite, il est plus facile de définir les différents PM correspondants à ces objectifs et de dériver des règles associées à ces processus et surtout des règles ubiquitaires CARe. Le processus de spécification intentionnel est fourni en détail dans le chapitre qui suit.

3.4.5 Synthèse

L'ingénierie des besoins est une composante primordiale pour l'ingénierie des systèmes, qui garantit une prise en compte permanente des exigences des utilisateurs ou des clients, ainsi que la considération des contraintes non fonctionnelles. L'approche intentionnelle que nous proposons permet de prendre en compte ces exigences en partant des objectifs exprimés par le cahier des charges ou les utilisateurs finaux de l'application. En effet, il est plus facile d'exprimer des objectifs et des attentes d'une application que d'exprimer directement des fonctionnalités et des besoins. Ces objectifs sont raffinés et classés en trois catégories : fonctionnels, non fonctionnels et CARe. Les objectifs CARe constituent l'originalité de cette approche, ils sont exprimés explicitement à côté des objectifs fonctionnels, permettant ainsi une identification plus facile des besoins ubiquitaires attachés aux fonctionnalités du système. Cette approche est complétée par l'usage d'une méthode d'ingénierie des besoins orientée objectifs comme KAOS, pour le raffinement, le nettoyage et la construction des hiérarchies d'objectifs.

3.5 Synthèse

L'objectif de ce chapitre est de présenter les concepts de base de la méthode E-CARe. Le framework fourni en section 3.1, présente le cadre logique pour la structuration d'une application ubiquitaire. Ensuite, l'approche orientée évènements permet de supporter les caractéristiques ubiquitaires liées aux technologies déployées. Un métamodèle de contexte permet de supporter les autres caractéristiques ubiquitaires qui concernent la sensibilité et la réactivité au contexte. Enfin, une approche intentionnelle est proposée par la méthode E-CARe pour permettre une spécification de tous ces besoins ubiquitaires.

Ce chapitre a permis de dégager les différents concepts de base pour le développement de systèmes ubiquitaires qui prennent en compte toutes les caractéristiques attendues de ces systèmes. Il reste à montrer comment ces concepts peuvent être organisée ensemble pour supporter la construction des applications. Le chapitre suivant présente une démarche d'ingénierie des systèmes qui profite de ces différents concepts pour proposer un processus original et complet dans la mesure où toutes les caractéristiques ubiquitaires sont mises en évidence.

Chapitre 4

E-CARe : démarche de spécification des besoins ubiquitaires

L'objectif de cette thèse est de fournir une méthode facilitant l'ingénierie des applications ubiquitaires. Cette méthode se base sur un ensemble de concepts résumant notre vision et proposant une structuration logique et architecturale de ce type d'application. Ces concepts de base ont été développés dans le chapitre précédent et dépendent naturellement des caractéristiques ubiquitaires qui sont : la sensibilité et la réactivité au contexte, la distribution, la mobilité et la sécurité.

La sensibilité et la réactivité au contexte forment une capacité représentant l'originalité et le succès de ces applications en créant de nouvelles fonctionnalités ce qui justifie notre intérêt accordé à ce point et résultant en un framework à cinq niveaux avec un niveau spécifique au contexte et nécessaire à toute communication entre les modules des différents niveaux.

La sensibilité et la réactivité au contexte sont une conséquence de la distribution et de la mobilité dans les applications ubiquitaires. Ainsi, la gestion et l'usage du contexte possèdent un aspect dynamique résolu grâce à une approche orientée évènements.

Finalement, ces applications sont assez délicates du moment qu'elles manipulent des données privées des usagers ou de l'application même et les utilisent pour prendre des décisions, traiter des données ou filtrer de l'information. Ainsi les fonctionnalités supportées par ces applications doivent obéir à certaines normes de sécurité qui peuvent dépendre de la nature et de la destination de l'application. Par conséquent, il est recommandé d'adopter une approche intentionnelle d'ingénierie des besoins pour mieux identifier les fonctionnalités sensibles au contexte.

Tous ces concepts de base précités doivent être renforcés avec un processus de développement d'applications ubiquitaires pour mieux les appliquer dans un enchaînement logique distribuant les tâches sur les développeurs selon leurs compétences. Les méthodes d'ingénierie des applications doivent donc être considérés tout en intégrant les aspects ubiquitaires.

Ce chapitre se concentre sur le processus de spécification des besoins ubiquitaires dans une démarche que nous appelons la démarche E-CARe (Engineering Context Aware and Reactive systems) [Ben Cheikh *et al.*, 2012].

4.1 Présentation de la démarche

La démarche E-CARe se base sur un ensemble de principes considérés comme des choix facilitant la séparation des tâches des développeurs et la spécification des modèles ubiquitaires. Tout d'abord, nous séparons les phases de la démarche en des phases fonctionnelles, ubiquitaires et techniques. D'ailleurs, la séparation des besoins fonctionnels et techniques a montré son intérêt dans les démarches

traditionnelles de conception de SI telles que 2TUP et Symphony. La création de phases d'analyse et de conception ubiquitaires induit l'intervention d'un nouvel acteur : l'analyste-concepteur de contexte. Ces choix ont motivé la création d'un cycle de vie de la démarche contenant une branche de développement ubiquitaire indépendante des besoins techniques et fonctionnels et qui fait l'objet de ce chapitre.

4.1.1 Séparation des besoins ubiquitaires, fonctionnels et techniques

Tout projet de développement logiciel est susceptible de subir des modifications et des changements continuels imposés par les clients ou par l'environnement (changement de politiques, de technologies ou de stratégies). C'est dans ce cadre, que le développement logiciel doit être réutilisable et les modèles fournis doivent être faiblement couplés pour faciliter les modifications et les mises à jour sans affecter tout le système. La séparation des besoins selon leur nature (fonctionnelle, ubiquitaire ou technique) est la solution adoptée dans notre démarche.

Séparation des besoins fonctionnels et techniques

Les besoins fonctionnels résument l'ensemble des métiers des utilisateurs . Il s'agit d'identifier ce que réalise le système, les services fournis et les interactions nécessaires avec les utilisateurs indépendamment des technologies et des équipements utilisés. Les besoins techniques regroupent l'ensemble des choix matériels et logiciels et leur intégration dans l'environnement existant de l'entreprise. Il est nécessaire de construire une architecture technique indépendante des besoins fonctionnels et utilisée par la suite pour le prototypage du système.

Un SI subit généralement des contraintes fonctionnelles et techniques séparées qui peuvent être traitées de façon parallèle sans affecter la totalité du système. Des approches comme 2TUP [Rocques et Vallée, 2002] et Symphony [Hassine, 2005] considèrent cette séparation en adoptant des démarches de développement avec un cycle de vie en "Y" (voir chapitre 2). La séparation de ces besoins présente les avantages suivants :

- *La réutilisation.* Les composants fonctionnels peuvent être utilisés avec différentes architectures techniques correspondant aux différentes contraintes matérielles et logicielles imposées. De même, les composants techniques conçus peuvent être réutilisés avec des composants fonctionnels différents.
- *La flexibilité.* Les systèmes ainsi conçus se basent sur le principe de découplage entre composants, ce qui facilite l'évolution de chaque composant indépendamment des autres. Ces systèmes acceptent plus facilement les innovations et la créativité des acteurs impliqués dans le projet.
- *Le gain de temps.* Les tâches de spécifications fonctionnelle et technique menées en parallèle permettent de gagner du temps dans les délais du projet puisque les développeurs de disciplines différentes peuvent avancer indépendamment. L'évolution du travail est plus fluide et aucun acteur n'est dépendant de l'autre.

Spécificités des besoins ubiquitaires

En plus des besoins fonctionnels et techniques, les systèmes ubiquitaires comportent des besoins liés à l'ubiquité, la mobilité et la distribution qui permettent d'utiliser des technologies innovantes pour créer de nouvelles fonctionnalités proches de l'utilisateur et de ses attentes. La spécification et l'analyse de ces besoins est loin d'être une tâche triviale et facile puisqu'elle fait intervenir des données personnelles des utilisateurs délicates à manipuler et à utiliser. Ainsi, nous séparons la spécification des besoins ubiquitaires des autres besoins fonctionnels et techniques et proposons une démarche indépendante pour l'identification et l'analyse de ces besoins. Cette séparation est due aux raisons suivantes :

- Les besoins ubiquitaires représentent en général des paramètres ajoutés aux métiers pour déterminer comment une fonctionnalité est réalisée : livrer des résultats, trier des résultats ou imposer des modes d'interaction. Dans ce cas les besoins ubiquitaires ne s'intéressent pas au cœur métier, mais plutôt à la manière de le présenter aux utilisateurs.
- D'autres besoins ubiquitaires font intervenir le cœur métier pour définir différentes manières d'organiser ces fonctionnalités en vue de les adapter à l'utilisateur final. La partie concernant la spécification de ces besoins ubiquitaires peut être traitée séparément du résultat d'adaptation métier qui est étudié dans la démarche purement fonctionnelle. Cette séparation est nécessaire dans la mesure où les données paramétrant un besoin d'adaptation métier permettent de construire le contexte.
- Les données contextuelles qui constituent la base de tous les besoins ubiquitaires sont différentes des données systèmes et n'affectent pas les métiers de l'entreprise.
- Les besoins ubiquitaires liés à la distribution et la mobilité permettent de définir des choix matériels et architecturaux qui sont traités avec le reste des besoins techniques et non dans la branche ubiquitaire dédiée à la sensibilité et la réaction au contexte.
- Les technologies ubiquitaires constituent des besoins traités dans la branche technique et qui n'affectent pas la spécification des besoins contextuels et des fonctionnalités qui en découlent.

Cette vision des besoins ubiquitaires incite à créer une démarche de spécification des besoins de sensibilité et de réaction au contexte qui étudie de façon séparée les différentes possibilités pour bénéficier des données contextuelles afin de fournir des fonctionnalités et des services plus proches de l'utilisateur et de son mode de pensée. Les aspects ubiquitaires concernant la distribution et la mobilité sont naturellement traités dans les spécifications techniques. Le tableau 4.1 résume les objectifs des trois dimensions menées en parallèle à savoir les spécifications : fonctionnelles, ubiquitaires et techniques.

TABLE 4.1 – Oobjectifs des spécifications fonctionnelles, ubiquitaires et techniques

Les spécifications fonctionnelles	Les spécifications ubiquitaires	Les spécifications techniques
– Définir l'organisation métier – Définir les modèles d'interaction – Définir les données métier – Modéliser les processus métier	– Identifier les besoins d'adaptation et d'accès – Définir les situations critiques ou d'alerte – Modéliser les données contextuelles – Définir l'acquisition et l'usage du contexte	– Définir les besoins matériels prenant en compte la distribution et la mobilité – Définir les besoins de sécurité – Définir les choix logiciels (y compris les technologies ubiquitaires)

Ainsi, la démarche E-CARe présente une approche innovatrice pour la spécification et l'analyse des besoins ubiquitaires indépendante des contraintes fonctionnelles et techniques. La prise en charge de cette démarche est assurée par un nouveau rôle nommé l'analyste-concepteur du contexte dont la description est fournie ci-dessous.

4.1.2 Un rôle majeur : l'analyste-concepteur de contexte

Le rôle de concepteur de contexte a souvent été lié au développement de jeux vidéos pour désigner des acteurs responsables de la définition du décor et des arrière-plans. Par ailleurs, ce rôle a été mentionné dans des travaux portant sur les systèmes ubiquitaires pour désigner la personne responsable de la modélisation du contexte [Vieira *et al.*, 2010]. Pourtant cet usage n'a pas été justifié et les responsabilités de ce rôle n'ont pas été délimitées. Dans notre approche nous proposons de mettre en évidence ce rôle qui prend en charge toutes les phases de la branche ubiquitaire E-CARe.

L'analyste-concepteur de contexte est responsable de l'identification des fonctionnalités ubiquitaires supportées par l'application. Ces fonctionnalités font intervenir des données de contexte propres aux utilisateurs ou à l'environnement de l'application. Ainsi, l'analyste-concepteur de contexte doit être capable de définir ces données contextuelles, leurs origines et leurs usages, de les modéliser et de les préparer à être exploitées par les développeurs de l'application. Par conséquent, l'analyste-concepteur de contexte doit être doté des compétences énumérées ci-dessous :

- *Expertise du domaine.* L'analyste-concepteur de contexte doit connaître le domaine de l'application pour pouvoir définir des fonctionnalités ubiquitaires respectant les différentes entités du domaine et leurs liens entre elles. Une bonne connaissance du domaine permet de créer des fonctionnalités originales et innovantes proches de l'utilisateur.
- *Expertise des besoins de l'utilisateur.* L'analyste-concepteur de contexte doit être capable de définir toutes les catégories d'utilisateurs et les besoins de chacune de ces catégories. En effet, les utilisateurs selon leurs âges, leurs niveaux professionnels, leurs personnalités, etc, ne présentent pas les mêmes besoins et attentes d'une application. Ainsi, l'analyste-concepteur de contexte doit être attentif à ces aspects et doit se procurer des mécanismes pour mieux identifier ces besoins et les utiliser pour définir les profils des utilisateurs.
- *Expertise de l'environnement.* L'analyste-concepteur de contexte doit être capable d'identifier les différentes données de l'environnement qui peuvent influencer certaines fonctionnalités de l'application et les utiliser pour définir le contexte de l'application. Cette expertise doit permettre de plus de définir comment exploiter l'environnement pour dériver des données de contexte ce qui représente les mécanismes d'acquisition du contexte.
- *Connaissance des technologies ubiquitaires.* La définition des données contextuelles et des mécanismes d'acquisition requiert une bonne connaissance des technologies ubiquitaires et de la possibilité de les utiliser pour ces finalités. Ainsi, la créativité de l'analyste-concepteur de contexte est limitée par les technologies ubiquitaires existantes.
- *Expertise informatique.* L'analyste-concepteur de contexte participe au développement de l'application et doit être doté des compétences informatiques pour la modélisation et la spécification des besoins. Ainsi, il doit être expert des langages de modélisation pour fournir des modèles formels pouvant être exploités par la suite.

4.1.3 Cycle de la démarche

La démarche E-CARe a été construite et améliorée au fur et à mesure de l'évolution de ce travail de thèse. L'étude de la littérature en terme de développement d'applications ubiquitaires (voir chapitre 2 section 2.3), notre expérience en terme d'ingénierie des SI (voir chapitre 2 section 2.2) et la réalisation d'applications ubiquitaires dans les domaines du transport et de la santé nous ont permis de mettre au point une démarche de spécification des besoins ubiquitaires. Cette démarche, dont le cycle de vie est représenté par le diagramme de la figure 4.1, présente l'aspect original de démarche E-CARe . Les expérimentations effectuées sur les métamodèles de la méthode E-CARe ont permis de supporter l'enchaînement des phases ubiquitaires concernant l'ordre des tâches de modélisation réalisées par les

sujets. Ce cycle de vie est un ensemble de phases successives, et chaque phase est un ensemble d'étapes. La branche ubiquitaire de E-CARe est itérative et elle est répétée pour atteindre la satisfaction du client et suivre l'évolution des besoins.

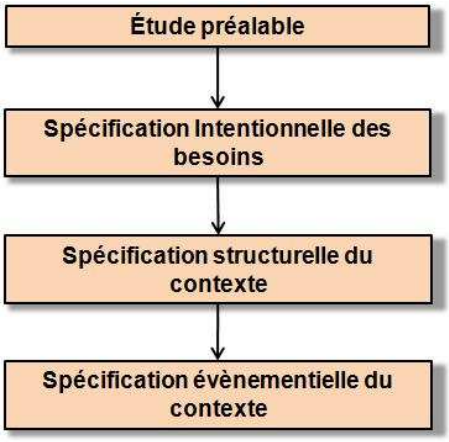


FIGURE 4.1 – Cycle de vie de la branche ubiquitaire de la démarche E-CARe

Le tableau présente une description de chaque phase ainsi que ses entrées et sorties. Le reste du chapitre est une étude approfondie de chaque phase appliquée sur un cas d'étude décrit dans la section suivante.

TABLE 4.2 – Les processus métier globaux et leurs objectifs : une vue intentionnelle de haut niveau

Nom de la phase	Fonction de la phase
Étude préalable	*Entrée :* cahier des charges général. *Description :* permet de faire une étude globale et générale des besoins du client et les placer dans le cadre de l'entreprise et du système visé. *Sorties :* Périmètre métier (ensemble de PM globaux), périmètre organisationnel, modèle de domaine et contraintes techniques.
Spécification intentionnelle des besoins	*Entrée :* périmètre métier. *Description :* permet d'utiliser les objectifs des PM pour générer une hiérarchie d'objectifs fonctionnels, ubiquitaires et techniques pour chaque PM et de dériver des règles du même type. *Sorties :* règles métier, CARe et non fonctionnelles.
Spécification structurelle des besoins	*Entrées :* diagramme de domaine, règles CARe *Description :* permet de raffiner les conditions de ces règles et les utiliser pour dériver des prescriptions contextuelles pour produire au final un modèle structurel de contexte. *Sorties :* règles CARe raffinées, modèle de contexte.
Spécification évènementielle des besoins des besoins	*Entrées :* règles CARe raffinées, modèles de contexte. *Description :* permet de raffiner les règles sous forme E-CARe, de définir un modèle d'acquisition du contexte et de produire une cartographie complète des évènements à actions ubiquitaires. *Sorties :* modèle de règles, Cartographie d'évènements.

4.2 Étude de cas : cahier des charges

Cahier des charges d'un projet de rénovation d'un Système d'Information de Transport

Présentation de l'entreprise de transport Il s'agit d'une entreprise des transports en commun gérant un réseau de bus, trams et de véhicules spécialisés. L'entreprise est responsable d'assurer le service quotidien des différentes lignes de transport. La promotion et la mise en valeur des modes de transport alternatifs à la voiture personnelle font partie de ses objectifs stratégiques.

Description du système actuel Le Système d'Information des Voyageurs (SIV) reste centré entreprise et ne répond pas aux besoins des voyageurs qui exigent une information plus complète et plus fraîche pendant les situations de perturbation. Cette volonté d'améliorer la qualité et la disponibilité des informations destinées aux voyageurs, nécessite la mise en place d'une procédure de rénovation globale du SIV actuel en vue de l'adapter aux technologies récentes centrées utilisateurs pour améliorer la qualité du service informationnel.

Objectifs L'objectif du projet est de fournir un système d'information mobile intégrable facilement dans les structures informatiques existantes de l'entreprise de transport et qui permettent de maximiser la qualité des informations fournies. Ces informations peuvent être utilisées par les voyageurs ou par le personnel en mobilité (conducteurs, équipe de maintenance, etc.) comme compléments et alternatives d'informations en cas d'incidents. Le SI de l'entreprise doit piloter et assister les agents pendant leur travail pour une meilleure et plus rapide gestion des incidents et pour assurer leur sécurité et le bon déroulement du travail.

Le système mobile doit être personnalisable pour s'adapter aux besoins des usagers et éviter de les submerger d'informations inutiles ou sans intérêt. Il doit garantir en plus la sécurité des informations personnelles ainsi que la sécurité des informations internes de l'entreprise. Le système mobile doit contenir un service d'assistance permettant de mieux aider les usagers (voyageurs, conducteurs, etc) ou même les personnes ayant un besoin spécifique d'assistance (mal-voyance, handicap mental, bas âge).

Les fonctions de l'entreprise Ces fonctions sont inspirées du modèle ACTIF [1] qui présente des services fonctionnels pour l'entreprise des transports.

Un "service d'exploitation" interne à l'entreprise s'occupe de la gestion du réseau de transport pour assurer l'organisation des services en planifiant le réseau des transports publics et en diffusant l'offre de transport (cartes+horaires) aux acteurs intéressés. De plus, le service d'exploitation s'occupe de réguler et suivre les transports publics ainsi que d'organiser les services spécialisés adressés aux usagers abonnés ou utilisés comme solutions alternatives en cas d'urgence. Les données de transport et de planification doivent être traitées, stockées et diffusées selon les destinataires.

Le "service ressources matérielles" s'occupe de la gestion des ressources matérielles notamment de leurs disponibilités, leurs états, et les incidents qui les concernent. La gestion des ressources matérielles (véhicules) est supportée par un "service de maintenance" localisé sur un autre site. Un autre "service des ressources humaines" gère les ressources humaines.

Une “unité de médiation” s’occupe de la gestion de tous les évènements qui peuvent affecter les transports publics. Ces évènements concernent :

- les conditions environnementales (pollution sonore, pollution atmosphérique, données météo, etc.) permettant au service de gestion des déplacements de conclure à l’état des déplacements.
- les projets de travaux concernant les infrastructures de transport et le réseau routier. Les autorités des communes peuvent déclarer des évènements qui concernent les infrastructures utilisés par les transports publics tels que les données des chantiers.
- les changements des données d’exploitation et de planification déclarés par le service d’exploitation.

Ces évènements doivent être vérifiés et analysés pour détecter d’éventuels incidents qui intéressent le “service de gestion des urgences et de la sécurité” et le “service de déplacement”.

Le “service de déplacement” est responsable de la diffusion de l’information de transport aux acteurs externes (sites web, applications mobiles, afficheurs) en analysant les données théoriques de transports produits par le “service d’exploitation” et les conditions de déplacements déduites des incidents. Ce service permet l’estimation des données réelles de transport.

Le “service de gestion des urgences et de la sécurité” est mis en place pour analyser les incidents provenant de l’unité de médiation, du service de maintenance, des utilisateurs et des standards d’appel d’urgence. Ce service doit évaluer et compléter les messages incidents qui doivent être envoyés vers les fournisseurs externes, la gestion des déplacements et vers un analyste de terrain qui analyse ces données. Pour gérer les urgences il faut prendre en compte les incidents, les données de déplacement, les informations reçues des fournisseurs externes. Ensuite, le chef de service doit choisir une stratégie de gestion des urgences parmi les stratégies définies par : l’autorité de transport, l’autorité d’application de la réglementation ou l’exploitant du réseau. Il faut prendre en compte pour chaque choix l’efficacité de la stratégie en analysant les données historisées. Une stratégie sert à mobiliser plusieurs partenaires d’urgence (pompiers, police), l’équipe d’intervention, l’équipe de maintenance. L’application d’une stratégie peut modifier certaines données telles que les conditions de déplacement ou de planification. Dans des situations d’urgence critiques rencontrant les voyageurs (panne d’un véhicule, accident, incident qualifié de grave), il faut envoyer des véhicules spécialisés.

Pour mobiliser les différents acteurs internes suite à un incident, les incidents sont classés par type et par niveau (de 1 à 4).

Description du produit : Système d’Information Mobile (SIM) Les usagers du SIM peuvent être les voyageurs, les conducteurs, les agents en mission et les contrôleurs. Ces usagers peuvent être intéressés aux incidents affectant le réseau et l’état des transports selon leurs rôles, leurs localisations ou leurs données personnelles.

Les services fournis par le SIM pour les voyageurs diffèrent des services fournis aux agents de l’entreprise :

1. Le voyageur possède une application représentant son SIM sur un dispositif mobile (PDA, téléphone portable, smart phone, etc.) pour :
   - effectuer des opérations classiques comme la consultation des horaires, la consultation des cartes du réseau ou le calcul d’itinéraires,
   - consulter des horaires en temps réel,
   - recevoir des alertes adaptées à ses besoins ou à sa situation géographique,
   - exprimer des préférences sur les moyens et les lignes de transport ou même exprimer des habitudes de transport ou d’éventuelles situations d’empêchement permanentes ou

occasionnelles (handicap, femme avec poussette, etc.),
- être assisté pendant un trajet (signaler quand le tram arrive ou quand il faut monter ou descendre),
- acheter ou recharger des titres de transport,
- avoir des liens vers des informations complémentaires (météo, trafic, loisirs, autres services de transport : SNCF, etc.),
- déclarer des évènements et des incidents rencontrés par le voyageur.

Les usagers valides peuvent choisir d'être assistés en permanence si l'itinéraire est nouveau. Une personne souffrant d'handicap mental ou une personne malvoyante doit être toujours suivie. Le contact d'un service partenaire d'urgence est parfois souhaité en cas de problème rencontrant le voyageur handicapé. De plus, la communication du voyageur avec le dispositif doit être adaptée à son handicap (malvoyant, malentendant).

L'assistance peut être déclenchée par l'utilisateur ou suite à une souscription particulière. Elle consiste à suivre le voyageur durant son déplacement et l'informer des étapes du trajet. Le SIM doit être assez autonome pour pouvoir calculer des solutions en cas de perturbation et il est possible de se procurer toute information de transport en communiquant avec le SI central.

2. Ce SI peut être destiné à des agents d'intervention qui interviennent sur le terrain pour réparer des pannes ou pour dégager des voies. Le SI consiste en une application sur dispositif mobile qui permet de :
   - communiquer pendant le travail pour envoyer et recevoir des alertes,
   - consulter le processus d'intervention ou être assisté (ceci dépend de l'expertise des agents et de l'ampleur de l'incident),
   - envoyer des rapports courts en fin d'activité.
3. Une autre application embarquée sur les véhicules de transport permet aux conducteurs de :
   - être suivi et/ou guidé (selon l'état du trafic) dans leurs parcours pour desservir les arrêts et les gares,
   - échanger des alertes (incidents, panne, etc).

Équipements Les arrêts, les véhicules de transport, les gares sont dotés d'afficheurs communiquant des informations de transport aux voyageurs. Plusieurs arrêts sont équipés par des afficheurs temps réel permettant d'informer le voyageur du temps d'attente réel d'une ligne (par opposition au temps théorique de passage qui est défini au préalable dans les fiches horaires consultables à l'arrêt ou sur Internet). De plus, des systèmes Bluetooth sont mis en place pour permettre aux arrêts et aux véhicules de communiquer avec les dispositifs mobiles à proximité.

Le SI central localise les bus et les véhicules spécialisés par des systèmes GPS, alors que les positions des trams sont détectées grâce à des capteurs RFID sur les rails. Le SI central contient toutes les informations théoriques telles que les cartes géographiques, les cartes du réseau, les fiches horaires théoriques...

4.3 Étude préalable

L'étude préalable est la première phase du processus E-CARe qui permet la capture et l'organisation générale des besoins du projet dont l'aboutissement dépend de l'efficience et de la précision lors de la réalisation des étapes de cette phase. L'objectif de cette phase est de comprendre l'organisation et les

métiers de l'entreprise et y placer les besoins exprimés dans le cahier des charges. L'effort et le temps de développement dédiés à cette phase sont plus importants dans la première itération du processus. Néanmoins cette phase doit être prise en compte dans les autres itérations pour suivre les besoins évolutifs des clients. L'étude préalable est constituée d'un ensemble d'étapes respectant le processus de la figure 4.2. Les acteurs intervenant dans cette phase sont de spécialités différentes (maîtrise d'ouvrage et expertise métier, technique et du domaine), ce qui permet de mieux appréhender les différents aspects du projet et mieux identifier les besoins mélangés.

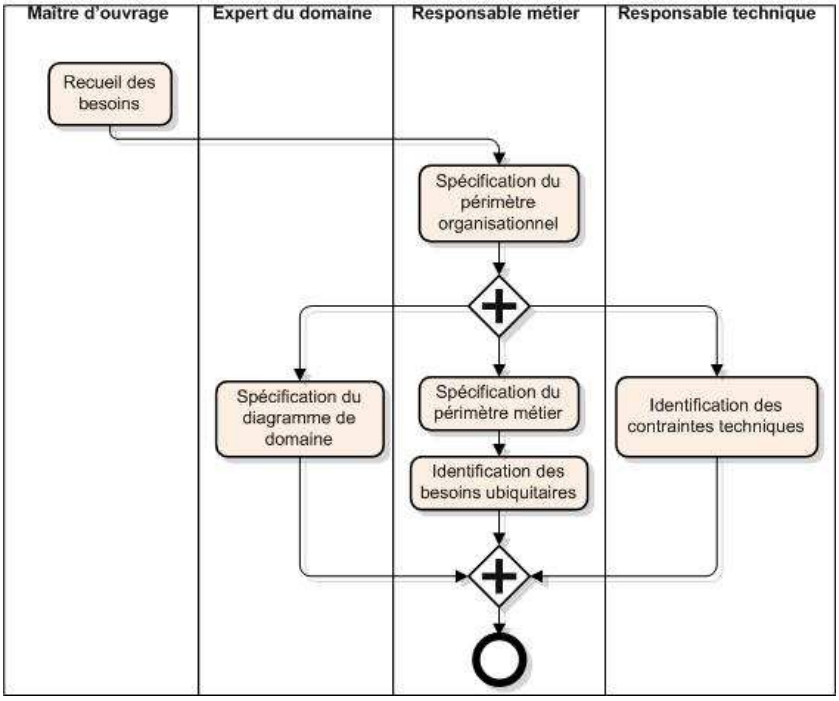


FIGURE 4.2 – Déroulement de la phase d'étude préalable

Cette phase prend en entrée le cahier des charges général et produit en sortie un diagramme de domaine, des contraintes techniques et un diagramme de cas d'utilisation enrichi avec les besoins ubiquitaires. Chaque étape de la phase est décrite ci-dessous et est illustrée avec des exemples du cas d'étude.

4.3.1 Recueil des besoins

Cette étape est réalisée par le maître d'ouvrage qui identifie et organise les exigences des utilisateurs pour rédiger un cahier des charges détaillé. Ce cahier des charges est un document textuel qui capture les besoins fonctionnels, techniques et ubiquitaires de deux types d'acteurs :

- le client : est le propriétaire de l'application ou du système à développer. Il s'intéresse généralement à obtenir une application performante respectant un coût optimal et des délais prédéfinis. Le texte du cahier des charges doit prendre en compte les attentes des clients en particulier des fonctionnalités ubiquitaires à réaliser et des technologies ubiquitaires que les clients souhaitent utiliser.
- l'usager : est le consommateur de l'application. Le système doit respecter les attentes et les besoins des usagers pour leur fournir des fonctionnalités avec une valeur ajoutée concurrentielle. Des mesures comme l'utilisabilité et la satisfaction des usagers sont essentielles pour estimer

les performances du système. Il importe de souligner que les fonctionnalités ubiquitaires doivent faire le sujet d'investigation auprès des usagers pour identifier leurs attentes et leur méfiance par rapport à ces fonctionnalités. Ceci permet de conclure à des besoins de paramétrer ou d'annuler certaines fonctionnalités ubiquitaires ou même d'ajouter des besoins non-fonctionnels au projet comme la sécurité et les délais des tâches. L'étude de ces besoins dans le cadre d'un SI ubiquitaire est pris en charge par le lot 3 du projet DéSIT en cours de réalisation au sein du laboratoire de psychologie LIP / PC2S (voir l'introduction de ce manuscrit).

Dans le cas du SI des transports en commun, le cahier des charges a été décrit dans la section 4.2.

Dans le cahier des charges d'une application ubiquitaire, les aspects contextuels et mobiles sont mélangés avec les spécifications fonctionnelles et techniques du système et elle ne sont pas visibles immédiatement. C'est dans les étapes qui suivent que les besoins ubiquitaires doivent être identifiés et analysés.

4.3.2 Spécification du périmètre organisationnel

Cette étape est effectuée par le responsable métier pour déterminer les structures de l'organisation qui entrent en jeux dans le cadre du travail de développement. La définition des organisations et des sous-organisations qui les composent permet de capturer les différentes unités prenant en charge les fonctionnalités du système ainsi que les acteurs et la hiérarchie des acteurs qui interviennent dans ces fonctions.
En appliquant cette étape à notre étude de cas, nous identifions l'organisation suivante de l'entreprise de transport :

- Un service d'exploitation : responsable de la gestion du réseau des transports publics et contenant quatre unités :
  - une unité de planification des transports publics,
  - une unité de gestion des ressources humaines,
  - une unité de maintenance,
  - une unité de gestion de l'infrastructure.
- Un service de gestion des urgences et de la sécurité : responsable de la détection et de la résolution des problèmes et des évènements internes ou externes composé de deux unités :
  - une unité de médiation,
  - une unité de gestion des incidents.
- Un service d'information : responsable du traitement et de la distribution des informations de transport aux opérateurs et acteurs internes et externes. Ce service contient quatre unités :
  - une unité de supervision des transports,
  - une unité de supervision du personnel,
  - une unité de production de l'information,
  - une unité de distribution de l'information.

4.3.3 Spécification du diagramme de domaine

Cette étape est réalisée par l'expert du domaine qui spécifie dans un diagramme de classes ou une ontologie les concepts du domaine et les liens entre eux pour faciliter la compréhension du domaine aux informaticiens. Cette compréhension est nécessaire aussi bien pour les experts métier que les experts techniques. Pour certains domaines, le résultat de cette étape paraît évident, mais pour d'autres domaines moins communs, la définition des concepts et de leurs liens n'est pas toujours facile. De plus la présentation d'un support représentant les termes manipulés constitue une référence

à utiliser par les informaticiens pour concevoir des systèmes respectant la nomenclature utilisée dans l'entreprise et les structures réelles. Concernant l'entreprise de transport, la figure 4.3 permet de capturer les termes clés représentant les modes de transport qu'elle gère.

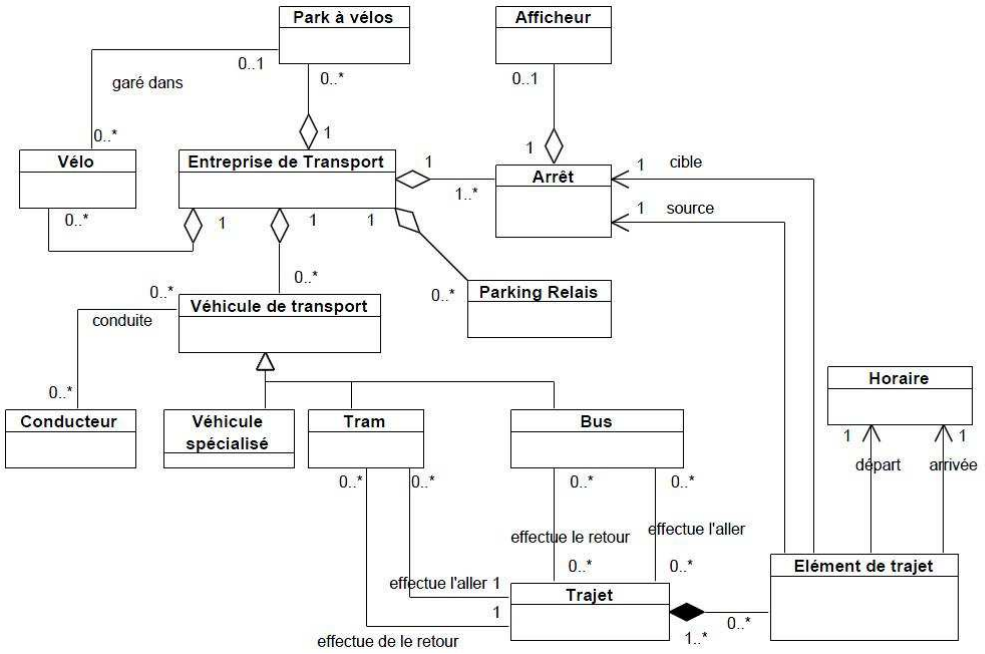


FIGURE 4.3 – Diagramme de domaine pour l'entreprise des transports publics

4.3.4 Spécification du périmètre métier

Cette étape est réalisée par le responsable métier qui procède à une identification globale des différents PM décrits dans les besoins fonctionnels du système et les organise selon leurs unités organisationnelles. Il est nécessaire de procéder de plus à un repérage préliminaire de la vue intentionnelle en définissant les objectifs de chaque PM et sa place dans les stratégies de l'entreprise. Les objectifs peuvent être fonctionnels ou techniques.
Concernant notre étude de cas nous représentons les résultats de cette étape sous la forme d'un tableau regroupant pour chaque PM ses unités organisationnelles, ses objectifs et les stratégies de l'entreprise qu'il supporte. Le tableau 4.3 représente un extrait de 4 PM des PM de l'entreprise de transport : deux concernent la rénovation du SI central ("Gestion des évènements" et "Gestion des incidents") et deux concernent le développement d'applications mobiles ("Assistance au voyageur" et "Assistance à l'intervention").

Les différents PM doivent être décrits textuellement tout en identifiant les différents acteurs participant dans le processus. Ainsi, il est recommandé de rédiger un scénario primitif contenant tous les détails fonctionnels de chaque processus. Ces scénarios doivent être proches des exigences des clients et indépendants de tout effort d'interprétation et de structuration. Cette description décrit aussi bien des besoins métier que des besoins IHM. A titre d'exemple, le scénario primitif du processus "Assistance au voyageur" est décrit dans le tableau 4.4.

TABLE 4.3 – Processus métier globaux et leurs objectifs : une vue intentionnelle de haut niveau

Processus Métier	Organisations concernés	Objectifs	Stratégies
Gestion des évènements	Service de gestion des urgences et de la sécurité : Unité de médiation	détecter rapidement les incidents sur le réseau, notifier rapidement tous les acteurs concernés	améliorer le temps de réponse aux incidents et une meilleure information en cas de perturbation
Gestion des incidents	Service de gestion des urgences et de la sécurité : unité de gestion des incidents	répondre rapidement et efficacement à un évènement et notifier rapidement tous les acteurs concernés	améliorer le temps de réponse aux incidents et une meilleure information en cas de perturbation
Assistance au voyageur	Service d'information : unité de distribution de l'information et SIM (SI Mobile)	fournir une information fraîche et utile, faciliter l'accès aux transports pour les personnes invalides et fournir des solutions personnalisées en cas de perturbations	faciliter l'usage des transports publics
Assistance à l'intervention	Service d'information : unité de supervision du personnel et unité de distribution de l'information	optimiser le temps d'intervention, faciliter les interventions et éviter l'aggravation des incidents	assurer une résolution rapide et efficace des incidents

TABLE 4.4 – Scénario du PM "Assistance au voyageur"

Plusieurs évènements peuvent déclencher le processus d'assistance au voyageur :
- suite à la demande de l'usager,
- suite à une alerte affectant l'itinéraire courant,
- suite à la détection d'une déviation d'un sujet invalide,

Le système mobile doit chercher l'itinéraire à suivre, identifier la position du voyageur et appliquer cet itinéraire étape par étape en partant de la localisation courante qui doit être surveillée tout au long du processus de suivi en détectant les véhicules et les arrêts dans lesquels se trouve le voyageur. L'arrivée à destination indique la fin du processus d'assistance. Pour calculer un itinéraire, le SIM peut demander des compléments d'information sur les données de transport si nécessaire. Il est de plus possible de consulter les détails des alertes ou d'avoir accès à des services externes pour choisir des modes de transport alternatifs.

Exceptions
- La réception d'une alerte concernant l'itinéraire en cours doit déclencher la recherche d'un chemin alternatif et l'assistance de l'usager au cours de l'itinéraire.
- Au cours d'un itinéraire l'usager peut notifier un évènement sur le réseau de transport qui doit être pris en compte et envoyé à l'unité de médiation et une solution alternative doit être calculée si nécessaire.
- Tout incident concernant le voyageur et détecté par le SIM doit alerter des parties externes d'urgence.

4.3.5 Identification des besoins ubiquitaires

En collaboration avec le responsable métier, l'analyste-concepteur de contexte étudie les besoins métier et le cahier des charges pour extraire et définir des fonctionnalités ubiquitaires qu'il attache aux PM et aux liens entre PM et acteurs. Certains besoins ubiquitaires sont définis globalement dans le cahier des charges tels que "Personnaliser les informations fournies" ou "Adapter l'affichage". D'autres besoins sont spécifiques à des PM particuliers, tels que "Adapter l'assistance au handicap"

ou “Le guidage à l'intervention doit être personnalisé selon l'expertise de l'équipe”. Le rôle d'analyste-concepteur de contexte est d'analyser ces besoins explicitement exprimés et de les placer dans le périmètre métier. Il peut d'autre part définir de nouveaux besoins ubiquitaires qu'il juge (grâce à ses compétences) utiles et réalisables. Le résultat de cette étape est un diagramme de cas d'utilisation contenant les PM globaux, les liens entre eux et avec les acteurs et les besoins ubiquitaires qui leurs sont associés (voir la figure 4.4). Les besoins ubiquitaires apparaissent comme des annotations attachées aux composants métier ou aux liens avec les acteurs.

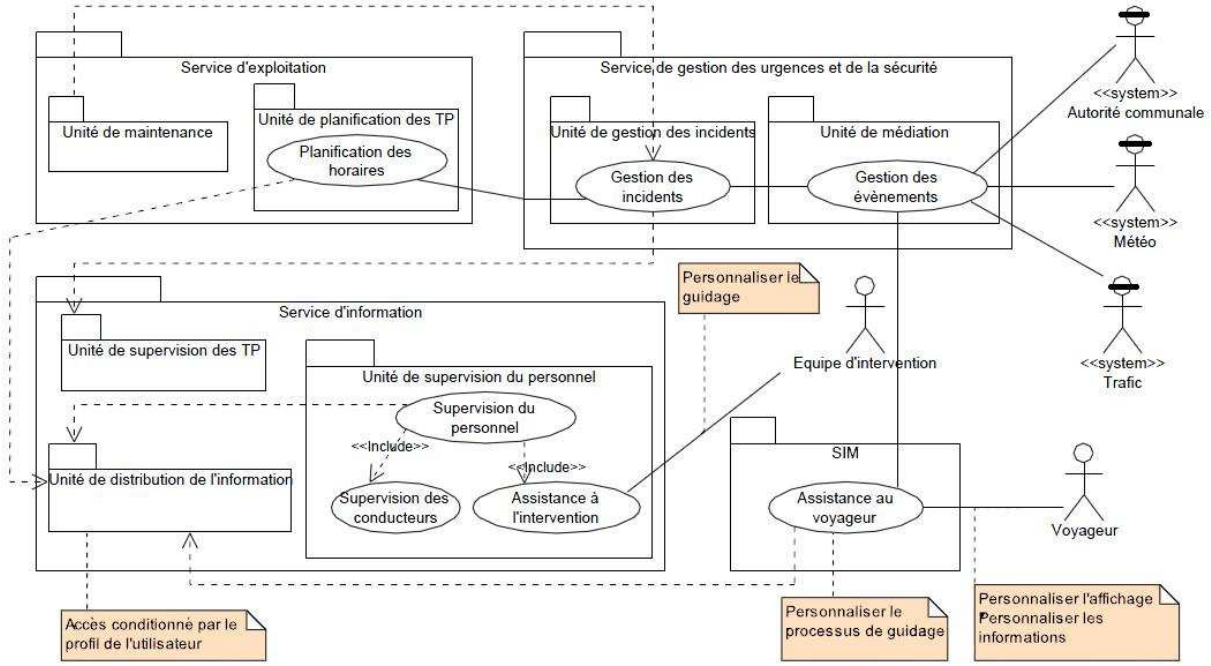


FIGURE 4.4 – Diagramme de cas d'utilisation correspondant au périmètre métier annoté par les besoins ubiquitaires

4.3.6 Identification des contraintes techniques

Il s'agit dans cette étape d'identifier les différentes contraintes techniques spécifiées par le client. En effet, il est possible de capturer ces besoins à partir des spécifications des cas d'utilisation ou du système informatique existant. Cette étape permet de retenir les contraintes techniques et les éléments matériels et logiciels nécessaires pour la spécification des besoins techniques.

Concernant notre étude de cas les besoins techniques sont résumés dans les points suivants :

- maximiser la sécurité des données de transport internes,
- maximiser la sécurité des données personnelles des usagers,
- utiliser des dispositifs mobiles, des dispositifs embarqués sur les véhicules et des afficheurs sur les infrastructures,
- localiser par GPS des voyageurs, des bus et des véhicules spécialisés,
- localiser par capteurs RFID des trams,
- communiquer par Bluetooth entre les dispositifs mobiles, les véhicules et les arrêts.

4.4 Spécification intentionnelle des besoins

La phase précédente d'étude préalable permet d'étudier le cahier des charges et d'extraire et d'organiser les besoins des clients et des utilisateurs sous forme de PM globaux, de besoins ubiquitaires et de contraintes techniques. Le rôle de la phase intentionnelle est d'analyser ces différents besoins par des experts des trois domaines (fonctionnel, ubiquitaire et technique) pour détailler ces besoins et les formaliser sous forme de PM détaillés et de règles. Les modèles ainsi obtenus sont des spécifications intentionnelles (dirigées par les objectifs) et respectent les points de vues fonctionnels, ubiquitaires et techniques. Par conséquent, la séparation des dimensions garantit la cohérence entre les modèles qui seront produits dans chaque branche. Ainsi, cette phase joue un rôle primordial dans notre démarche et en constitue l'originalité.

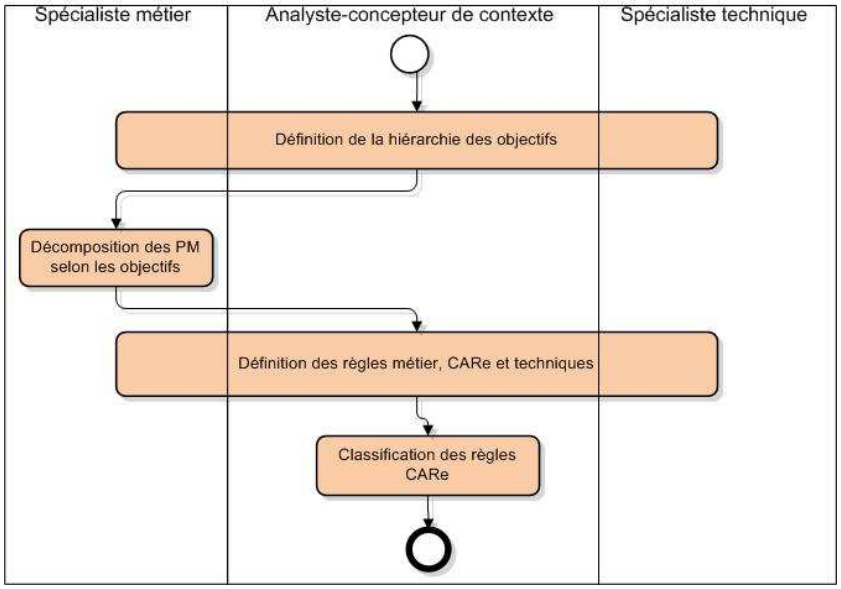


FIGURE 4.5 – Processus de la phase spécification intentionnelle des besoins

Cette phase est composée de quatre étapes comme le montre la figure 4.5. Ces étapes se basent sur la méthode intentionnelle introduite dans le chapitre précédent qui consiste à utiliser les objectifs métier et les besoins ubiquitaires pour dériver une hiérarchie détaillée des objectifs permettant d'identifier des besoins de plus fine granularité sous forme de règles. La méthode KAOS dirigée par les objectifs est utilisée pour le raffinement et la validation des objectifs, ce qui permet de décomposer les PM globaux. Les différentes étapes de cette démarche sont détaillées ci-dessous.

4.4.1 Définition de la hiérarchie des objectifs

Au cours de l'étape de définition du périmètre métier dans la phase d'étude préalable le responsable métier définit les PM globaux et leurs objectifs exprimés dans le cahier des charges. Le but de cette étape est de revoir ces objectifs et les raffiner pour obtenir des objectifs plus ciblés et hiérarchisés selon le modèle d'objectifs présenté dans le chapitre 3 (section 3.4). La production de la hiérarchie des objectifs selon la méthode KAOS se base sur l'identification des sous-objectifs et la gestion des conflits qui peuvent avoir lieu entre certains objectifs. La méthode KAOS fournit un ensemble de tactiques et de conseils qui facilitent la décomposition des objectifs en des sous-objectifs qui sont opérationnels [Dardenne *et al.*, 1993]. Nous retenons de ces tactiques les suivantes :

- il faut procéder à une spécialisation ou une instanciation des descriptions des objectifs génériques,
- la décomposition des objectifs s'arrête quand ces sous-objectifs deviennent opérationnels,
- la décomposition d'un objectif doit produire des sous-objectifs qui font intervenir moins d'acteurs pour les réaliser,
- la recherche des sous-objectifs alternatifs permet de minimiser les coûts,
- la décomposition doit minimiser le nombre de conflits entre objectifs.

La réalisation de cette hiérarchie doit prendre en compte les objectifs ubiquitaires et non fonctionnels et les attacher aux objectifs fonctionnels par une décomposition. Nous utilisons les stéréotypes "Objectif fonctionnel", "Objectif CARe" et "Objectif non fonctionnel" pour différencier les objectifs. Si un objectif CARe ou un objectif non fonctionnel est associé avec un objectif fonctionnel de plus haut niveau, alors il doit être supporté par tous ses sous-objectifs.

L'application de cette étape sur le cas d'étude des transports en commun utilise les objectifs définis dans la phase précédente pour l'ensemble des PM. Chaque objectif peut produire une hiérarchie indépendante de sous-objectifs. Nous étudions le cas du PM d'assistance au voyageur dont la hiérarchie obtenue pour les objectifs est représentée par la figure 4.6. Les objectifs CARe obtenus dans cette étape sont associés avec le processus global de haut niveau puisque l'adaptation de l'affichage ou la personnalisation des informations complètent tous les objectifs fonctionnels de plus bas niveau. L'objectif CARe "Considérer les préférences de transport" est relié à l'objectif fonctionnel "Calculer itinéraire".

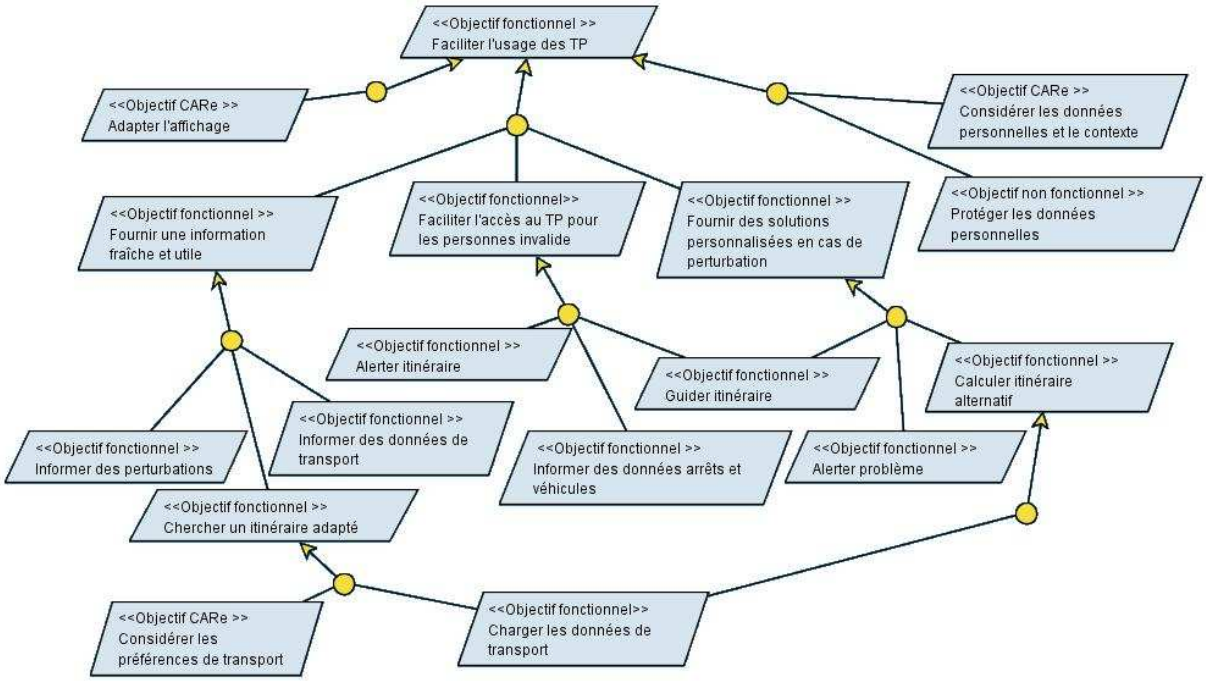


FIGURE 4.6 – Hiérarchie des objectifs du PM "Assistance au voyageur"

4.4.2 Décomposition des PM selon les objectifs

La hiérarchie des objectifs fonctionnels obtenue précédemment est utilisée dans cette étape pour produire une structure de PM qui permet de supporter ces objectifs. Un PM supporte un ou plusieurs

objectifs fonctionnels et un objectif fonctionnel peut être supporté par un ou plusieurs processus. Le processus global qui correspond à l'objectif de plus haut niveau est ainsi facilement décomposé en un ensemble de PM détaillés. Le responsable métier prend en charge cette étape et il est responsable de certains choix comme la création de différentes versions de PM pour réaliser les mêmes objectifs ou la création d'un seul PM pour un ensemble d'objectifs.

La considération du PM "Assistance au voyageur" passe par la hiérarchie d'objectifs définie dans la figure 4.6 et permet de produire un ensemble de PM supportant des objectifs fonctionnels comme le montre le tableau 4.5.

TABLE 4.5 – Décomposition du PM "Assistance au voyageur" selon les objectifs fonctionnels

Processus métier	**Objectifs fonctionnels**
Calculer un itinéraire	– Chercher itinéraire adapté
Notifier un évènement	– Informer des perturbations – Informer des données de transport – Alerter itinéraire
Chercher des informations	– Informer des données de transport – Charger les données de transport
Assister un itinéraire	– Informer des données des véhicules et des arrêts – Guider un itinéraire
Notifier une perturbation	– Informer des perturbations – Informer des données de transport
Rappeler un déplacement	– Alerter un itinéraire
Émettre une alerte	– Alerter un problème
Résoudre une perturbation	– Informer des perturbations – Calculer itinéraire alternatif – Charger les données de transport

Concernant les objectifs CARe ou les objectifs non fonctionnels, ils suivent leurs objectifs fonctionnels et ainsi les PM qui les supportent. Dans certains cas, il est nécessaire de définir des processus qui ne mettent pas en évidence une fonctionnalité du système c'est-à-dire ne supportent pas des objectifs fonctionnels mais qui supportent des objectifs CARe ou non fonctionnels. Ces processus sont dits des processus de support ; ce sont des processus transverses et génériques communs à plusieurs PM et indispensables à leur fonctionnement [Ben Cheikh, 2008]. Par exemple, l'usage de données de profil nécessite un processus "Gestion de profil" et le besoin de sécuriser certaines données nécessite un processus pour l'authentification.

Les objectifs CARe et non fonctionnels attachés aux objectifs fonctionnels de chaque PM (ou les objectifs fonctionnels de plus haut niveau) doivent être associés avec le PM concerné comme le

montre le tableau 4.6. Dans ce tableau, les objectifs CARe "Adapter l'affichage" et "Considérer les données personnelles et le contexte" et l'objectif non fonctionnel "Protéger les données personnelles" sont associés avec tous les PM puisqu'ils sont attachés à l'objectif fonctionnel de plus haut niveau "Faciliter l'usage des transports publics".

TABLE 4.6 – Association des objectifs CARe et non fonctionnels aux PM de l'assistance au voyageur

Processus métier	Objectifs CARe	Objectifs non fonctionnels
Calculer un itinéraire	– Considérer les préférences de transports – Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Notifier un évènement	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Chercher des informations	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Assister un itinéraire	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Notifier une perturbation	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Rappeler un déplacement	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Émettre une alerte	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles
Résoudre une perturbation	– Adapter l'affichage – Considérer les données personnelles et le contexte	– Protéger les données personnelles

4.4.3 Définition des règles métier, CARe et techniques

Les différents objectifs fonctionnels, CARe et non fonctionnels sont à ce stade généraux et globaux. Pour mieux les mettre en œuvre dans ce système nous proposons aux différents spécialistes métier,

de contexte et technique de détailler ces objectifs en les transformant en des règles métier, CARe et techniques et les relier aux PM auxquels elles s'appliquent. Il est recommandé de commencer avec les objectifs de plus bas niveau et monter petit-à-petit dans la hiérarchie des objectifs. Chaque objectif est utilisé séparément et peut générer une ou plusieurs règles. Cette méthode de génération des règles est illustrée avec le métamodèle intentionnel de la figure 3.9. Elle permet de donner un aperçu sur les différents besoins de chaque branche et les prendre en compte par les différents spécialistes pour éviter une incohérence dans les modèles. Ainsi la séparation des branches est plus sûre.

L'application de cette étape sur notre étude de cas fournit le tableau 4.7 contenant pour chaque objectif un ensemble de règles.

TABLE 4.7 – Un ensemble de règles dérivées de la hiérarchie des objectifs pour le processus "Assistance au voyageur"

Objectifs	PM	Règles	Types de règles
Protéger les données personnelles	Assister un itinéraire	Les données personnelles ne doivent pas être échangées avec les acteurs externes	Non fonctionnelle
		le voyageur peut à tout moment annuler l'assistance	Métier
Considérer les données personnelles et le contexte	Calculer un itinéraire	L'empêchement et la localisation doivent être considérés	CARe
	Notifier une perturbation	Les habitudes de transport et la localisation doivent être considérées avant de notifier	CARe
	Assister un itinéraire	Les perturbations environnantes doivent être considérées lors du déplacement	CARe
		la modification des données personnelles doit engendrer un recalcul de l'itinéraire	Métier
Adapter l'affichage	Assister un itinéraire	L'affichage doit considérer l'handicap du voyageur et ses préférences d'affichage	CARe
Guider itinéraire	Assister un itinéraire	L'assistance est obligatoire pour les personnes invalides	CARe
Informer des perturbations	Notifier une perturbation	Les voyageurs doivent être informés de la cause des perturbations rapidement	Métier
	Assister un itinéraire	Le voyageur peut notifier des évènements qu'il rencontre	Métier

4.4.4 Classification des règles CARe

Les différentes règles CARe doivent être classifiées selon le type d'adaptation (voir le métamodèle intentionnel de la figure 3.9) : adaptation de la présentation, adaptation des informations, adaptation métier et adaptation du contrôle d'accès. La classification des règles dans cette étape permet aux spécialistes métier et technique d'avoir une idée sur les besoins d'adaptation qui les concernent. Ces besoins CARe seront analysés et formalisés par le concepteur du contexte dans les phases suivantes.

Le tableau 4.8 présente la classification des règles CARe identifiées dans le tableau 4.7. Ces règles contiennent des règles d'adaptation de l'information qui consistent à prendre en compte le contexte pour informer le voyageur ou calculer son itinéraire. Nous trouvons également des règles d'adaptation métier qui permettent d'imposer des fonctionnalités supplémentaires aux PM en fonction de certaines conditions comme la considération des perturbations environnantes dans le processus de guidage ou l'obligation de guidage pour les personnes invalides. L'adaptation de la présentation permet de changer

le mode de livraison de l'information en fonction du handicap : une personne malvoyante reçoit une information sonore alors qu'une personne malentendante reçoit une information vibrante et visuelle.

TABLE 4.8 – Classification des règles CARe du PM "Assistance au voyageur"

Règle CARe	Classe de règle
L'empêchement et la localisation doivent être considérés pour calculer un itinéraire	Adaptation de l'information
Les habitudes de transport et la localisation doivent être considérées avant de notifier	Adaptation de l'information
Les perturbations environnantes doivent être considérées lors du déplacement	Adaptation métier
L'affichage doit considérer le handicap du voyageur et ses préférences d'affichage	Adaptation de la présentation
L'assistance est obligatoire pour les personnes invalides	Adaptation métier

4.5 Spécification structurelle du contexte

Cette phase constitue la première phase purement ubiquitaire qui est prise en charge par l'analyste-concepteur de contexte. Elle a pour objectif la formalisation du contexte par l'analyse, de façon graduelle, des différents besoins ubiquitaires identifiés dans la phase intentionnelle et la définition du modèle de contexte sur lequel se base toute fonctionnalité ubiquitaire. Elle est constituée d'un ensemble d'étapes (voir figure 4.7) qui permettent de détailler les règles CARe exprimées jusque là de façon textuelle en identifiant des conditions plus ciblées et les réactions correspondantes. Ces conditions permettent de dégager les données de contexte à considérer dans le projet et de les modéliser en instanciant le métamodèle de contexte de la figure 4.8 (décrit dans le chapitre 3 à la section 3.3).

4.5.1 Raffinement des règles par conditions

Les règles CARe définies dans la phase précédente représentent l'entrée principale de cette étape : les règles textuelles sont transformées dans la forme "si Condition alors Action". Cette transformation est nécessaire pour permettre de séparer les conditions des réactions. La partie condition d'une règle CARe permet d'évaluer des conditions sur l'environnement, le contexte et le profil des usagers. Elle nous donne ainsi une idée sur ce qui doit être retenu du contexte. Au cours de cette étape une règle métier est divisée en plusieurs situations d'adaptation obtenues après une analyse détaillée des attentes des usagers pour la réalisation des objectifs de contextualisation exprimés dans le métier à savoir la facilité d'usage, la personnalisation, l'ergonomie, l'intérêt, l'utilité et la cohérence.

Les règles définies jusqu'à ce stade peuvent contenir des termes abstraits et généraux qui regroupent des détails et des cas particuliers tels que le terme perturbation qui peut signifier un retard, une avance ou une annulation. Une règle peut contenir dans certains cas des réactions différentes à différentes conditions comme le cas de la règle "L'assistance est obligatoire pour les personnes invalides" qui doit s'assurer du suivi de l'itinéraire et le cas échéant informer un tuteur et déclencher un guidage automatique au moment de l'itinéraire.

Identifier ces règles détaillées favorise une meilleure application des besoins d'adaptation aux fonctionnalités du système. Ce travail de raffinement appliqué à notre étude de cas produit le tableau 4.9. Suite à ce raffinement un grand nombre de règles a été défini et certaines règles partagent des conditions similaires. Ces conditions portent en général sur le contexte du voyageur et son environnement, ce qui permet de passer à l'étape suivante de définition des prescriptions contextuelles.

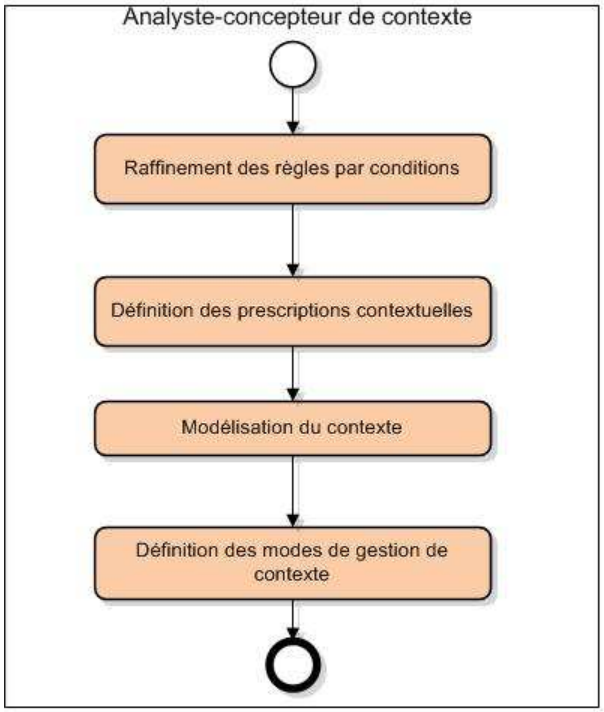


FIGURE 4.7 – Processus de la phase “Spécification structurelle du contexte"

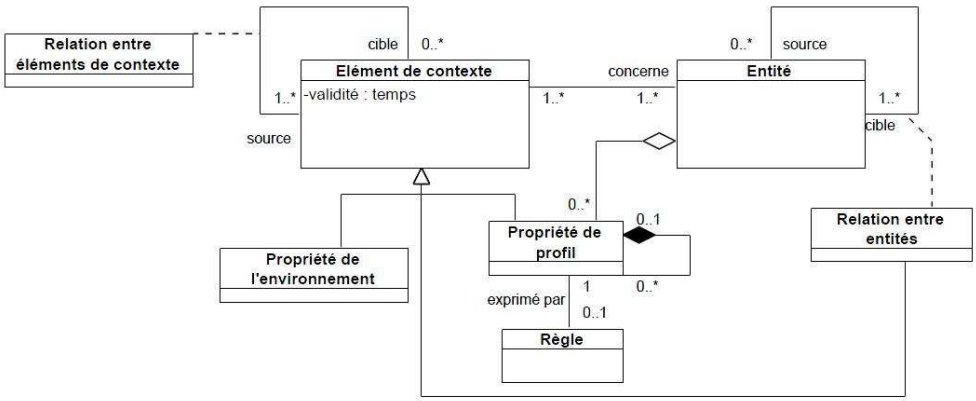


FIGURE 4.8 – Métamodèle structurel de contexte

4.5.2 Définition des prescriptions contextuelles

Les systèmes ubiquitaires sont des systèmes sociaux qui doivent s'approcher au maximum des comportements et des besoins humains. Dans cette étape, l'expert du contexte focalise sur la partie condition de chaque règle CARe pour identifier et spécifier les dépendances contextuelles. Le résultat de cette étape est un ensemble de prescriptions contextuelles exprimées sous forme textuelle et référençant les conditions correspondantes (une même prescription peut concerner plusieurs situations). En effet, chaque condition porte sur un ou plusieurs données du contexte d'où il est possible d'utiliser ces conditions pour identifier les données du contexte qui intéressent l'application. Ces données sont présentées sous forme de recommandations textuelles dites prescriptions contextuelles qui concernent les entités interagissant avec l'application et contiennent les données qui les caractérisent ou qui

TABLE 4.9 – Raffinement par conditions des règles CARe du PM "Assistance au voyageur"

Règle CARe	Condition	Action
L'empêchement et la localisation doivent être considérés pour calculer un itinéraire	La localisation correspond au point de départ	Calculer l'itinéraire
	Une personne empêchée : avec une poussette, des bagages ou béquilles	Calculer un itinéraire avec un arrêt aménagé
	Une personne avec fauteuil roulant	Calculer un itinéraire avec un véhicule et un arrêt aménagé.
	Une personne avec un vélo	Calculer un itinéraire avec un véhicule aménagé
Les habitudes de transport et la localisation doivent être considérées avant de notifier	Un incident concernant un véhicule d'un itinéraire habituel (panne, accident, retard)	Informer le voyageur
	Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule d'un itinéraire habituel	Informer le voyageur
	Une manifestation sur le chemin d'un itinéraire habituel (dans un rayon de 500m)	Informer le voyageur
	Une alerte trafic sur le chemin d'un itinéraire habituel	Informer le voyageur
Les perturbations environnantes doivent être considérées lors du déplacement	Un incident concernant un véhicule de l'itinéraire (panne, accident, retard)	Informer le voyageur
	Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule de l'itinéraire	Informer le voyageur
	Une manifestation sur le chemin de l'itinéraire (dans un rayon de 500m)	Informer le voyageur
	Une alerte trafic sur le chemin de l'itinéraire	Informer le voyageur
L'affichage doit considérer le handicap du voyageur et ses préférences d'affichage	Une personne malvoyante	Envoyer une information sonore
	Une personne malentendante	Envoyer une information vibrante et textuelle
	Des préférences d'affichage	Interagir en respectant ces préférences
L'assistance est obligatoire pour les personnes invalides	Une personne invalide en déplacement	Déclencher l'assistance
	Une personne invalide qui ne suit pas son itinéraire	Alerter son tuteur

caractérisent leur environnement :

- les entités ou les acteurs dont le contexte intéresse l'application,
- les données de contexte caractérisant ces entités,
- les définitions et le vocabulaire de contexte utilisé. Par exemple, le concept "voyageur invalide" doit être défini.

– les relations entre les données de contexte pour dériver de nouvelles données. Par exemple, il est possible de définir certains concepts en fonction de certaines données du contexte tels que le concept “personne âgée” qui concerne une personne dont l’âge est supérieur à 75 ans.

Le tableau 4.10 présente un ensemble de prescriptions contextuelles qui correspondent aux règles CARe précédemment détaillées.

4.5.3 Modélisation du contexte

L’ensemble des prescriptions contextuelles définies par l’expert métier permet de résumer les différentes entités impliquées dans le raisonnement sur le contexte et les éléments de contexte qui les concernent.

Les différentes prescriptions contextuelles concernant toutes les règles métier permettent de visualiser toutes les entités du système et leurs éléments contextuels ce qui facilite la modélisation. Le modèle de contexte généré à ce stade est un modèle conceptuel qui regroupe les entités, leurs profils, leurs relations entre eux et avec l’environnement. Il s’agit d’une instanciation du métamodèle générique de contexte (voir figure 4.8). La figure 4.9 représente le modèle de contexte de l’application "Assistance au voyageur".

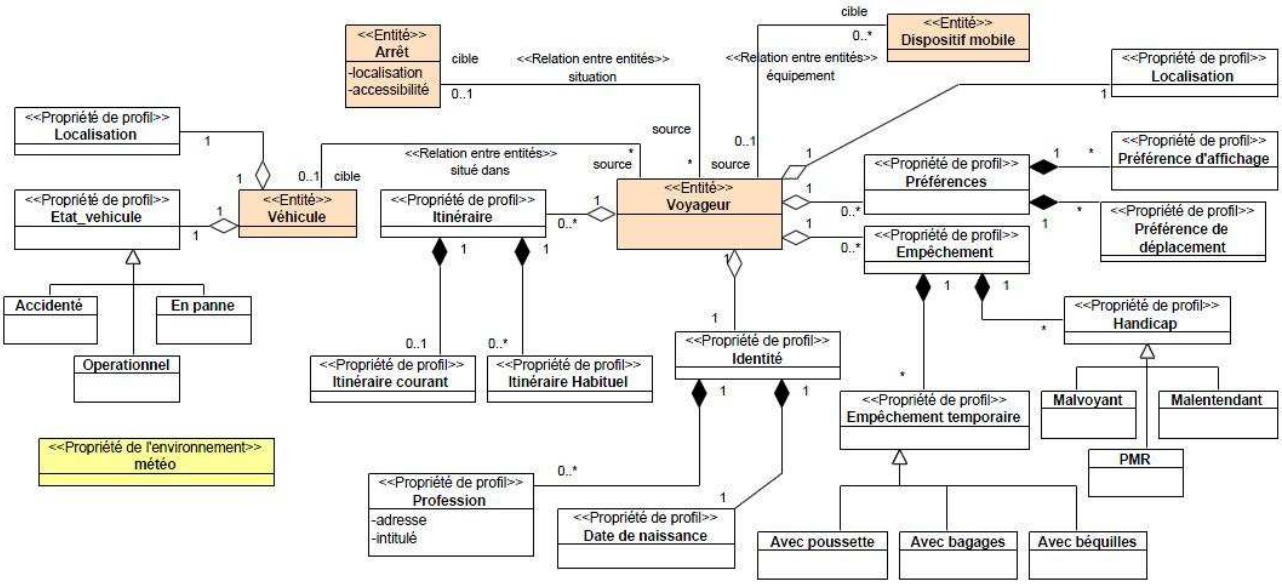


FIGURE 4.9 – Modèle de contexte de l’application “Assistance au voyageur”

4.5.4 Définition des modes de gestion du contexte

Après la définition du modèle de contexte, l’analyste-concepteur de contexte doit définir des modes de gestion et de configuration des données de contexte pour permettre à l’administrateur de l’application ou même à l’utilisateur de paramétrer les données de contexte telles que la définition de la confidentialité de certaines données, les fréquences de mise à jour de certaines données, l’usage d’un historique, le groupement de certaines données, etc. Ces paramètres seront pris en compte dans la conception de la base de données de contexte et sa gestion.

Dans le cas des applications de réseaux sociaux, ces modes de gestion sont primordiaux pour l’utilisateur qui doit avoir le contrôle absolu sur ses données personnelles. Dans le cas des transports

TABLE 4.10 – Prescriptions contextuelles de l'application "Assistance au voyageur"

Conditions	Prescriptions contextuelles
La localisation correspond au point de départ	La localisation du voyageur doit être identifiée par GPS ou dans le réseau de transport
Une personne empêchée : avec une poussette, des bagages ou des béquilles	Un empêchement peut être causé par différentes raisons : poussette, bagages, vélo, fauteuil roulant, béquilles.
Une personne avec fauteuil roulant	
Une personne avec un vélo	
Un incident concernant un véhicule d'un itinéraire habituel (panne, accident, retard)	Un itinéraire habituel est un itinéraire utilisé de façon quotidienne pour se rendre à une adresse comme le travail, la maison, le club, etc.
Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule d'un itinéraire habituel	
Une manifestation sur le chemin d'un itinéraire habituel (dans un rayon de 500m)	
Une alerte trafic sur le chemin d'un itinéraire habituel	
Un incident concernant un véhicule de l'itinéraire (panne, accident, retard)	– Le déplacement est identifié avec l'activation d'un itinéraire dit "Itinéraire courant" – Chaque véhicule de transport doit avoir un état qui peut être en panne, accidenté ou opérationnel – Les données météo doivent être considérées
Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule de l'itinéraire	
Une manifestation sur le chemin de l'itinéraire (dans un rayon de 500m)	
Une alerte trafic sur le chemin de l'itinéraire	
Une personne malvoyante	– Il faut connaître si le voyageur est malentendant ou malvoyant – L'utilisateur peut définir des préférences d'affichage
Une personne malentendante	
Des préférences d'affichage	
Une personne invalide en déplacement	Une personne invalide est une personne : – handicapée mental – personne de moins de 12 ans – âgée de plus de 75 ans
Une personne invalide qui ne suit pas son itinéraire	Chaque personne invalide doit avoir un tuteur avec ses coordonnées

en commun, l'application mobile peut être configurée pour communiquer la localisation et l'identité du voyageur pour pouvoir être perçue par d'autres dispositifs tels que ceux des amis. De même l'utilisateur peut sauvegarder les itinéraires empruntés et qui ne sont pas habituels ou encore certaines adresses.

4.6 Spécification évènementielle du contexte

Cette phase permet de compléter la caractérisation des données contextuelles dynamiques en identifiant la vue évènementielle. Cette vue permet de capturer les sources des informations contextuelles et les évènements qui portent cette information ou qui permettent de déclencher des comportements ubiquitaires comme l'adaptation et la sensibilité au contexte. L'introduction de cette phase est une conséquence de l'adoption d'une approche dirigée par les évènements (voir la section 3.2) où tout est considéré comme un évènement : un évènement peut être une donnée, un comportement ou un état.

Prise en charge par l'analyste-concepteur de contexte, cette phase a pour objectif de fournir un modèle complet et exhaustif de l'ensemble des évènements (appelé aussi nuage d'évènements [Luckham, 2002]) produits et utilisés par l'application et qui permet de participer à une action ubiquitaire quelconque. Une action ubiquitaire peut être une acquisition d'une donnée de contexte, un déclenchement d'un comportement ubiquitaire ou un transfert d'une alerte ou d'un message à un acteur ou un participant dans l'application. Pour produire ce modèle d'évènements, un ensemble d'étapes permet de produire un modèle d'évènements pour chaque action ubiquitaire. Une dernière étape automatique et basée sur des transformations IDM, permet de rassembler ces différents modèles d'évènements et de les transformer en une cartographie unique d'évènements. La figure 4.10 représente le processus de cette phase en fonction des étapes qui la composent.

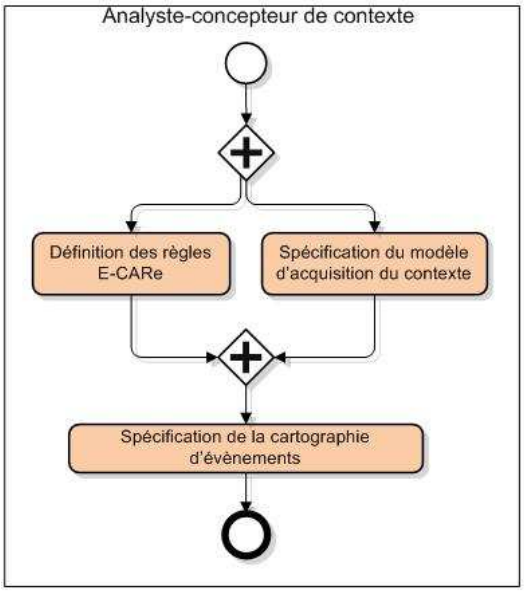


FIGURE 4.10 – Processus de la phase spécification évènementielle du contexte

4.6.1 Définition des règles E-CARe

Les règles utilisées dans notre approche sont de la forme E-CARe : Évènement-Condition Contextuelle- Action d'adaptation (voir chapitre 3 section 3.2.3). L'objectif de cette étape est de

modéliser ces règles avec des diagrammes de classes instanciant le métamodèle de règles E-CARe de la figure 4.11.

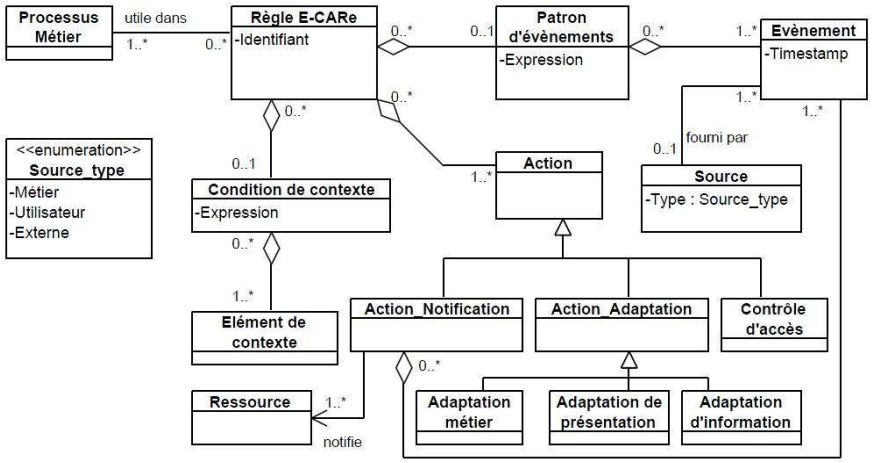


FIGURE 4.11 – Métamodèle de règles E-CARe

Les règles traitées jusqu'à présent avaient la forme Condition/Action, car il est nécessaire dans un premier temps d'ignorer la partie Évènement déclenchant la règle pour ne pas assimiler une condition contextuelle à un évènement ou l'inverse. C'est dans cette phase de spécification évènementielle du contexte qu'il est judicieux d'analyser les règles et les situations contextuelles qui leurs sont associées en vue d'identifier les éléments déclencheurs de la règle.

Pour définir les règles E-CARe, un tableau est utilisé dans un premier temps pour séparer les parties Evènement, Condition contextuelle et Action d'adaptation. Le tableau 4.11 présente les règles E-CARe de l'application d'assistance au voyageur.

La définition de ces règles E-CARe dans un tableau ne couvre pas tous les détails concernant les évènements déclencheurs, les liens entre conditions et contexte ainsi que la nature de l'adaptation. Ainsi, nous complétons avec un métamodèle de règles pour mettre en évidence ces différents détails dans des diagrammes de classes. Les figures 4.12 et 4.13 montrent par exemple la structure de deux règles E-CARe_2 et E-CARe_5 extraites du tableau 4.11. La règle E-CARe_2 est déclenchée par un évènement métier exprimant une demande de calcul d'itinéraire. Cette règle vérifie le type d'empêchement et sélectionne uniquement les itinéraires comportant un arrêt adapté, ce qui correspond à une action d'adaptation d'information (voir figure 4.12). La règle E-CARe_5 (figure 4.13) permet de notifier un incident véhicule au voyageur uniquement si ce véhicule est utilisé dans un des itinéraires habituels.

Une formalisation complète des règles E-CARe avec des diagrammes de classes doit être réalisée pour faciliter la conception et l'implémentation de ces règles dans un moteur d'évènements ou un moteur de règles. De plus, l'ensemble des évènements manipulés dans le système peut ainsi être identifié.

4.6.2 Spécification du modèle d'acquisition du contexte

La réalisation de cette étape est indépendante de l'étape précédente, c'est pourquoi il est possible de réaliser ces deux étapes en parallèle (voir processus de la phase de la figure 4.10).

L'objectif de cette étape est d'identifier pour toutes les données contextuelles leurs origines en définissant l'ensemble des évènements et leurs sources selon le métamodèle d'acquisition défini par la figure 4.14. En effet, dans cette phase orientée évènements, il est utile de relier la vue structurelle du

TABLE 4.11 – Règles E-CARe avec les évènements déclencheurs

Evènement	Condition contextuelle	Action d'adaptation	Identifiant
Demande de calcul d'itinéraire	La localisation correspond au point de départ	Calculer l'itinéraire	E-CARe_1
	Une personne empêchée : avec une poussette, des bagages ou béquilles	Calculer un itinéraire avec un arrêt aménagé	E-CARe_2
	Une personne avec fauteuil roulant	Calculer un itinéraire avec un véhicule et un arrêt aménagé	E-CARe_3
	Une personne avec un vélo	Calculer un itinéraire avec un véhicule aménagé	E-CARe_4
Incident véhicule	Un incident concernant un véhicule d'un itinéraire habituel (panne, accident, retard)	Informer le voyageur	E-CARe_5
Perturbation travaux	Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule d'un itinéraire habituel	Informer le voyageur	E-CARe_6
Manifestation	Une manifestation sur le chemin d'un itinéraire habituel (dans un rayon de 500m)	Informer le voyageur	E-CARe_7
Perturbation trafic	Une alerte trafic sur le chemin d'un itinéraire habituel	Informer le voyageur	E-CARe_8
Incident véhicule	Un incident concernant un véhicule de l'itinéraire (panne, accident, retard)	Informer le voyageur	E-CARe_9
Perturbation travaux	Un incident concernant l'infrastructure (travaux, route glissante) et affectant les arrêts desservis par un véhicule de l'itinéraire	Informer le voyageur	E-CARe_10
Manifestation	Une manifestation sur le chemin de l'itinéraire (dans un rayon de 500m)	Informer le voyageur	E-CARe_11
Perturbation trafic	Une alerte trafic sur le chemin de l'itinéraire	Informer le voyageur	E-CARe_12
-	Une personne malvoyante	Envoyer une information sonore	E-CARe_13
	Une personne malentendante	Envoyer une information vibrante et textuelle	E-CARe_14
	Des préférences d'affichage	Interagir en respectant ces préférences	E-CARe_15
–	Une personne invalide en déplacement	Déclencher l'assistance	E-CARe_16
	Une personne invalide qui ne suit pas son itinéraire	Alerter son tuteur	E-CARe_17

contexte (modèle de contexte) à une vue dynamique permettant de capturer l'évolution du contexte en fonction des données du système ou de l'environnement. Ainsi, nous identifions pour l'ensemble des éléments de contexte les évènements qui permettent de les acquérir et leurs sources : métier (découlant des données de l'application même), utilisateur (les données de contexte sont entrées par l'utilisateur) ou externes. Ces évènements sont responsables de la mise à jour continue du contexte. Certains éléments de contexte sont dérivés d'autres éléments de contexte, ce qui explique la non considération de la totalité des éléments de contexte dans le modèle d'acquisition.

Le modèle d'acquisition du contexte est ainsi un diagramme de classes instanciant le métamodèle

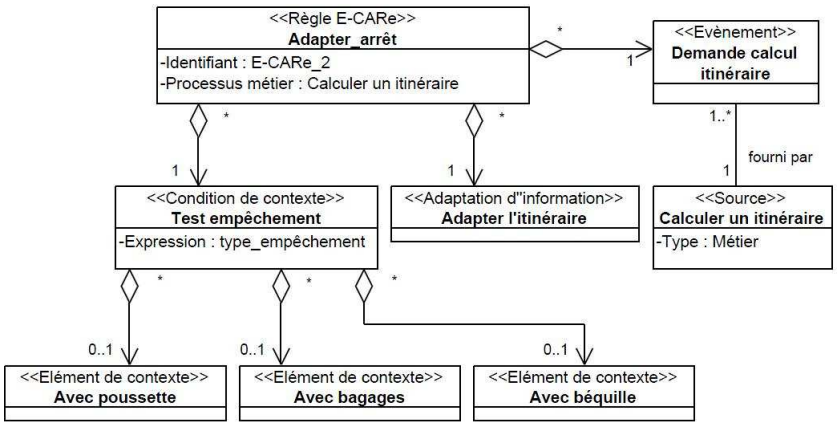


FIGURE 4.12 – Structure de la règle permettant un calcul d'itinéraire prenant en compte un empêchement

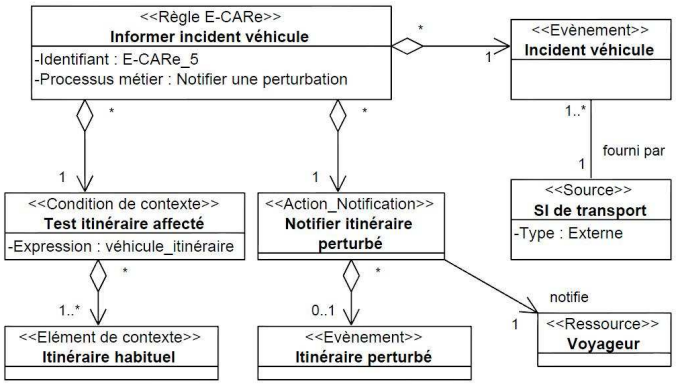


FIGURE 4.13 – Structure de la règle permettant de notifier un incident de véhicule utilisé habituellement par le voyageur

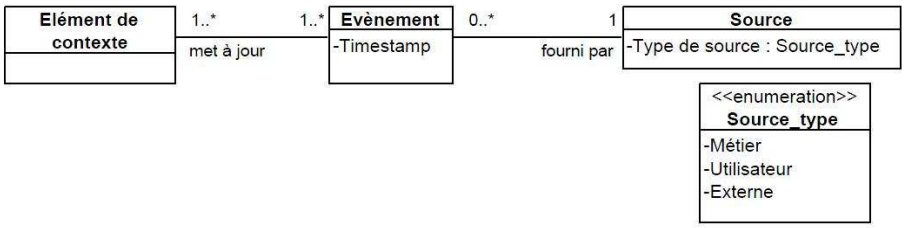


FIGURE 4.14 – Métamodèle d'acquisition du contexte

de la figure 4.14. La figure 4.15 représente le modèle d'acquisition du contexte correspondant à l'application d'assistance au voyageur. Dans ce modèle, certains éléments de contexte sont obtenus par des évènements métiers : par exemple l'évènement "Activer _itinéraire" fourni par le PM "Assister un itinéraire" met à jour l'élément de contexte "itinéraire courant". Certains éléments de contexte sont définis par le voyageur, ce sont par exemple ses préférences ou ses empêchements. D'autres éléments de contexte tels que la localisation ou les données de transport sont fournis par des sources externes.

Le modèle d'acquisition du contexte permet de recenser une partie des évènements utilisés pour des objectifs ubiquitaires. La totalité des évènements est définie dans l'étape suivante.

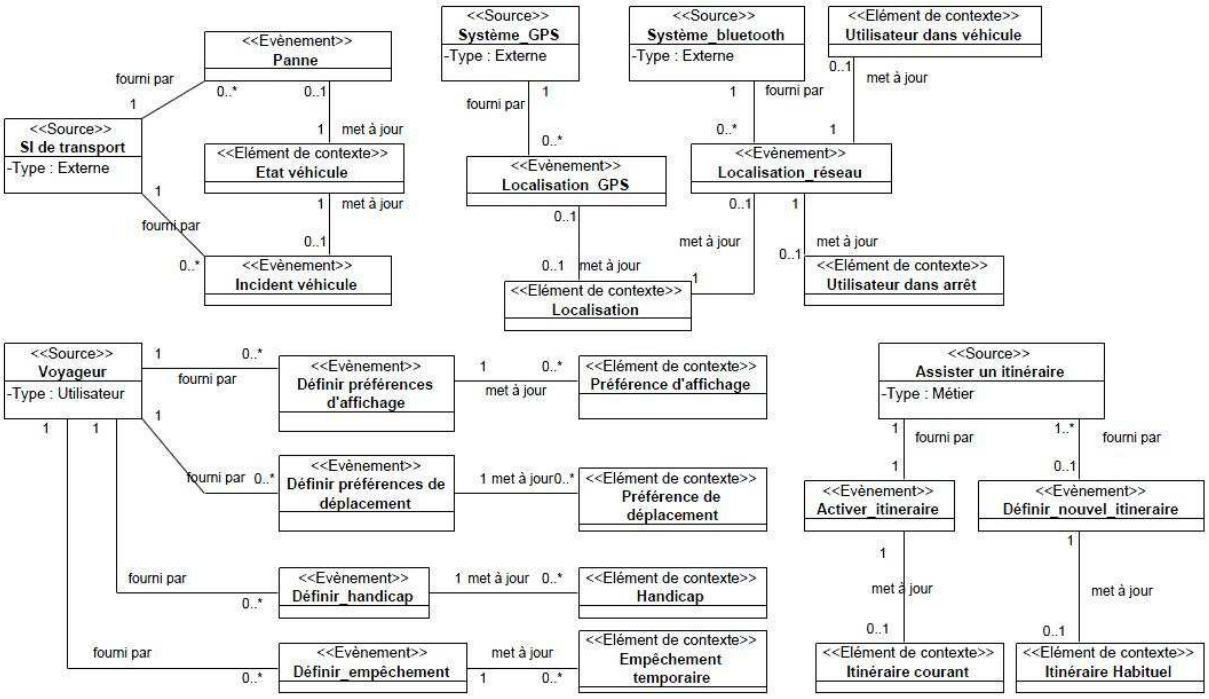


FIGURE 4.15 – Modèle d'acquisition du contexte de l'application d'assistance au voyageur

4.6.3 Spécification de la cartographie d'évènements

La conception d'une application ubiquitaire, dans notre approche orientée évènements, nécessite une gestion des évènements dans un ensemble de moteurs d'évènements. Ainsi, la définition de l'ensemble des évènements, de leurs structures et de leurs usages est effectuée dans cette étape. L'objectif est d'utiliser les modèles de règles et d'acquisition de contexte pour définir la totalité des évènements dans une cartographie d'évènements qui est en réalité un diagramme de classes obtenu après instanciation du métamodèle d'évènements de la figure 4.16 (ce même métamodèle est décrit en détails dans le chapitre 3 à la section 3.3.2). Cette étape est réalisée automatiquement par un jeu de transformations de modèles.

Approche IDM

Notre objectif est d'utiliser les modèles de règles et d'acquisition de contexte pour produire des modèles d'évènements respectant le métamodèle d'évènements de la figure 4.16. Ainsi, nous adoptons une approche IDM en utilisant comme métamodèles de départ le métamodèle de règles (voir figure 4.11) et le métamodèle d'acquisition du contexte (voir figure 4.14) et nous appliquons un ensemble de règles de transformation pour construire le modèle cible. Les règles de transformation concernant le métamodèle de règles sont les suivantes :

- Un patron d'évènements qui déclenche une règle est transformé en un évènement complexe en conservant sa composition en évènements. Une action de déclenchement de la règle est attachée à cet évènement complexe.
- Un évènement déclenchant une règle est transformé en un évènement temporel s'il possède un timestamp. Dans le cas ou l'évènement est relié à une source, alors il s'agit d'un évènement simple qui doit être transformé en un évènement externe, métier ou utilisateur selon le type de la source. La source de l'évènement et la règle déclenchée doivent être attachées à cet évènement.

– Un évènement relié à une action de notification d'une règle doit être transformé en un évènement temporel (s'il possède un timestamp) ou simple : métier, utilisateur ou externe (s'il est relié à une source de ce type). Cet évènement doit être relié aux ressources qu'il notifie.

Les règles qui ne sont pas déclenchées par des évènements ne peuvent pas être utilisées dans cette étape. Elles seront attachées à des emplacements métier dans les PM pour qu'elles soient exécutées quand il y a besoin et quand leurs conditions contextuelles sont vérifiées. Par exemple, dans le tableau 4.11, les règles E-CARe_13 à E-CARe_17 ne comportent pas d'évènement.

La règle de transformation appliquée au métamodèle d'acquisition du contexte est la suivante.

– Un évènement est transformé en un évènement temporel (s'il possède un timestamp) ou en un évènement simple (métier, utilisateur ou externes). Cet évènement doit être relié à sa source et aux éléments de contexte qu'il met à jour.

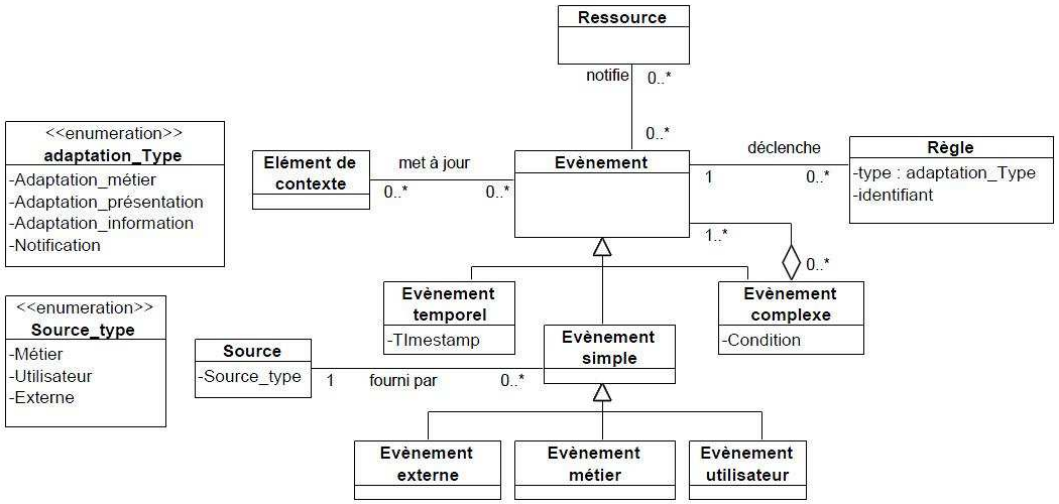


FIGURE 4.16 – Métamodèle d'évènements

En plus de ces règles, il convient de vérifier à chaque ajout d'évènement s'il existe déjà ou non. Dans le premier cas, il suffit d'attacher la nouvelle action à cet évènement.

Le résultat de cette étape est un diagramme de classes contenant l'ensemble des évènements. Par exemple, la cartographie d'évènements obtenue dans le cas de l'application d'assistance est représentée dans la figure 4.17. L'automatisation de cette étape avec l'usage d'Atlas Transformation Language (ATL) [3] pour la transformation de modèles est décrite dans l'annexe E.1.

4.7 Synthèse

Ce chapitre a développé la spécification et l'analyse des besoins ubiquitaires pour produire des modèles de systèmes ubiquitaires en respectant notre vision et nos concepts de base présentés dans le chapitre précédent. Nous proposons une démarche ubiquitaire qui constitue la branche ubiquitaire de la démarche E-CARe composée d'un ensemble de phases permettant de produire l'ensemble des modèles ubiquitaires. Les deux premières phases sont réalisées par la collaboration de spécialistes métier et techniques alors que les deux phases suivantes sont des phases purement ubiquitaires et ne font intervenir que l'analyste-concepteur de contexte qui joue un rôle majeur dans l'identification et le choix des besoins ubiquitaires vu les compétences et les moyens qu'il possède.

La première phase des spécifications ubiquitaires est une phase classique utilisant le cahier des charges pour structurer de façon globale les besoins du client et les placer dans le cadre de l'or-

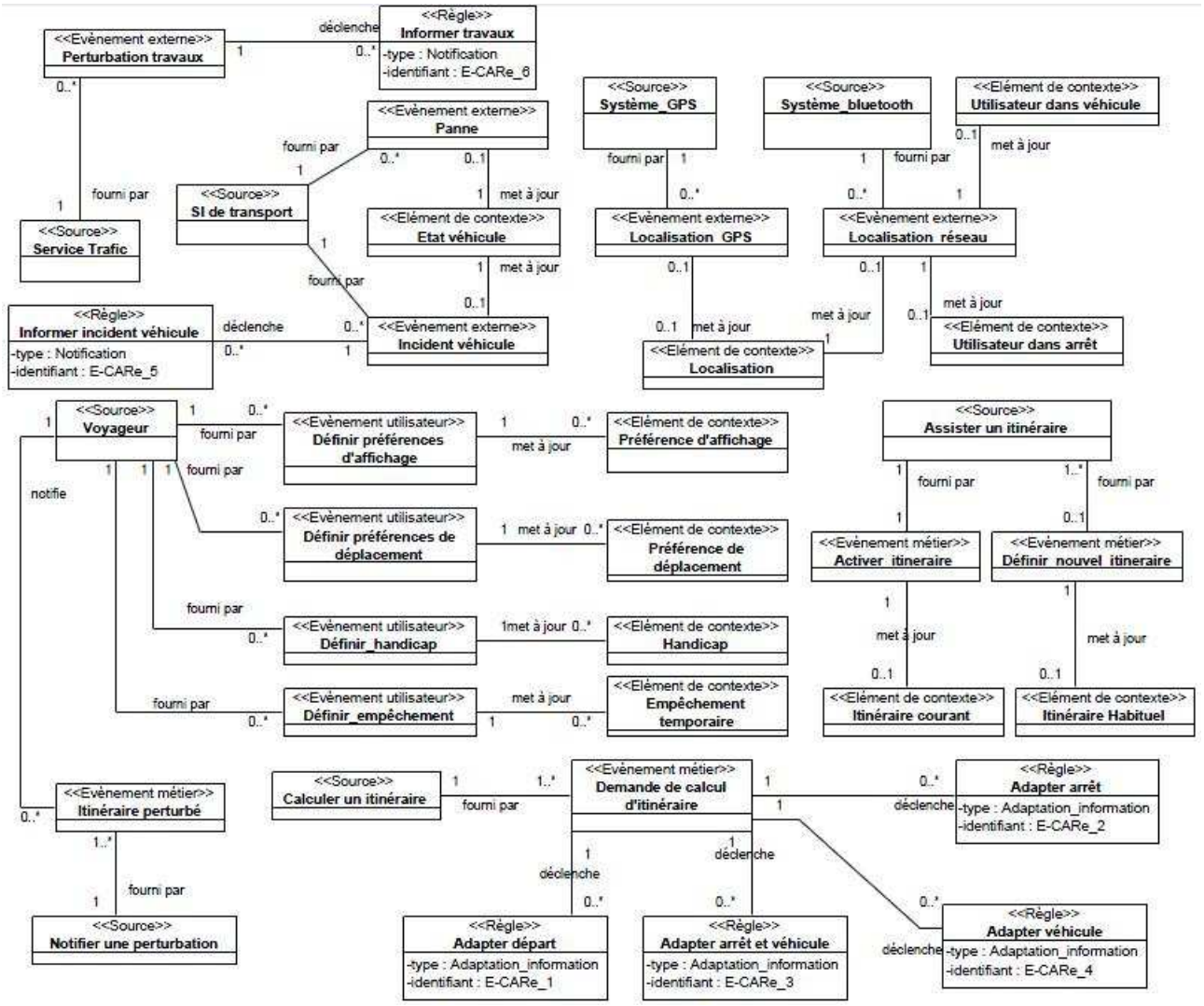


FIGURE 4.17 – Extrait de la cartographie finale d'évènements

ganisation cible. La deuxième phase résulte d'une approche intentionnelle pour l'identification des besoins ubiquitaires. Elle permet d'utiliser les objectifs de trois types (fonctionnels, ubiquitaires et non fonctionnels) pour produire des besoins métier, ubiquitaires et techniques sous forme de règles.

La branche ubiquitaire de E-CARe se concentre par la suite à la spécification des besoins ubiquitaires qui est réalisée par l'analyste-concepteur de contexte. La phase de la modélisation structurelle du contexte comporte un ensemble d'étapes permettant l'analyse détaillée des règles ubiquitaires jusqu'à l'identification des données de contexte. La phase suivante concerne la modélisation évènementielle du contexte par application d'une approche orientée évènements. Elle permet de spécifier les flux d'évènements responsables de déclencher des comportements ubiquitaires ou d'acquérir les données du contexte. La réalisation de cette phase bénéficie de l'approche IDM pour automatiser certaines étapes.

A l'issue des spécification ubiquitaires de la démarche E-CARe, les modèles ubiquitaires suivants sont générés : un modèle de contexte, un modèle de règles et une cartographie d'évènements. Ces modèles doivent s'intégrer de nouveau avec les besoins fonctionnels et techniques pour permettre le passage à la conception et l'implémentation. Ainsi, nous avons besoin de considérer des aspects

classiques de conception des applications et comment y intégrer les modèles ubiquitaires. Le chapitre suivant se concentre sur l'usage de la démarche de conception de SI "Symphony" et son enrichissement avec la branche ubiquitaire pour former la démarche complète de conception des SI ubiquitaires : E-CARe.

Chapitre 5

E-CARe : démarche complète d'ingénierie des SI ubiquitaires

Le chapitre précédent présente la branche ubiquitaire de la démarche E-CARe qui s'intéresse uniquement au développement des modèles ubiquitaires indépendamment des aspects fonctionnels et techniques de l'application. Cependant, les aspects fonctionnels et techniques doivent être considérés et liés aux modèles ubiquitaires pour fournir un système complet et utilisable.

Le domaine d'ingénierie des systèmes d'information profite d'une expérience de plusieurs décennies qui ont donné naissance à plusieurs méthodes visant l'amélioration et l'optimisation du travail de développement (voir chapitre 2). Des méthodes comme MERISE, RUP et les méthodes Agiles résultent de cette évolution. Ainsi, nous profitons de ces méthodes et de l'expérience de l'équipe SIGMA [17] en cette matière pour proposer une méthode complète d'ingénierie des systèmes ubiquitaires qui s'appuie sur la méthode de développement de SI Symphony [Godet-Bar, 2009].

Ce chapitre présente cette démarche complète que nous nommons "E-CARe" en détaillant les différentes phases qui forment un cycle en forme de "ψ". Nous présentons pour chaque phase ses différentes étapes, les tâches automatiques par transformations de modèles, les tâches du développeur ainsi que des exemples extraits de l'application de rénovation du SI voyageur.

5.1 Présentation générale de la démarche E-CARe

La démarche E-CARe est une extension de la démarche Symphony [Godet-Bar, 2009] par intégration de la branche ubiquitaire. Ceci permet de créer une démarche de développement de SI couvrant toutes les phases et les spécialités de développement d'un SI.

5.1.1 Démarche Symphony

La démarche Symphony, développée au sein de l'équipe SIGMA [17] en collaboration avec la société UMANIS [20], est une démarche en "Y" qui permet de couvrir les aspects fonctionnels et techniques de développement des SI. Une description détaillée de la démarche Symphony est fournie dans le chapitre 2 à la section 2.2.3. Le choix de cette démarche est justifié par les faits suivants :

- La démarche Symphony est le fruit de plusieurs évolutions visant à améliorer la réutilisation et la réutilisabilité qui sont nécessaires pour tout projet de développement de SI ubiquitaires puisque ces derniers risquent de subir de multiples changements et évolutions de besoins.
- Symphony sépare la spécification des besoins fonctionnels et techniques et permet de les fusionner dans la phase de conception, ce qui facilite l'intégration des besoins ubiquitaires influençant

les modèles fonctionnels et techniques.
- Symphony est orientée métier et intègre une approche fonctionnelle qui peut être utilisée pour l'identification des fonctionnalités ubiquitaires.
- Symphony intègre un modèle de composants métier tripartite original, le modèle des Objets Métier, que nous pouvons utiliser pour construire des composants métier intégrant en plus les données contextuelles exploitées dans les fonctionnalités du système.

Il est tout à fait possible d'utiliser une autre démarche d'ingénierie des SI en adaptant ses phases en fonction des caractéristiques des systèmes ubiquitaires. Le plus important est de définir des phases d'intégration permettant de faire le lien avec la démarche ubiquitaire E-CARe. Ainsi, la démarche E-CARe doit être cohérente et les modèles ubiquitaires doivent s'accorder avec les modèles originels de Symphony. Ceci est garanti par le cycle de vie original défini ci-après.

5.1.2 Cycle de la démarche "E-CARe"

L'intégration de la branche ubiquitaire dans la démarche Symphony se fait par l'ajout d'une nouvelle branche ubiquitaire entre les deux branches fonctionnelle et technique pour former un cycle en "ψ". Ce cycle est présenté par la figure 5.1 et comporte trois branches séparant les besoins ubiquitaires, fonctionnels et techniques ainsi qu'une branche commune permettant l'intégration des 3 branches précédentes et l'implémentation et le déploiement du SI. Chaque branche est composée de phases issues ou non de Symphony : certaines phases Symphony sont conservées dans la démarche E-CARe (boîtes blanches dans la figure 5.1), alors que d'autres phases Symphony sont modifiées par notre approche (boîtes noires dans la figure 5.1). Enfin, certaines phases sont ajoutées à la démarche Symphony (boîtes à fonds colorés dans la figure 5.1), ce sont les phases de la démarche E-CARe ainsi que deux nouvelles phases d'intégration : la première phase d'intégration permet de rapprocher la branche ubiquitaire de la branche fonctionnelle et la deuxième permet de rapprocher la branche ubiquitaire de la branche technique.

Ce chapitre présente brièvement la démarche E-CARe dans sa globalité. Les phases d'intégration et la phase de conception sont présentées plus en détail.

Dans certaines phases, des approches de l'Ingénierie Dirigée par les Modèles (IDM) sont utilisées pour la génération automatique de modèles.

5.1.3 Usage de l'IDM

L'Ingénierie Dirigée par les Modèles permet de construire de façon automatique des modèles à partir d'autres modèles. Dans le cas des grosses applications ou des SI, l'IDM permet de faciliter la tâche aux développeurs surtout quand il s'agit de dériver de façon automatique des modèles volumineux sans prises de décision ou interventions humaines. Nous avons recours à l'IDM pour faciliter et simplifier certaines phases de notre démarche, cet usage est différent selon les trois objectifs suivants :
- vérification de la cohérence : permet de vérifier certaines conditions portant sur des éléments de deux ou plusieurs modèles. Le résultat de la vérification est un modèle représentant les éléments incohérents et conformes à un métamodèle de problèmes tel que dans [Bézivin et Jouault, 2006]. Cette approche est utile dans les phases d'intégration qui doivent garantir la cohérence des modèles fournis par les différentes branches.
- transformation de modèles : permet de produire à partir d'un ou de plusieurs modèles de nouveaux modèles. C'est l'usage classique de l'IDM où un ensemble de métamodèles d'entrée fournit les concepts nécessaires pour construire une instanciation des métamodèles de sortie.
- enrichissement de modèles : permet d'utiliser certains modèles pour enrichir d'autres modèles ou encore fusionner certains modèles pour former des modèles plus larges et plus détaillés. Cette

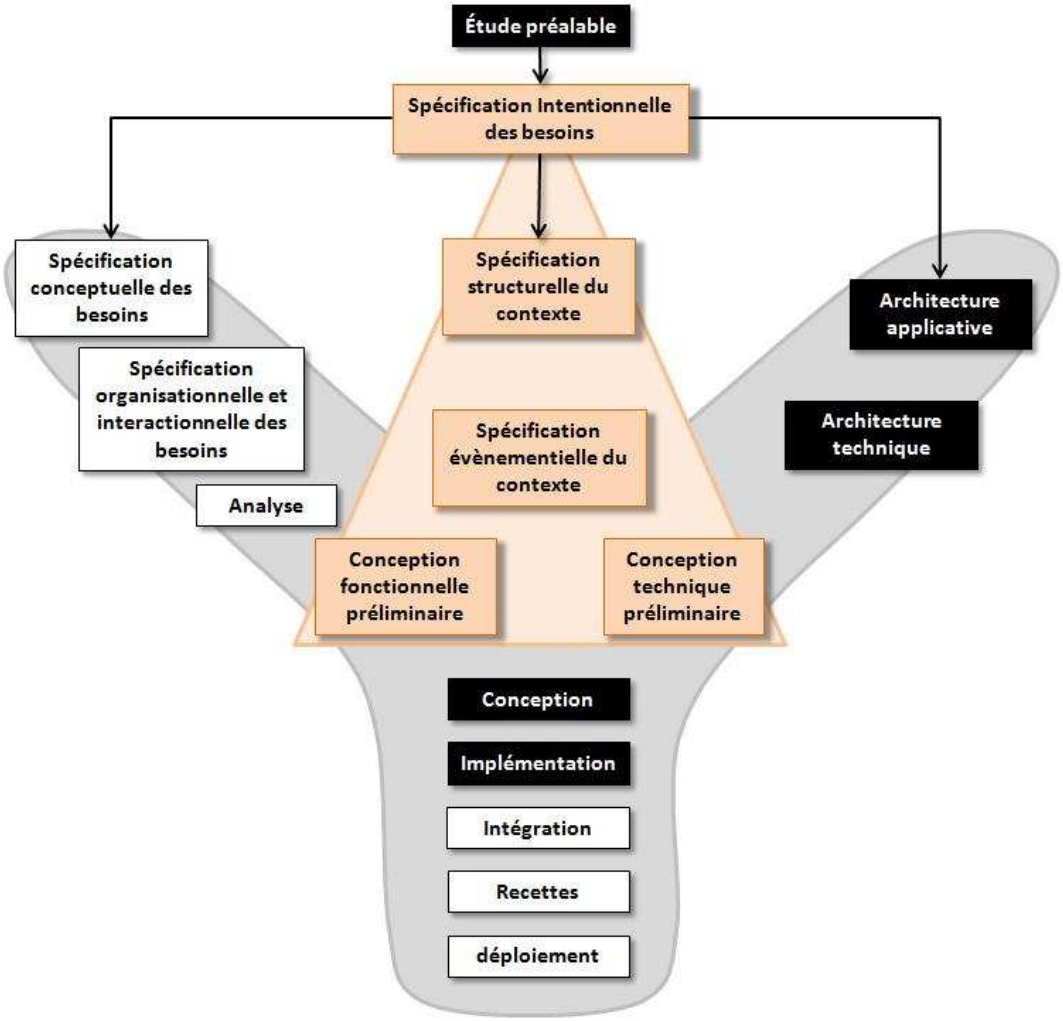


FIGURE 5.1 – Cycle de la démarche "E-CARe"

approche permet d'ajouter des détails à certains modèles notamment dans le cas de fusion de modèles ubiquitaires et techniques ou de modèles fonctionnels et ubiquitaires.

L'usage de l'IDM dans notre démarche est récapitulé dans le schéma de la figure 5.2 qui montre les différents modèles de la démarche et les transformations appliquées pour les générer. Une explication détaillée de chaque transformation est fournie dans la description des phases présentée dans la suite de ce chapitre.

5.2 Branche fonctionnelle

Cette branche comporte des phases métier permettant d'étudier et de spécifier de façon graduelle et organisée les modèles de PM et les modèles interactionnels. Les phases de cette branche sont conservées dans la démarche E-CARe et ne sont décrites ici que pour appliquer notre étude de cas et produire des modèles métier que nous exploitons dans la suite de la démarche. En effet, un SI ubiquitaire comme tout autre SI doit être spécifié d'un point de vue métier pour expliciter et formaliser ses fonctionnalités. Certes certaines fonctionnalités peuvent avoir un caractère ubiquitaire qui correspond à un besoin d'adaptation ou de sensibilité au contexte, mais ces fonctionnalités sont

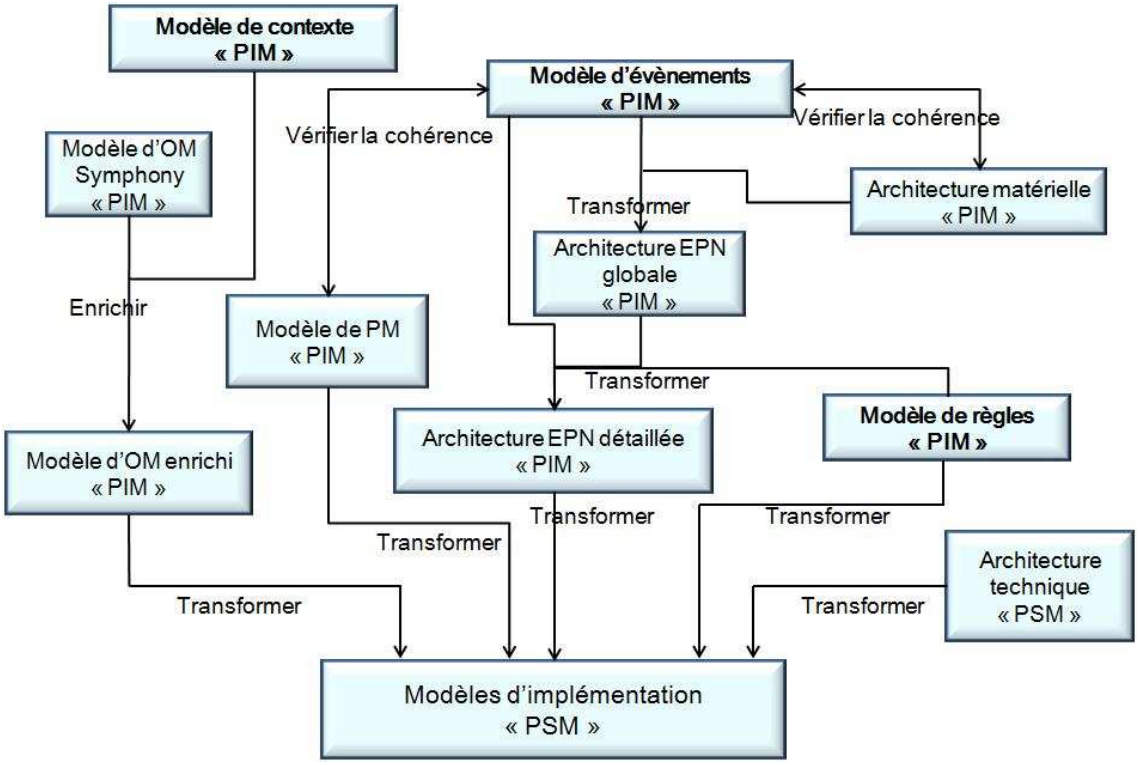


FIGURE 5.2 – Transformations IDM dans la démarche E-CARe

spécifiées parallèlement aux modèles ubiquitaires qui sont pris en charge par la démarche E-CARe.

La branche fonctionnelle, dont le responsable est le concepteur métier, comporte trois phases réalisées de façon itérative pour chaque PM identifié dans la phase intentionnelle de spécification des besoins (voir chapitre 4). Ces phases sont détaillées ci-dessous.

5.2.1 Spécification conceptuelle des besoins

L'objectif de cette phase est d'identifier les différents besoins métier et les structurer dans des cas d'utilisation détaillés et décomposés. Il s'agit ici d'une formalisation détaillée du contenu de chaque PM et sa réorganisation selon une optique métier plus précise que l'optique intentionnelle menée dans la phase précédente. La définition de la structure statique du système dépend de la bonne description des détails métier et de leur dynamique qui peut subir de nouveaux choix et de nouveaux besoins de conception décelés par le concepteur métier. Cette phase comporte les étapes suivantes :

- **Description conceptuelle de chaque PM sous la forme d'un scénario principal** nous partons des PM décomposés selon les objectifs (voir section 4.4.2 du chapitre précédent) et nous mettons en évidence les interactions avec les acteurs externes, les pré et les post-conditions et les règles métier identifiées dans la phase précédente de spécification intentionnelle des besoins (voir dans la section 4.4.3 le tableau 4.7).
  Nous préconisons pour cette étape l'usage d'un scénario de cas d'utilisation comme le suivant qui est appliqué au PM "Assister un itinéraire" (voir section 4.4.2 du chapitre précédent) :

Titre : Assister un itinéraire.
Résumé : Ce processus permet de suivre une personne utilisant les transports publics dans son trajet et de l'assister en recommandant les changements de mode

de transport et en notifiant des alertes au cours du déplacement.
Acteurs : SIM, voyageur, système de localisation, SIT, tuteur.
Préconditions : Personne localisée dans le réseau de transport.
Evènements déclencheurs : demande de l'usager ou conditions de souscription satisfaites (si souscription au service).
Scénario nominal :
- chargement de l'itinéraire personnalisé,
- chargement de la localisation,
- repérage du voyageur dans le trajet de l'itinéraire,
- suivi de l'itinéraire.

Scénarios alternatifs : Si la localisation de l'usager ne correspond à aucune portion de l'itinéraire, alors le voyageur doit choisir un nouvel itinéraire ou annuler l'assistance.
Règles métier :
- Le voyageur peut notifier des évènements qu'il rencontre.
- Si le voyageur modifie ses données pendant un itinéraire alors il faut recalculer l'itinéraire.
- Le voyageur peut à tout moment annuler l'assistance.

Postconditions : arrivée à destination ou prise en charge par un tuteur.

- **Description conceptuelle formelle de chaque PM** qui permet de documenter le cas d'utilisation établi précédemment avec un diagramme de séquence.

Dans la version étendue de Symphony présentée par [Godet-Bar, 2009], des étapes permettant d'identifier des besoins interactionnels sont ajoutées qui sont : l'élaboration des prescriptions ergonomiques, l'identification du type d'interaction et la sélection de panels représentatifs d'utilisateurs. Ces étapes sont nécessaires et recommandées dans le cas des SI ubiquitaires qui sont des systèmes interactionnels par excellence.

5.2.2 Spécification organisationnelle et interactionnelle des besoins

Cette phase, définie dans [Jausseran, 2005] comme une phase purement organisationnelle et enrichie dans [Godet-Bar, 2009] pour ajouter une dimension interactionnelle, permet de déterminer le « qui fait quoi et quand »du projet. Il s'agit de définir la structure détaillée des PM en incluant le comportement des acteurs internes. La spécification interactionnelle est mise en œuvre par la définition du choix du style d'interaction effectué dans la phase précédente. Cette phase comporte les étapes suivantes :

- **Décomposition organisationnelle des PM** qui décompose les PM en des activités pour élaborer un diagramme d'activités UML représentant une base organisationnelle pour les spécifications fonctionnelles et interactionnelles. Nous utilisons dans cette étape BPMN (Business Process Modeling Notation)[6] comme alternative au diagramme d'activités [Ben Cheikh *et al.*, 2009a] car il représente désormais un standard reconnu et supporté par la plupart des systèmes pour la modélisation des PM. La figure 5.3 illustre le diagramme BPMN du processus "Assister un itinéraire" et présente le scénario nominal et les scénarii alternatifs. L'utilisation de BPMN pour produire des modèles de PM spécifiques au domaine de transport est décrite dans [Ben Cheikh *et al.*, 2009c].
- **Organisation des PM** qui permet d'extraire les activités informatisées du diagramme BPMN pour les factoriser et les organiser dans des cas d'utilisation.
- **Description de la cartographie des OM Processus** réalisant les cas d'utilisation tels que les

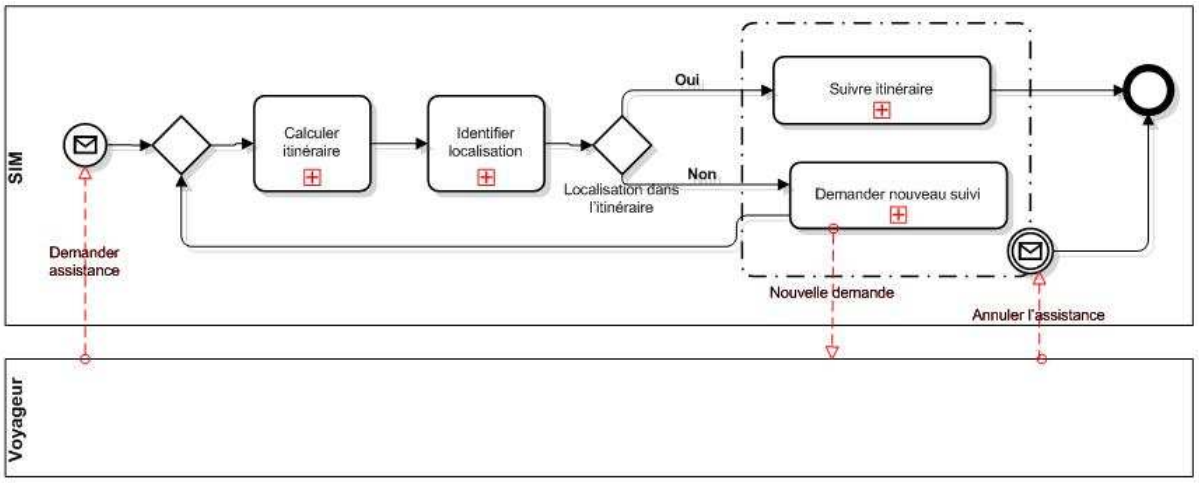


FIGURE 5.3 – Diagramme BPMN du processus "Assister un itinéraire"

OM processus "Assister un itinéraire", "Suivre un itinéraire" et "Calculer un itinéraire".

- **Spécification interactionnelle des besoins** qui permet de décrire l'interface homme-machine en focalisant sur les acteurs internes et en utilisant les prescriptions ergonomiques et les activités informatisées et manuelles.
- **Description de la cartographie des Objets Interactionnels** qui peuvent être des objets interactionnels entités tels que : "Message vocal", "Cartes du réseau" et "Zone d'affichage" ou des objets interactionnels processus tels que "Identifier itinéraire sur une carte".
- **Connexion des espaces métier et interactionnels** qui est une étape décrite exhaustivement dans [Godet-Bar, 2009] et qui permet de produire un prototype des interfaces supportant les fonctionnalités du système.

Les modèles produits à l'issue de cette phase fournissent une vision détaillée des PM et des interactions connectées dans un modèle de cas d'utilisation. Les étapes interactionnelles sont primordiales dans un système ubiquitaire qui peut s'intégrer dans l'environnement ambiant pour faire intervenir des modes innovants d'interaction. Ainsi, le spécialiste IHM doit prendre en compte les fonctionnalités ubiquitaires dans l'objectif de faciliter et d'améliorer les interactions.

5.2.3 Analyse

Cette phase met en évidence ce que doit faire le système sans tenir compte de la manière de le réaliser. Elle permet de modéliser le comportement et la structure des OM Processus, Interactionnel et Entité. Elle est composée d'une analyse dynamique et d'une analyse statique.

- **Analyse dynamique.** Elle permet de formaliser de façon détaillée le scénario principal et les scénarios détaillés de chaque cas d'utilisation avec des diagrammes de séquence. La prise en compte des OM entité dans les diagrammes dynamiques des PM nécessite la construction de diagrammes de séquence représentant les lignes de vie de chaque OM et les interactions entre eux. Les diagrammes de séquence ainsi construits permettent de capturer les interfaces des OM et de faciliter par la suite la construction de la cartographie finale des OM.
  La figure 5.4 représente une partie du diagramme de séquence du PM "Assister un itinéraire" et fait apparaître les entités métier utilisées.
  Il convient aussi de définir les cycles de vie des OM et interactionnels en utilisant des diagrammes d'états UML.

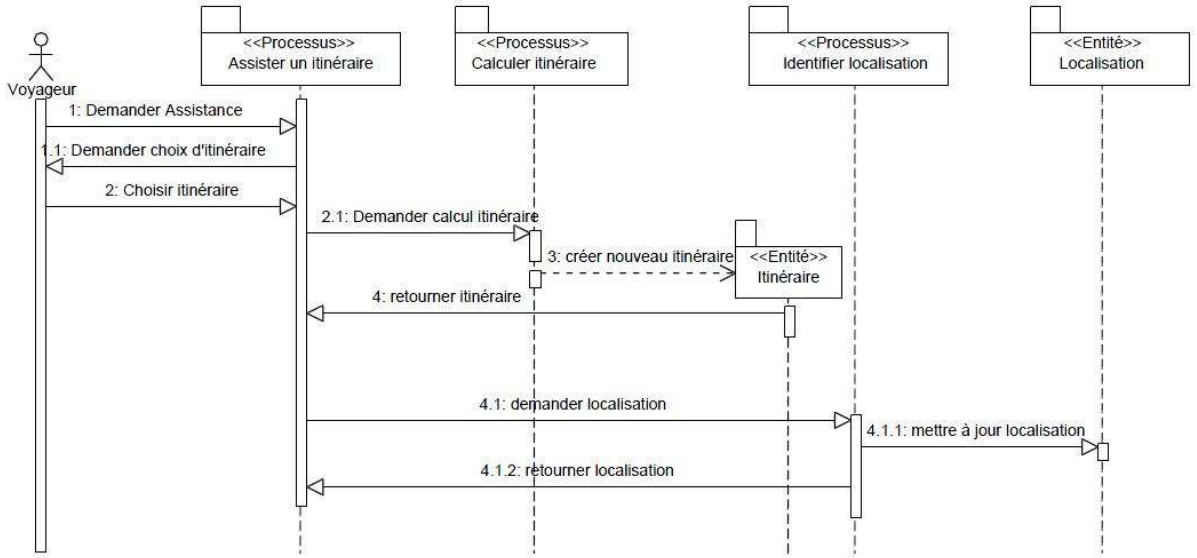


FIGURE 5.4 – Un extrait du diagramme de séquence détaillé du PM "Assister un itinéraire"

– **Analyse structurelle.** Elle permet de recenser les services employés lors de la phase d'analyse dynamique et de les structurer sous forme d'Objets Symphony de niveau analyse. Les entités sont organisées dans des OM entités et leurs classes parties. La figure 5.5 représente l'OM entité Voyageur obtenu lors de l'analyse structurelle. Cet OM contient deux classes parties représentant la localisation et les itinéraires d'un voyageur et une classe interface permettant l'accès à l'OM. La classe rôle de cet OM représente l'entité voyageur en cours d'assistance et permet d'accéder à des services spécifiques à ce rôle.

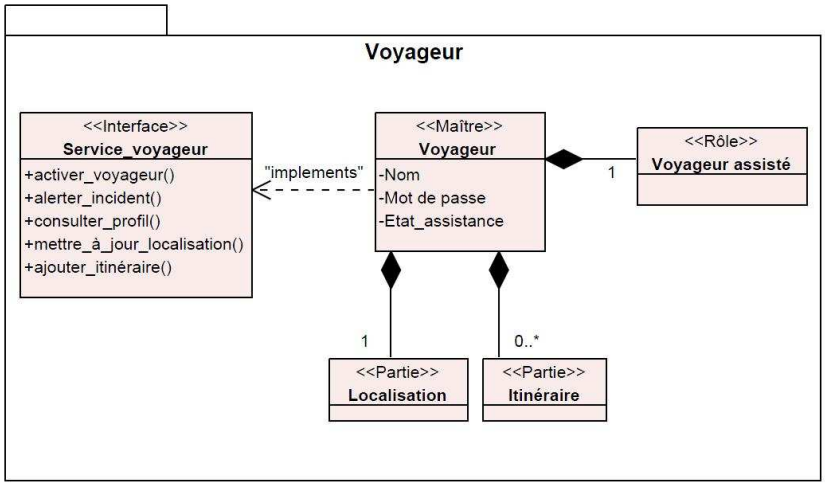


FIGURE 5.5 – Objet Métier (OM) Symphony de l'entité voyageur

5.3 Branche technique

La branche technique dans un processus de développement en "Y" est entièrement indépendante de la branche fonctionnelle. Dans la démarche E-CARe, elle est de plus indépendante des besoins ubi-

quitaires. Néanmoins, la branche technique peut considérer des besoins ubiquitaires non-fonctionnels tels que la distribution et la sécurité qui sont exprimés par des règles techniques identifiées dans la phase intentionnelle du processus E-CARe (voir chapitre précédent).

La branche technique de Symphony contient deux phases : l'architecture applicative et l'architecture technique. Au cours de ces deux phases, l'expert technique doit capturer les besoins techniques, les formaliser et construire des architectures du système et des composants techniques réutilisables avec n'importe quelle spécification fonctionnelle. Nous conservons ces deux phases dans notre démarche mais nous modifions leurs objectifs et leurs contenus pour qu'elles s'adaptent au cas des systèmes ubiquitaires qui exigent l'usage de composants ubiquitaires et la prise en compte des environnements pervasifs et ambiants.

5.3.1 Architecture applicative

Le cahier des charges détaillé établi par le maître d'œuvre dans la phase d'étude préalable doit contenir un ensemble suffisant d'informations sur les prérequis techniques, à savoir le système informatique existant, l'organisation du matériel, la distribution géographique de l'entreprise, les modes de communication, les systèmes à conserver et les systèmes à rénover.

La spécification de l'organisation du système, du matériel existant ou à utiliser et des unités fonctionnelles du projet (ensemble de processus de haut niveau) permet à l'architecte technique de construire une vue initiale du système et de son environnement dans une architecture matérielle. L'architecture matérielle représente la distribution du matériel informatique de l'organisation et les connexions entre eux. Cette architecture peut contenir les éventuelles contraintes techniques reliées au matériel utilisé telles que les performances, l'interopérabilité, la sécurité pour les communications, la volumétrie des données. La prise en compte des règles techniques définies dans la phase intentionnelle permet de guider certains choix architecturaux tels que l'usage de capteurs de systèmes GPS et les liens de sécurité entre les composants matériaux.

Dans notre étude de cas de rénovation d'un SIT, nous rappelons que le projet consiste à rénover les processus de gestion des évènements et des incidents et à fournir des applications mobiles à des usagers du système tels que les voyageurs et le personnel en mobilité (conducteurs et équipe d'intervention). Un SIT est par nature extrêmement distribué, contenant du matériel et des usagers en mobilité. Ces usagers représentent des intentions différentes à l'usage du système et la séparation entre les parties publiques et privées des données est nécessaire. La figure 5.6 représente un exemple d'architecture matérielle du SI de l'entreprise où la distribution des services de l'organisation est considérée en séparant l'unité de maintenance du site central. La séparation du serveur d'exploitation et du serveur d'application central découle du fait que les processus d'exploitation sont peu couplés aux autres applications de l'entreprise et que cette unité gère des données stratégiques de haut niveau (hiérarchique) menées par les décideurs et les gérants.

5.3.2 Architecture technique

Cette phase concerne la construction d'une architecture logique du système qui respecte le framework E-CARe proposé dans le chapitre 3 en vue d'identifier des architectures N-tiers pour l'ensemble des sous-systèmes ou des applications. En effet, le framework E-CARe n'est pas une architecture mais un cadre théorique pour construire des architectures selon les besoins des projets ubiquitaires. Il s'agit dans cette étape d'identifier pour chaque application ou service le positionnement de chaque module dans l'architecture matérielle.

Tout d'abord, il faut faire des choix techniques pour définir l'architecture logicielle du système indépendamment des choix fonctionnels. Cette architecture et ses composants techniques peuvent

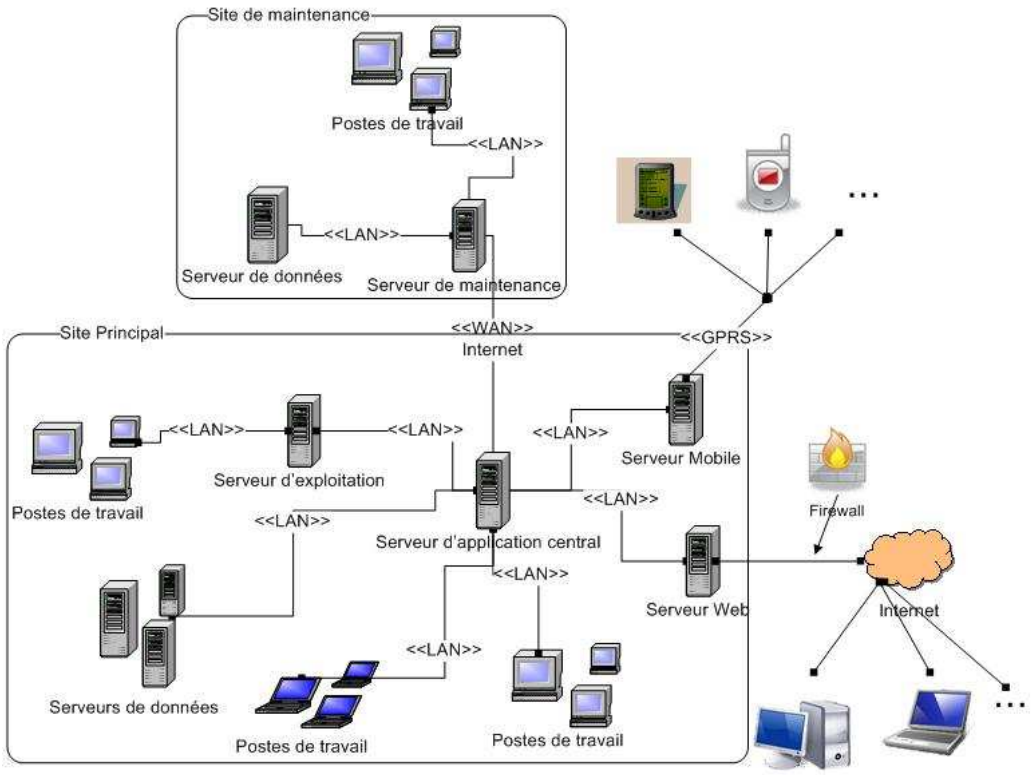


FIGURE 5.6 – Exemple d'architecture matérielle du SIT de l'entreprise

ainsi être réutilisés avec n'importe quelle spécification fonctionnelle. La conception de l'architecture technique doit garantir une prise en compte complète des exigences des systèmes ubiquitaires notamment la mobilité, la sécurité, la réponse aux évènements et le raisonnement sur le contexte. Il s'agit dans cette phase de choisir les composants à utiliser pour chaque module du framework en tenant compte des unités de traitement des évènements à utiliser. Il faut identifier pour chaque unité ses composants et leurs liens avec les autres composants des autres modules : métier, présentation, données, communication et contexte.

Il faut définir de plus les environnements de développement et les langages de programmation à utiliser. La liste suivante énumère les composants possibles dans chaque module :

- **Module présentation,** c'est-à-dire l'ensemble des composants pour l'interaction avec l'utilisateur et la gestion des interfaces. Les composants utilisés diffèrent selon la nature de l'application : application mobile, application web, application client... Parmi les frameworks qui peuvent être utilisés pour créer les interfaces de ce module et effectuer des actions citons ASP.NET, WPF (Windows Presentation Foundation) et WinFORMs. Les interfaces peuvent être créées aussi avec des techniques comme JSP (Java Server Page), JSF (Java Server Faces) ou Struts qui se transforment par compilation en des servlets JAVA. Les technologies les plus reconnues pour créer les pages web dynamiques sont AJAX, Flash, Silverlight.
- **Module communication,** c'est-à-dire les mécanismes à utiliser pour la gestion des souscriptions et l'échange d'informations (ou évènements) (JMS : Java Mailing Service, nServiceBus de .NET), l'authentification et le contrôle d'accès (JAAS : Java Authentication and Authorization Service).
- **Module métier :** c'est-à-dire les moteurs workflow et les moteurs de règles à utiliser. Il est

possible d'utiliser des composant EJB et le framework J2EE.

- **Module contexte :** un moteur d'évènements ou un moteur de règles constituent les composants centraux de ce module. De plus, des composants doivent permettre d'enrichir la gestion du contexte et le raisonnement sur le contexte : par exemple, un context wrapper (obtenir des données des capteurs), un context reasoner (enrichir et agréger les données de contexte), context proxy (cacher les aspects distribués des sources de contexte) et un context broker (interface de découverte des sources de contexte et du type de contexte).
  Enfin, ce module fait le lien avec le module de données et gère les accès aux données, donc elle doit contenir des framework de mapping comme JDO ou ADO.NET.
- **Module données :** c'est-à-dire les SGBD (Oracle, MySQL). Dans le cas des applications mobiles nous recommandons de conserver le traitement des données dans le dispositif mobile car la sécurité des données personnelles et de leur confidentialité constituent des priorités. Ce choix permet de plus de créer des applications mobiles plus indépendantes en minimisant les échanges de données avec le SI central et en permettant à ces applications d'effectuer les traitements et la prise de décision. Dans le cas du SIT, l'échange de données avec le SI central n'est nécessaire qu'en cas de perturbations ou du changement des informations transport utilisées. Ainsi l'application doit contacter le SI central pour récupérer les données qui lui manquent. Cette vision est possible avec l'évolution continue des dispositifs mobiles augmentant leur capacité en mémoire et en CPU.

Le choix de ces technologies et de ces outils n'est pas toujours libre puisqu'il est contraint par les besoins techniques tels que les problèmes de licence, de coût, de réutilisation et de tendances technologiques actuelles. Il permet d'établir une architecture logicielle du système telle que celle montrée en figure 5.7 pour l'application d'assistance sur dispositif mobile.

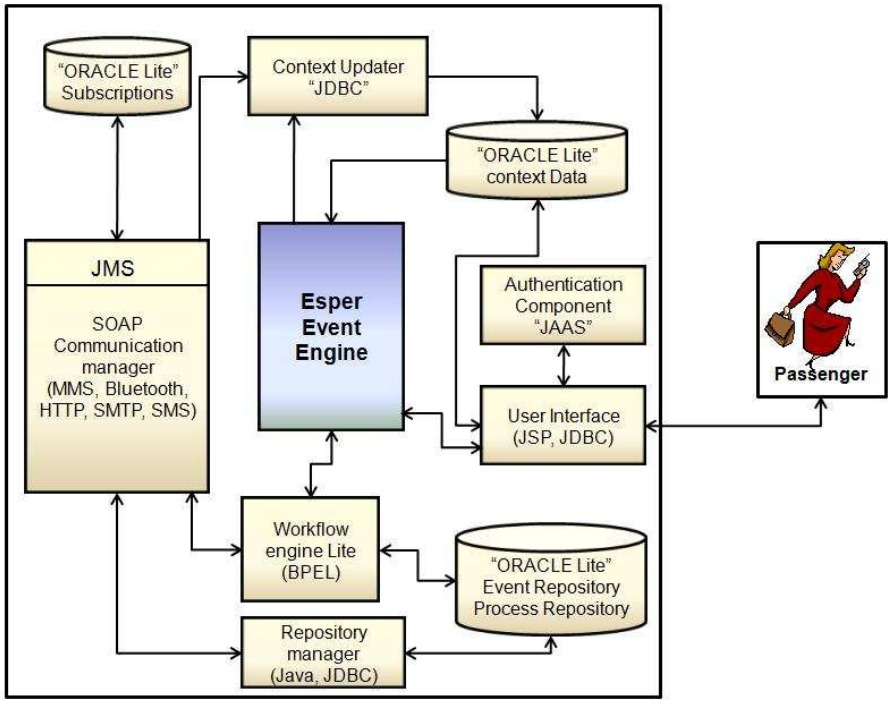


FIGURE 5.7 – Architecture logicielle de l'application d'assistance au voyageur sur un dispositif mobile

5.4 Conception fonctionnelle préliminaire

La phase de conception fonctionnelle préliminaire est une phase d'intégration de la démarche E-CARe dans la démarche Symphony. Elle permet de rapprocher la branche ubiquitaire et la branche fonctionnelle en vérifiant la cohérence des modèles produits et en enrichissant les modèles fonctionnels pour les exploiter dans la phase de conception. Cette phase est constituée de trois étapes (voir figure 5.8) qui permettent de vérifier la cohérence entre les modèles dynamiques et les modèles d'évènements, d'enrichir les modèles des OM avec les données structurelles de contexte et d'enrichir les modèles dynamiques avec les règles d'adaptation. Pour faciliter le travail du concepteur métier, certaines étapes sont automatisées avec des transformations IDM.

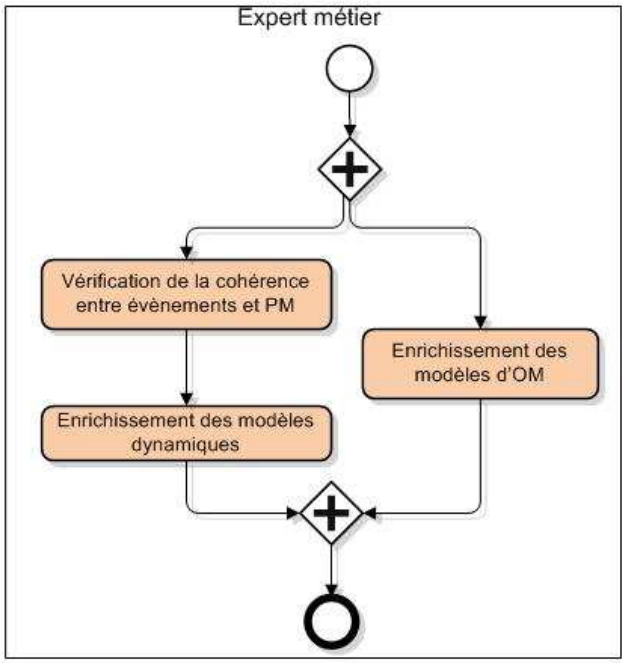


FIGURE 5.8 – Processus de la phase de conception fonctionnelle préliminaire

5.4.1 Vérification de la cohérence entre évènements et PM

Objectifs et principe

Chacune des branches fonctionnelle et ubiquitaire produit des modèles d'évènements. La branche ubiquitaire produit des cartographies d'évènements à usage ubiquitaire tels que la mise à jour du contexte, le déclenchement d'un comportement d'adaptation ou la notification des acteurs. Ces évènements peuvent être des évènements métier, utilisateurs ou externes. La branche fonctionnelle produit des modèles dynamiques de PM sous forme de diagrammes BPMN qui contiennent des évènements qui interviennent implicitement ou explicitement dans le déroulement du processus. Dans le langage BPMN [6], les évènements sont appelés explicitement grâce aux concepts : évènement de début, évènement intermédiaire et évènement de fin. Un évènement est appelé implicitement s'il fait référence à une action telle que le début ou la fin d'une activité ou une interaction.

L'objectif ici est de vérifier la cohérence entre les modèles d'évènements produits par la branche ubiquitaire et les modèles d'évènements fonctionnels. En effet, les évènements métier et utilisateur

contenus dans la cartographie (branche ubiquitaire) doivent être présents dans les modèles dynamiques de la branche fonctionnelle. Ceci garantit le déclenchement des comportements ubiquitaires par les modèles fonctionnels.

Dans le cas où certains évènements métier ou utilisateur ne figurent pas dans les modèles dynamiques, il faut revoir ces modèles pour ajouter les évènements manquants aux modèles de PM. Dans certains cas, un conflit entre noms d'évènements peut surgir, il est alors recommandé de revoir cette dénomination et de choisir le nom attribué par le concepteur de contexte car il est déjà utilisé dans d'autres modèles comme les modèles de règles et les modèles de contexte.

Pour faciliter la vérification de cohérence qui peut devenir une lourde tâche pour de gros systèmes, nous automatisons cette étape en utilisant une approche IDM.

Approche IDM

Pour vérifier la cohérence entre modèles, nous utilisons l'approche proposée par [Bézivin et Jouault, 2006] qui consiste à adopter un métamodèle de problème comme métamodèle produit qui permet de récupérer le résultat de la vérification (réussite ou échec) et de modéliser les incohérences. Dans cette approche, la comparaison est faite entre deux ou plusieurs modèles instanciant un ou plusieurs métamodèles et les concepts sujet de la comparaison doivent être compris dans le métamodèle de problème.

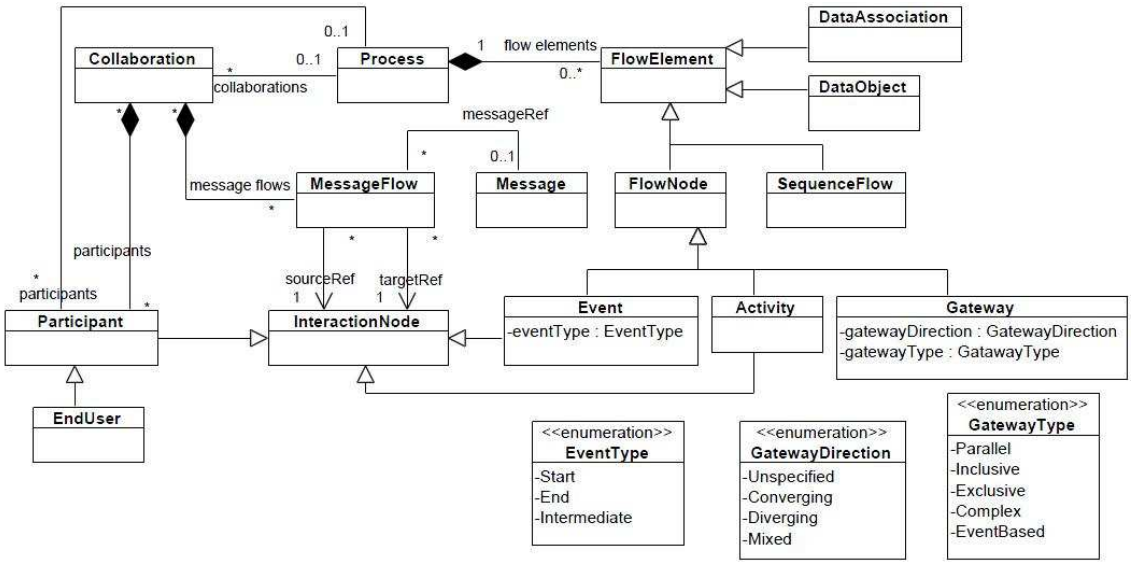


FIGURE 5.9 – Métamodèle simplifié de BPMN

Les métamodèles utilisés sont le métamodèle d'évènements (voir figure 4.16) et un métamodèle de BPMN extrait des spécifications de BPMN [6] (voir figure 5.9). Le métamodèle de problème utilisé est représenté par la figure 5.10 et permet de récupérer le résultat de la vérification et les évènements métier et utilisateur manquants dans les modèles de PM.

Les règles de transformation appliquées dans ce cas sont les suivantes :

- le résultat de la vérification est par défaut "Réussi" ;
- chaque évènement métier de la cartographie instance du métamodèle d'évènements doit apparaître dans au moins un modèle de PM instance du métamodèle BPMN sous le concept "Event" ou sous le concept "MessageFlow" dont la source du message est un processus (figure 5.9).

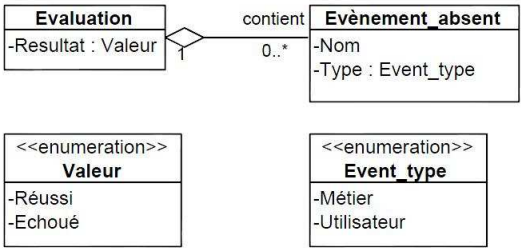


FIGURE 5.10 – Métamodèle de problème pour le résultat de la vérification de cohérence entre modèles d'évènements et PM

- un évènement utilisateur de la cartographie (figure 4.16) doit apparaitre dans au moins un modèle de PM sous le concept "MessageFlow" (figure 5.9) qui a pour source un utilisateur final "EndUser".
- si un évènement n'existe pas alors le résultat de la vérification devient "Echoué" et l'évènement concerné est ajouté au modèle produit de problème avec son type métier ou utilisateur.

L'usage de cette approche de vérification de cohérence avec ATL est illustrée dans l'annexe E.2.

Exemples

Pour notre exemple de SIT, le PM "Assister un itinéraire" défini dans la branche fonctionnelle (dont le scénario est défini dans la section 5.2.1) permet d'assister un voyageur au cours de son déplacement en utilisant le sous-processus "Suivre un itinéraire" (voir figure 5.3). Le modèle dynamique de ce dernier est présenté par le diagramme BPMN de la figure 5.11. La vérification de la cohérence entre la cartographie d'évènements (voir figure 4.17) et ce PM montre que l'évènement nommé "Activer_itinéraire" de la cartographie n'est pas présent dans le PM "Suivre un itinéraire" (voir figure 5.3). Le concepteur métier doit ainsi revoir ce PM et ajouter un évènement de type message attaché à la tâche "Activer l'itinéraire" comme le montre le diagramme PM de la figure 5.12.

5.4.2 Enrichissement des modèles dynamiques

Après avoir corrigé les modèles de PM en ajoutant les évènements nécessaires à des objectifs ubiquitaires, il est judicieux de positionner ces fonctionnalités ubiquitaires dans les modèles de PM qui reflètent les fonctionnalités du système. L'objectif de cette étape est de réaliser le lien entre les règles d'adaptation et les PM.

En effet, les règles d'adaptation obtenues dans la branche ubiquitaire résument l'ensemble des fonctionnalités ubiquitaires sous la forme Évènement-Condition-Action. Elles permettent de réagir à une situation conditionnelle et contextuelle. Certaines règles sont indépendantes des métiers quand elles réagissent à des situations externes sans l'activation d'aucun processus. Par exemple, la règle permettant d'alerter sur un changement d'horaire peut être exécutée sans besoin d'être assisté mais seulement après avoir comparé l'alerte avec le contexte du voyageur.

Dans d'autres cas, certaines activités métier doivent s'exécuter en vérifiant en permanence certaines situations contextuelles. Par exemple, le processus d'assistance au voyageur doit prendre en compte les perturbations ambiantes et adapter l'itinéraire en fonction de ces perturbations.

L'étape d'enrichissement des modèles dynamiques permet de positionner, dans la mesure du possible, ces règles dans les modèles de PM.

La version 2.0 de BPMN [6] permet de prendre en compte les règles métier sous forme d'annotations. Nous introduisons de cette manière les règles d'adaptation dans les PM en BPMN : les

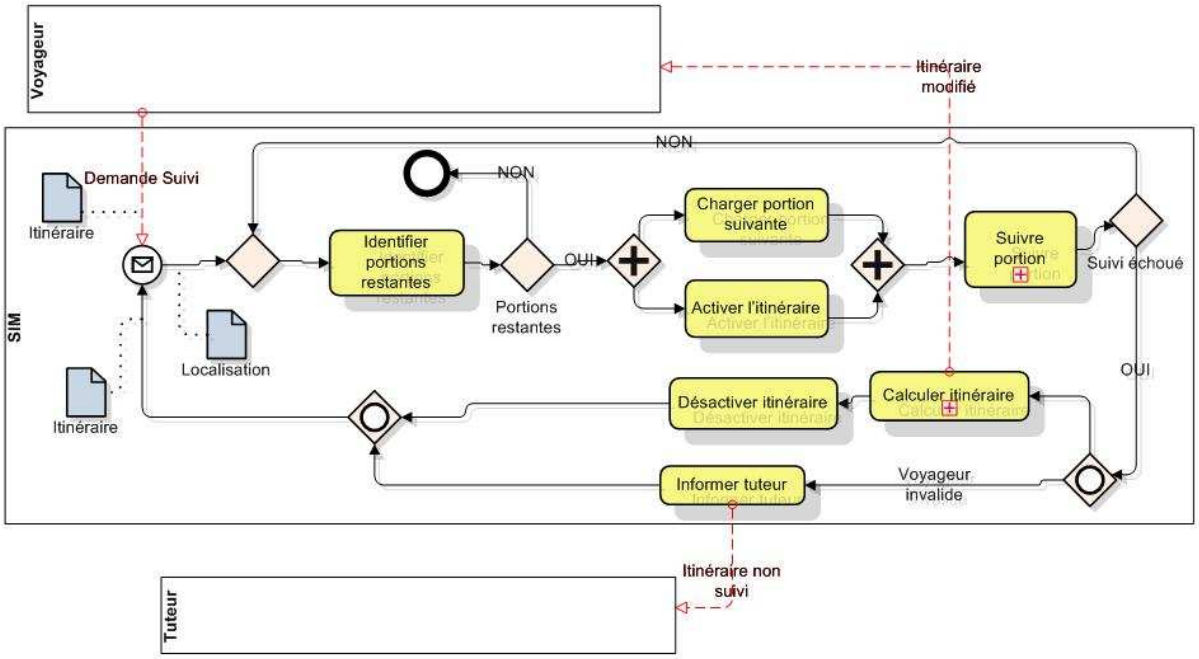


FIGURE 5.11 – Première version du PM "Suivre un itinéraire" obtenu dans la branche fonctionnelle

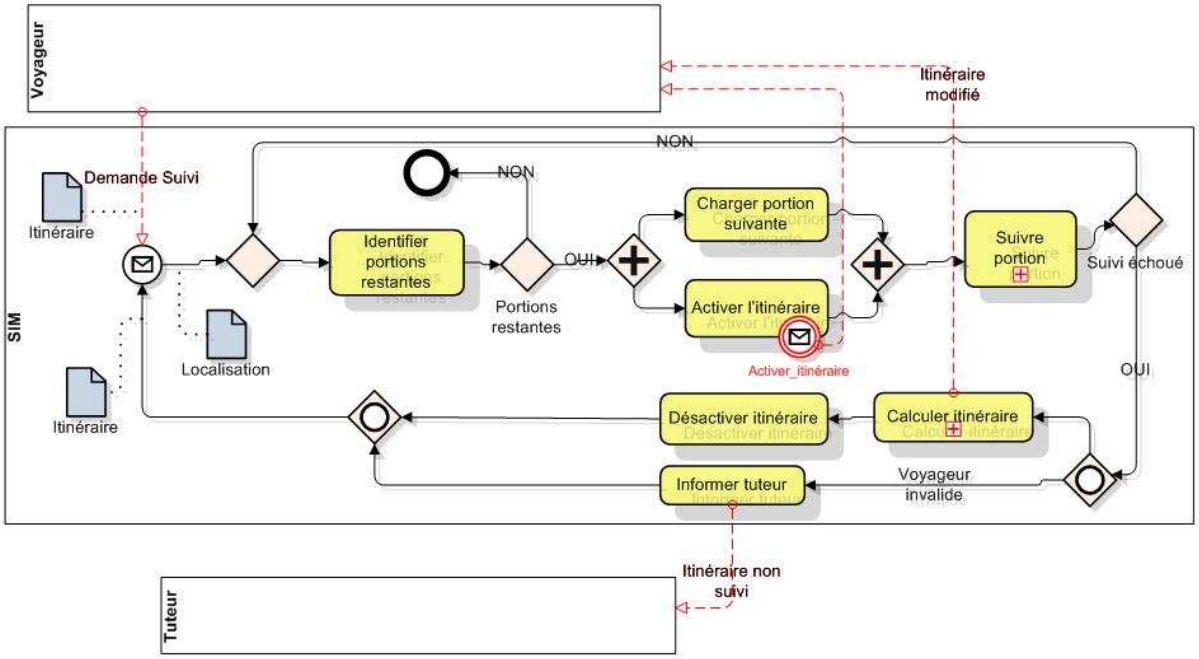


FIGURE 5.12 – Nouvelle version du PM "Suivre un itinéraire"

règles sont ajoutées avec leurs identifiants obtenus dans la démarche ubiquitaire (voir tableau 4.11) comme le montre la figure 5.13. Cette figure présente le PM "Suivre un itinéraire" avec les règles d'adaptation concernées. La règle E-CARe_16 utilisée au début du processus permet de déclencher instantanément le suivi dans le cas d'une personne invalide sans une demande explicite de l'utilisateur. Les règles E-CARe_13, E-CARe_14 et E-CARe_15 permettent d'adapter l'affichage en fonction des

préférences de l'utilisateur ou son handicap (malvoyant ou malentendant) et sont appliquées dans le sous-processus de suivi d'une portion qui nécessite une interaction avec le voyageur. Quant aux règles E-CARe_1, E-CARe_2, E-CARe_3 et E-CARe_4, elle permettent d'adapter le calcul d'itinéraire à la localisation et aux empêchements de la personne.

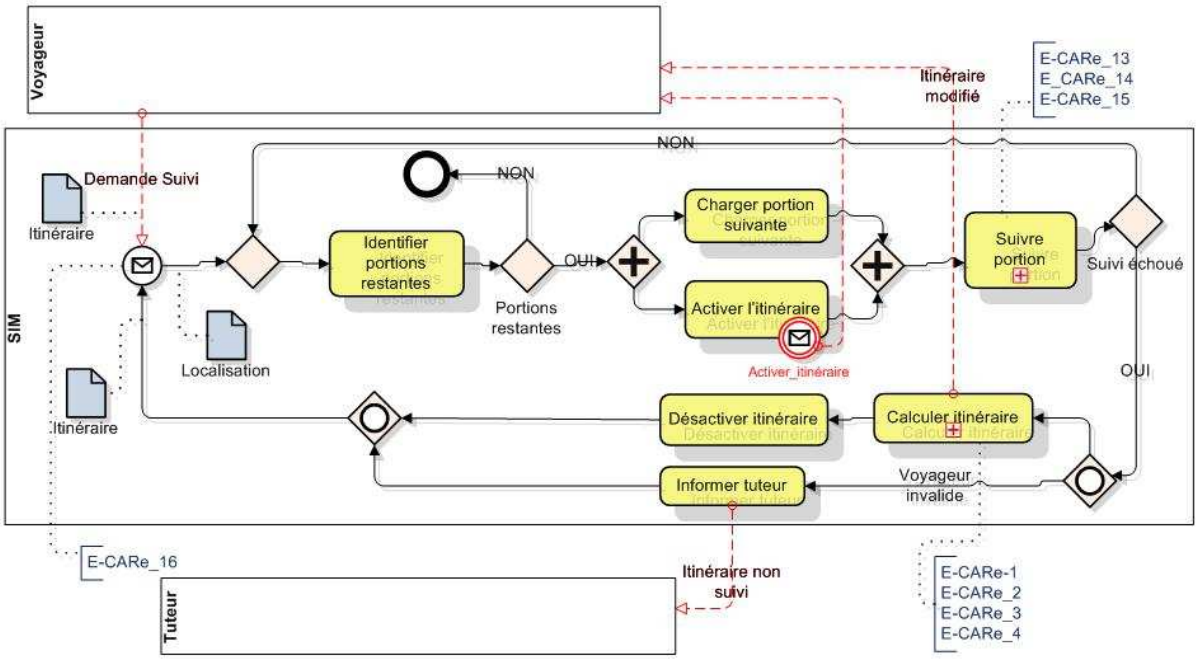


FIGURE 5.13 – PMC "Suivre un itinéraire" enrichi avec les règles ubiquitaires

Le lien réalisé entre les règles et les PM facilite par la suite la conception du moteur de workflow et du moteur de règles qui travaillent ensemble pour la gestion des fonctionnalités du système.

5.4.3 Enrichissement des modèles d'OM

Objectifs et principe

L'objectif de cette étape est d'intégrer les modèles structurels d'analyse de la branche fonctionnelle et les modèles structurels de contexte. En effet, dans la phase d'analyse structurelle, un ensemble d'OM entité est produit et permet de visualiser une entité métier sous la forme d'un paquetage tripartite : interface, maître et rôle.

Ces mêmes OM entités peuvent faire l'objet de fonctionnalités ubiquitaires qui sont traitées dans la branche ubiquitaire où le contexte de ces entités est considéré pour assurer les fonctionnalités d'adaptation et de réactivité au contexte. Ainsi, ces entités font partie du modèle de contexte construit lors de la phase de spécification structurelle du contexte (voir la section 4.5.3 du chapitre précédent).

Par conséquent, le lien conceptuel entre ces entités doit être réalisé en vue d'assurer la cohérence entre les modèles structurels. Nous proposons d'intégrer le modèle de contexte dans les modèles d'OM en vue de produire des modèles structurels unifiés qui faciliteront la conception des bases de données et qui permettent d'attacher à chaque OM entité son contexte. Cette étape a l'avantage de séparer les contextes des entités, maximisant ainsi la sécurité des données personnelles et facilitant toute tâche de maintenance et de modification.

Notre approche consiste à attacher les éléments de contexte d'une entité à la classe "Maître" comme des classes "Partie" de l'OM correspondant. Pour accéder à ces parties, il est nécessaire

d'ajouter des opérations à la classe "Interface" de l'OM. Les relations entre entités dans le modèle de contexte sont exprimées sous forme de liens entre les classes "Rôles" des OM correspondants.

Approche IDM

Cette étape est automatisée grâce à l'usage de transformations de modèles utilisant le modèle de contexte structurel pour enrichir les OM issus de l'analyse structurelle. Dans ce but, deux métamodèles d'OM sont utilisés : le métamodèle des OM Symphony [Hassine, 2005] représenté par la figure 5.14 et un métamodèle représentant des OM enrichis par des concepts de contexte (voir figure 5.15) et utilisant la notion d'élément de contexte comme classe partie de l'OM maître et le concept de relation entre entités comme relation entre un rôle et une interface.

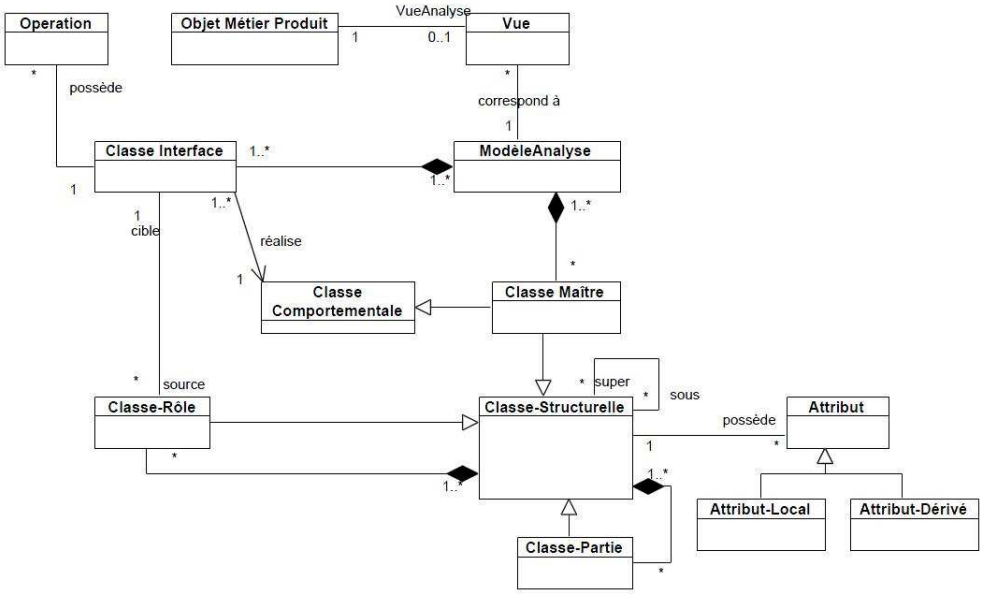


FIGURE 5.14 – Métamodèle des OM Symphony [Hassine, 2005]

L'approche d'enrichissement de modèles appliquée dans cette étape utilise comme entrée le métamodèle d'OM de Symphony et le métamodèle de contexte (voir figure 4.8) et comme sortie le métamodèle d'OM enrichi. Cette approche prend en entrée un modèle de contexte et un ensemble d'OM entité et produit un ensemble d'OM reliés entre eux. Les règles de transformation utilisées sont les suivantes :

- chercher pour entité du modèle du contexte l'OM qui lui correspond,
- chaque propriété de profil de cette entité doit être agrégée comme classe "Partie" à la classe maître de l'OM, si elle n'existent pas déjà,
- ajouter une opération d'accès à la classe "Partie" dans la classe "Interface" de l'OM,
- chaque relation entre entités dans le modèle de contexte est transformée en une relation reliant la classe "Rôle" du propriétaire à la classe "Interface" de la cible,

Exemples

L'OM entité principal dans l'étude de cas du SIT est l'entité "Voyageur" qui est à la fois l'entité centrale dans le modèle de contexte (voir le modèle de contexte de la figure 4.9 obtenu dans la section 4.5.3 du chapitre précédent). Le contexte du voyageur contient les propriétés de profil suivantes :

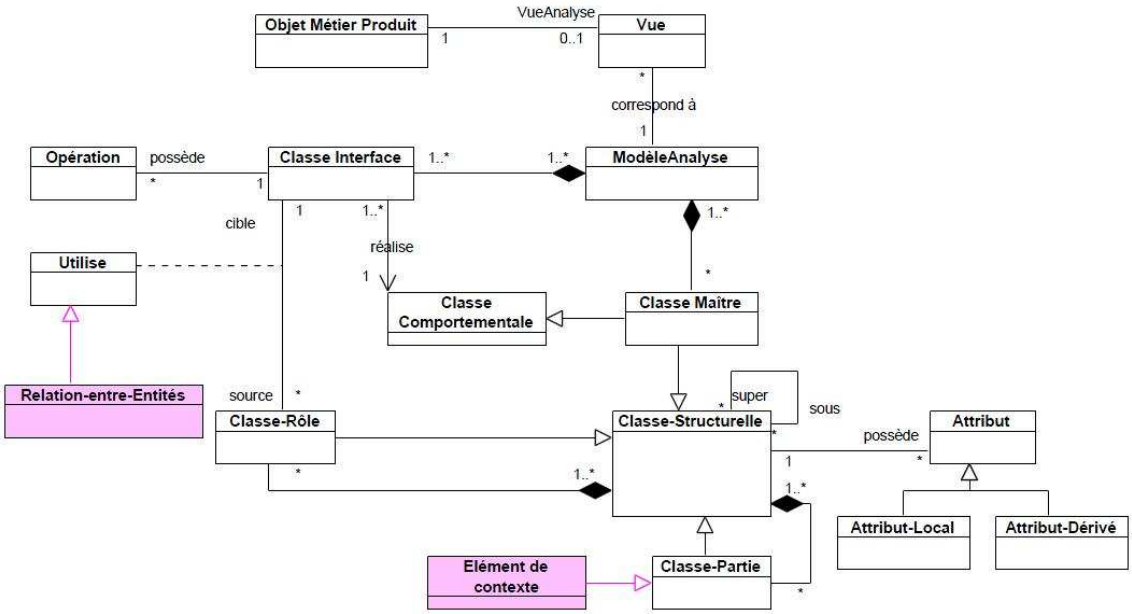


FIGURE 5.15 – Métamodèle des OM Symphony enrichi avec les concepts de contexte

localisation, préférences, empêchements, identité et itinéraires. Cette entité est reliée au dispositif qu'elle utilise et à l'arrêt ou au véhicule dans lequel elle est située. Ces données sont ajoutées à l'OM du voyageur présenté dans la figure 5.5 qui devient désormais un OM enrichi présenté par la figure 5.16.

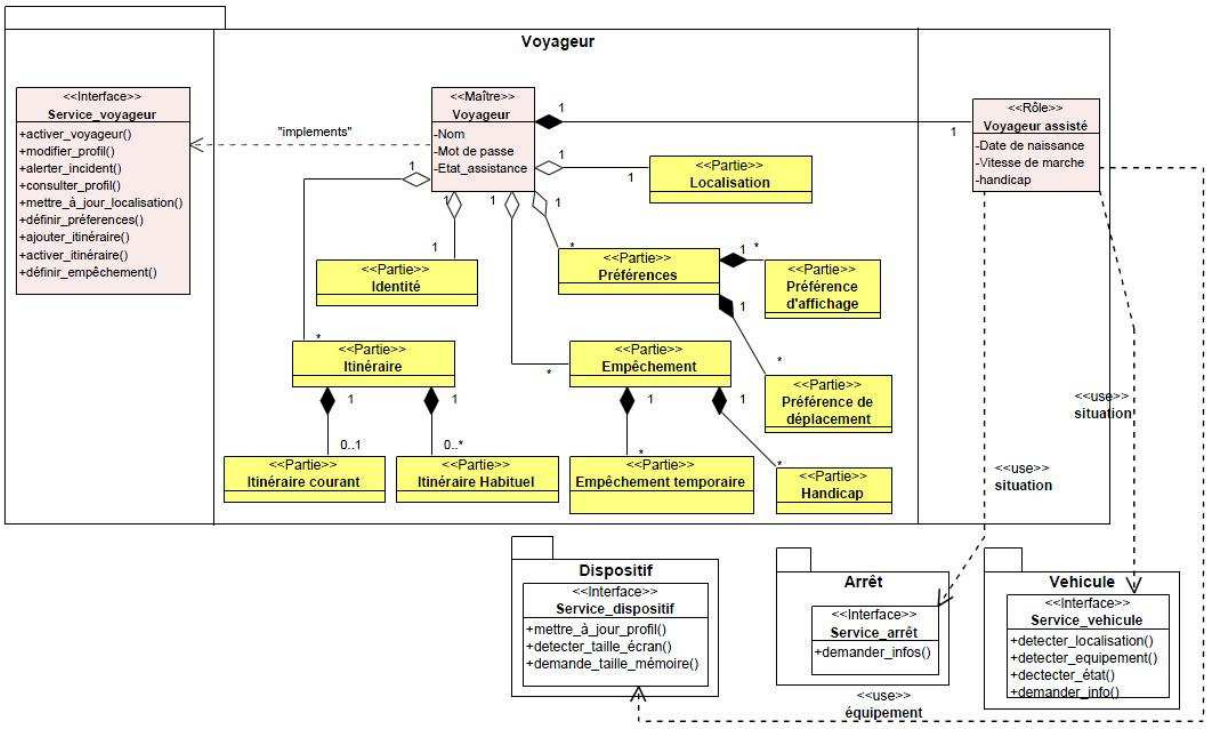


FIGURE 5.16 – OM du voyageur enrichi avec les éléments de contexte

La phase de conception fonctionnelle préliminaire permet ainsi l'intégration des modèles ubiquitaires dans les modèles fonctionnels dynamiques et statiques pour fournir des fonctionnalités ubiquitaires prêtes à la phase de conception. Cependant, les modèles techniques doivent prendre en compte les besoins ubiquitaires pour pouvoir les supporter matériellement et logiciellement.

5.5 Conception technique préliminaire

Cette phase permet de rapprocher les branches ubiquitaire et technique en intégrant les modèles ubiquitaires dans les choix techniques et les architectures du système. En effet, la branche ubiquitaire permet de découvrir les fonctionnalités ubiquitaires du système qui nécessitent des interactions avec l'environnement et avec de nouveaux dispositifs. La prise en compte de ces interactions est importante dans la définition de l'architecture du système et le choix des composants techniques assurant ces interactions. De plus, l'adoption d'une approche orientée évènements nécessite la construction d'une architecture orientée évènements, ce qui n'est possible qu'après la spécification de la cartographie d'évènements.

Pour réaliser ces objectifs, cette phase comporte deux étapes (voir figure 5.17) : la vérification de la cohérence entre les modèles d'évènements et l'architecture matérielle, et la définition du réseau global de traitement des évènements.

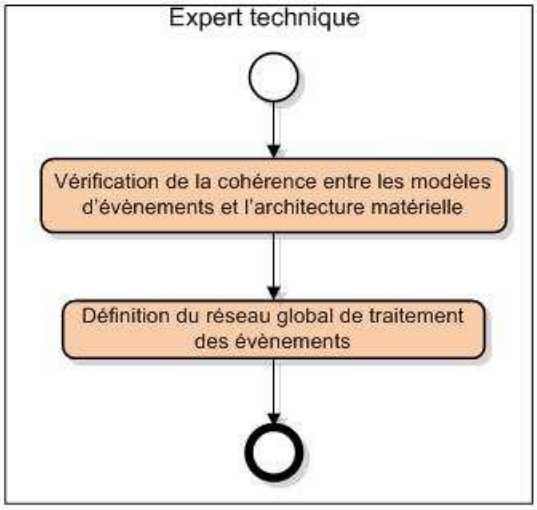


FIGURE 5.17 – Processus de la phase de conception technique préliminaire

5.5.1 Vérification de la cohérence entre les modèles d'évènements et l'architecture matérielle

Objectifs et principe

La cartographie d'évènements obtenue dans la branche ubiquitaire présente les évènements qui servent parfois à notifier certaines ressources externes. Elle présente de plus les sources des évènements qui proviennent dans certains cas de ressources externes. Ainsi, la cartographie d'évènements met en évidence des échanges d'évènements avec des ressources externes qui peuvent être des acteurs, d'autres systèmes ou même des dispositifs tels que les capteurs. Par conséquent, il est nécessaire de vérifier la prise en charge de ces échanges par l'architecture matérielle du système.

La vérification de cohérence entre les modèles évènementiels et l'architecture matérielle doit concerner l'existence de toutes les ressources externes dans l'architecture matérielle et l'existence des liens nécessaires entre les différents composants de l'architecture. Dans le cas où la cohérence n'est pas assurée, il convient de revoir l'architecture matérielle pour ajouter des ressources ou unifier les nomenclatures et ajouter les liens nécessaires.

Approche IDM

Pour vérifier la cohérence entre les modèles d'évènements et l'architecture matérielle, nous adoptons la même approche que dans la section 5.4.1 : l'approche par métamodèle de problème [Bézivin et Jouault, 2006]. Le métamodèle de problème utilisé (voir figure 5.18) permet de modéliser le résultat de la vérification qui peut être "Réussi" ou "Echoué", les ressources absentes et les liens absents.

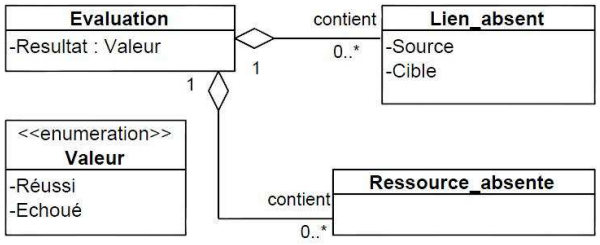


FIGURE 5.18 – Métamodèle de problème pour le résultat de la vérification de cohérence entre architecture matérielle et cartographie d'évènements

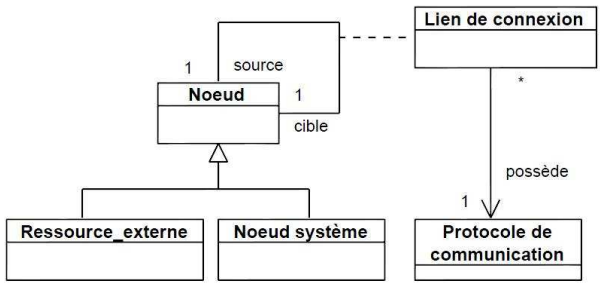


FIGURE 5.19 – Métamodèle de l'architecture matérielle

Pour réaliser cette vérification de cohérence, les deux métamodèles d'entrée sont le métamodèle d'évènements (voir figure 4.16) et le métamodèle d'architecture matérielle (voir figure 5.19). La comparaison entre ces deux métamodèles respecte les règles suivantes :

- chercher tous les évènements externes du modèle d'évènements et récupérer leurs sources,
- vérifier pour chaque source son existence dans le modèle d'architecture matérielle sous le concept "Ressource_externe" et l'existence d'une connexion allant de cette ressource à un nœud système,
- chercher dans le modèle d'évènements toutes les ressources notifiées,
- vérifier, de même, pour chaque ressource son existence dans le modèle d'architecture matérielle sous le concept "Ressource_externe" et l'existence d'une connexion allant d'un nœud système à cette ressource,
- une ressource qui n'existe pas change le résultat de la vérification à "Echoué" et doit être transformée dans le modèle de problème en une "Ressource_absente",

– une connexion qui n'existe pas change le résultat de la vérification à "Echoué" et doit être transformée dans le modèle de problème en un "Lien_absent",

Exemples

La définition de l'architecture matérielle du SIT (voir figure 5.6) montre que le concepteur technique n'a pas considéré des ressources externes comme le service météo et le service trafic dont l'usage a été défini dans la branche ubiquitaire. Ainsi, après la vérification de cohérence le concepteur doit ajouter ces ressources et les relier aux nœuds systèmes appropriés pour cet échange d'évènements.

5.5.2 Définition du réseau global de traitement des évènements

Objectifs et principe

Un réseau de traitement des évènements est une architecture orientée évènements qui met en évidence les échanges d'évènements (voir section 3.2.4). Nous définissons un réseau global de traitement des évènements comme une architecture qui se concentre uniquement sur les échanges d'évènements entre le système et les ressources externes. Les échanges internes ne sont pas considérés. Cette architecture est utile dans la mesure où elle permet de concevoir les liens externes d'un moteur d'évènements avec l'environnement du système.

Pour construire une telle architecture, nous utilisons la cartographie d'évènements de la branche ubiquitaire et l'architecture matérielle de la branche technique. La vérification de cohérence entre ces deux modèles a permis de produire une architecture matérielle en harmonie totale avec les modèles évènementiels. Ainsi, nous pouvons utiliser les liens de l'architecture matérielle et les annoter avec les données évènementielles pour définir un réseau de traitement d'évènements.

Pour ce faire, nous considérons l'application comme une boîte noire et nous ne nous intéressons qu'aux échanges d'évènements avec l'extérieur du système. Nous cherchons pour chaque connexion entre le système et une ressource externe, l'ensemble des évènements qui correspondent à un tel échange. Nous représentons le réseau global de traitement des évènements par un diagramme de communication UML qui permet d'afficher les échanges de messages entre les participants d'un système. L'obtention de cette architecture est automatisée grâce à l'usage de l'IDM.

Approche IDM

Pour produire l'architecture de réseau global, nous utilisons une approche de transformation IDM à partir de la cartographie d'évènements et de l'architecture matérielle. Les métamodèles d'entrée sont le métamodèle d'évènements (figure 4.16) et le métamodèle de l'architecture matérielle (figure 5.19). Le métamodèle de sortie est celui du réseau de traitement d'évènements (figure 5.20).

Pour produire un réseau de traitement des évènements conforme au métamodèle de sortie, les règles de transformation suivantes sont appliquées :

– créer une classe application,
– dans le modèle d'architecture matérielle, chaque "Lien de connexion" ayant pour source une "Ressource_externe" est transformé en un "Lien_entrant" attaché à l'application et la "Ressource_externe" est transformée en une "Ressource" agrégé à ce "Lien_entrant" (si elle n'existe pas déjà),
– Dans la cartographie d'évènements, chaque évènement ayant pour source cette "Ressource_externe" est transformé en un "Evènement" agrégé à ce "Lien_entrant",

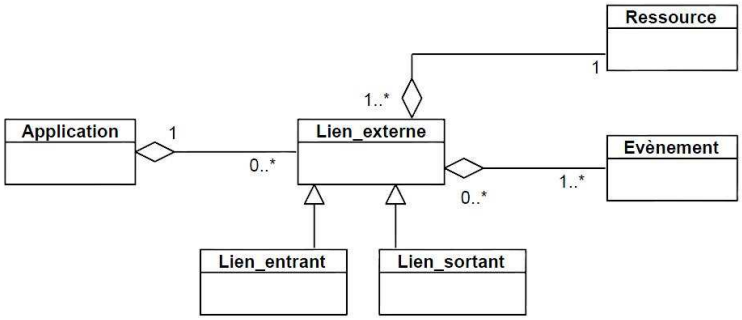


FIGURE 5.20 – Métamodèle du réseau global de traitement d'évènements

- dans le modèle d'architecture matérielle, chaque "Lien de connexion" ayant pour cible une "Ressource_externe" est transformé en un "Lien_sortant" attaché à l'application ; et la "Ressource_externe" est transformée en une "Ressource" agrégée à ce "Lien_sortant" (si elle n'existe pas déjà),
- Dans la cartographie d'évènements, chaque évènement notifiant cette "Ressource_externe" est transformé en un "Evènement" agrégé à ce "Lien_sortant",

Exemples

Le réseau global de traitement d'évènements de l'application d'assistance au voyageur est représenté par la figure 5.21. Ce réseau est un diagramme de communication UML qui utilise les échanges de messages pour représenter les évènements et les participants pour représenter l'application et les ressources qui l'entourent (en pointillés).

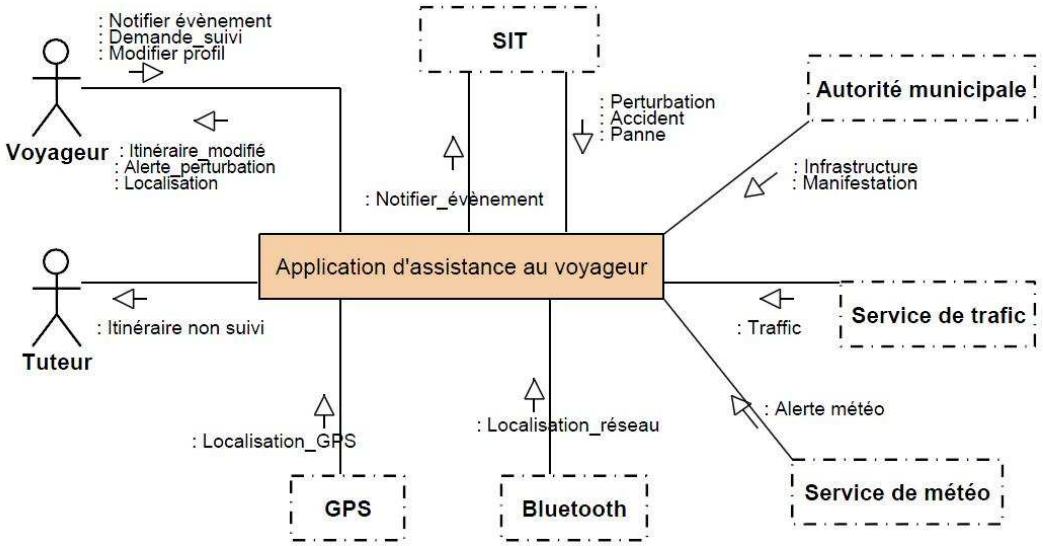


FIGURE 5.21 – Réseau global de traitement d'évènements de l'application d'assistance au voyageur

Ce réseau montre les ressources externes avec lesquelles communique l'application d'assistance, à savoir les services de météo et de trafic, les autorités municipales, le dispositif Bluetooth (qui communique avec les véhicules et les arrêts) et les acteurs qui sont le voyageur et le tuteur.

A l'issue de cette phase préliminaire de conception, les modèles techniques intègrent désormais les choix ubiquitaires et l'approche orientée évènements. Les modèles fonctionnels et les modèles

techniques sont enrichis avec les modèles ubiquitaires. Il est désormais nécessaire de rapprocher ces différents modèles pour produire des modèles de conception.

5.6 Conception du système

Dans une démarche de cycle de vie en "Y", la conception permet la convergence des spécifications fonctionnelles et des choix technologiques de la branche technique.

Dans notre démarche, les phases préliminaires de conception ont permis la convergence de la branche ubiquitaire avec les branches fonctionnelle et technique pour produire des modèles fonctionnels et techniques prenant en compte les caractéristiques ubiquitaires du système. Ainsi, la phase de conception de la démarche E-CARe ressemble à toute phase classique de conception d'une démarche en "Y". Par conséquent, nous présentons dans cette phase uniquement les étapes de conception qui découlent des spécificités des systèmes ubiquitaires.

Notre méthode étant orientée évènements, la conception du réseau de traitement d'évènements est une priorité. De même, la conception des moteurs d'évènements et de règles est essentielle avant le passage à l'implémentation du système. D'autre part, l'usage du contexte nécessite une étape de conception des bases de données contextuelles et la réalisation des liens avec les flux d'évènements du système.

La phase de conception comporte deux étapes définies dans la figure 5.22 et détaillées ci-dessous.

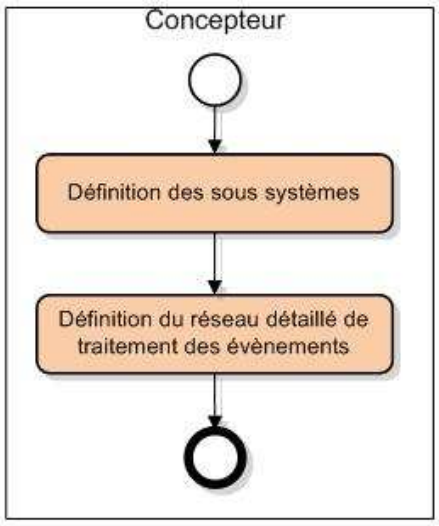


FIGURE 5.22 – Processus de la phase de conception du système

5.6.1 Définition des sous-systèmes

Objectifs et principe

La définition de l'architecture applicative du système permet de définir des sous-systèmes fonctionnels qui sont les nœuds de l'architecture. Cette décomposition fonctionnelle réalisée par le concepteur doit respecter les contraintes organisationnelles de l'entreprise telles que la structure hiérarchique des services et des divisions.

Nous recommandons dans notre approche orientée évènements la considération des échanges d'évènements pendant la décomposition du système. En effet, notre objectif est de créer des composants homogènes, cohérents et faiblement couplés pour faciliter la maintenance et les modifications qui sont nécessaires et inévitables dans le cas des systèmes ubiquitaires qui peuvent évoluer autant du côté fonctionnel que du côté technique.

Ainsi, la décomposition fonctionnelle du système est une responsabilité du concepteur qui doit considérer plusieurs aspects qui sont l'architecture matérielle, les échanges d'évènements, la cohérence fonctionnelle et la distribution et l'usage des données contextuelles.

Chaque sous-système fonctionnel est un ensemble de PM cohérents et fortement couplés ainsi que leurs Objets Métier. Un sous-système possède une architecture technique type qui est celle définie dans la branche technique. Chaque sous-système possède un moteur de règles et une base de données qui lui sont propres pour garantir des objectifs tels que l'autonomie et la confidentialité de chaque sous-système. Un sous-système est ainsi un mini système avec une architecture en 5 modules (présentation, métier, données, communication et contexte) (voir chapitre 3 section 3.1) et communique avec son environnement par des échanges d'évènements qui respectent des règles d'accès définies par le concepteur.

Exemples

Dans l'exemple de l'application d'assistance, il est judicieux de construire trois sous-systèmes prenant en charge les PM définis dans la section 4.4.2 (tableau 4.5) du chapitre précédent lors de la spécification intentionnelle des besoins :

- la gestion et la notification des perturbations : sous-système composé par les trois PM "Notifier une perturbation", "Notifier un évènement" et "Résoudre une perturbation",
- l'assistance et l'information des déplacement : sous-système composé par les PM "Calculer un itinéraire", "Assister un itinéraire", "Rappeler un déplacement" et "Emettre une alerte",
- la gestion des données personnelle : sous-système composé par le processus de support "Gestion de profil".

5.6.2 Définition du réseau détaillé de traitement des évènements

Objectifs et principe

Un réseau détaillé de traitement des évènements (EPN -Event Processing Network) est une architecture orientée évènements qui considère aussi bien les échanges d'évènements externes que les échanges d'évènements internes. La structure d'un réseau détaillé est représentée par le métamodèle de la figure 5.23 qui montre un EPN comme un ensemble de Event Processing Units (EPU) possédant des PM, des OM et des règles d'adaptation. Les liens entre EPU permettent des échanges d'évènements respectant des règles d'accès.

La définition des sous-systèmes réalisée dans l'étape précédente a permis de définir l'architecture interne du système. Il reste dans cette étape à annoter les liens entre les sous-systèmes par les échanges d'évènements dans un diagramme de communication UML comme dans le cas d'un réseau global de traitement des évènements (voir figure 5.21).

Chaque système possède sa propre architecture avec un ensemble de règles d'adaptation à exécuter dans le moteur de règles. Cette architecture est en fait une extension du réseau global de traitement d'évènements. Ainsi, il convient dans cette étape de se concentrer sur les échanges internes des évènements et l'identification des sous-systèmes communiquant avec les ressources externes.

Cette étape doit de plus identifier pour chaque sous-système l'ensemble des règles d'adaptation correspondant à l'ensemble des PM pris en charge par le sous-système. Si la spécification des besoins

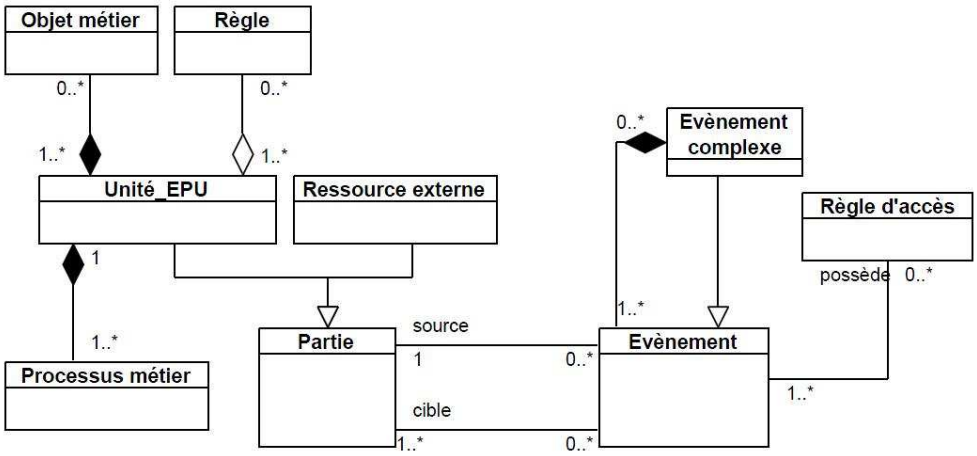


FIGURE 5.23 – Métamodèle du réseau détaillé de traitement d'évènements

a permis d'attribuer des contraintes d'accès à certaines données évènementielles, alors il convient d'ajouter ces règles d'accès à l'évènement et à la connexion concernés.

Pour faciliter la tâche au concepteur cette étape peut être réalisée automatiquement par l'adoption d'une approche IDM sauf pour l'ajout des règles de contrôle d'accès qui est fait manuellement.

Approche IDM

Pour produire un réseau détaillé de traitement des évènements, il faut considérer les différents sous-systèmes, leurs échanges d'évènements et leurs règles de traitement des évènements. Ainsi, il convient d'utiliser le modèle de sous-systèmes et les PM qui les composent pour identifier pour chaque sous-système les différents traitements d'évènements réalisés et les évènements en entrée et en sortie.

Par conséquent, nous utilisons les modèles de sous-systèmes, les modèles de règles et les modèles d'évènements. En effet, en connaissant les PM d'un sous-système, il est possible de déterminer les règles qui sont utilisées. Ensuite, il est facile de déterminer les évènements utilisés ou produits dans ces règles. Les métamodèles d'entrée sont donc le métamodèle de sous-systèmes de la figure 5.24, le métamodèle de règles de la figure 4.11, le métamodèle d'évènements de la figure 4.16 et le métamodèle d'OM de la figure 5.15. Le métamodèle de sortie est celui du réseau détaillé de traitement des évènements (voir figure 5.23).

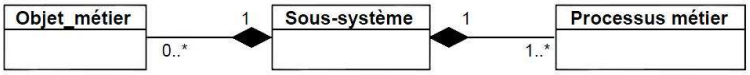


FIGURE 5.24 – Métamodèle de sous-systèmes

Les règles de transformation appliquées sont les suivantes :

- chaque sous-système devient un EPU en conservant l'ensemble de ses PM et ses OM dans le réseau EPN détaillé,
- chaque règle utilisée dans un PM (dans le modèle de règles) relié à un sous-système (modèle de sous-système) est agrégée à l'unité EPU correspondante à ce sous-système,
- chaque "Ressource" dans le modèle d'évènements est transformée en une "Ressource externe" dans le modèle EPN,
- chaque "Source" de type Externe ou Utilisateur dans le modèle d'évènements est transformée en une "Ressource externe" dans le modèle EPN,

- chaque "Evènement simple" du modèle d'évènements est transformé en un "Evènement" dans le modèle EPN et il est relié à sa source :
  - si sa source dans le modèle d'évènements est un PM (c'est-à-dire de type "Métier") alors relier l'évènement avec l'"EPU" possédant ce PM,
  - si la source est de type "Utilisateur" alors relier l'évènement avec la "Ressource" externe représentant l'utilisateur,
  - si la source est de type "Externe" alors relier l'évènement avec la ressource externe,
- l'évènement est relié à ses cibles :
  - si l'évènement permet de notifier une "Ressource" dans le modèle d'évènements alors il est relié à la "Ressource externe" représentant la ressource notifiée,
  - si l'évènement permet de déclencher une "Règle" dans le modèle d'évènements alors il a pour cibles les EPU possédant cette règle,
  - si l'évènement permet de mettre à jour un "Elément de contexte" alors il a pour cible L'EPU possédant l'OM contenant cet élément de contexte dans ses classes "Partie",
- un "Evènement complexe" dans le modèle d'évènements est transformé en un "Evènement complexe" dans le modèle EPN et sa composition en "Evènement"(s) est conservée,
- l'évènement complexe est relié à une source qui est l'EPU contenant les évènements qui le compose (comme source ou cible) et il est relié à des cibles :
  - si l'évènement complexe permet de notifier une "Ressource" dans le modèle d'évènements alors il est relié à la "Ressource externe" représentant la ressource notifiée,
  - si l'évènement complexe permet de déclencher une "Règle" dans le modèle d'évènements alors il a pour cibles les EPU possédant cette règle,
  - si l'évènement complexe permet de mettre à jour un "Elément de contexte" alors il a pour cible L'EPU possédant l'OM contenant cet élément de contexte dans ses classes "Partie",

Exemples

Le réseau détaillé de traitement des évènements obtenu dans le cas de l'application d'assistance au voyageur est représenté par la figure 5.25. Dans cette architecture, trois EPU sont utilisées, elles correspondent aux trois sous-systèmes identifiés dans l'étape précédente. Les échanges internes et externes des évènements sont représentés.

Le diagramme de communication ne représente pas une vue complète d'un réseau de traitement des évènements puisque la composition des EPU en fonction des OM des PM et des règles E-CARe n'est pas représentée. Pour ne pas surcharger le diagramme de communication nous suggérons l'ajout d'une vue complémentaire représentant ces détails. La figure 5.26 présente les détails de la composition de l'EPU d'assistance et d'information des déplacements.

5.7 Implémentation

La phase de conception permet la préparation des modèles pour la phase d'implémentation. Lors de l'implémentation, les modèles conceptuels sont transformés en des modèles spécifiques à la plateforme et l'ensemble du système est configuré et codé. Nous recommandons l'organisation de l'implémentation selon les différents sous-systèmes : pour chaque sous-système, coder et développer l'ensemble des modules (métier, données, présentation, communication et contexte) ; ensuite, organiser le tout dans un système complet et cohérent ; enfin, intégrer l'ensemble du système dans l'architecture physique et les différents composants matériels. La phase d'implémentation est composée par les étapes définies dans la figure 5.27 et détaillées ci-dessous.

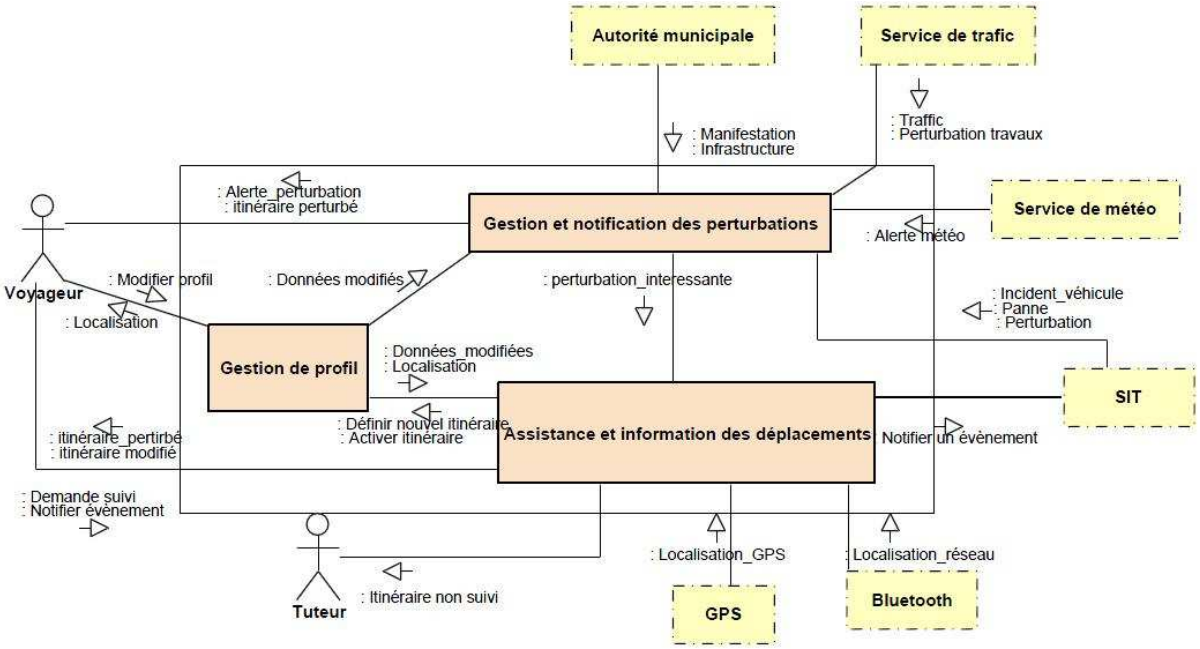


FIGURE 5.25 – Réseau détaillé de traitement d'évènements de l'application d'assistance au voyageur

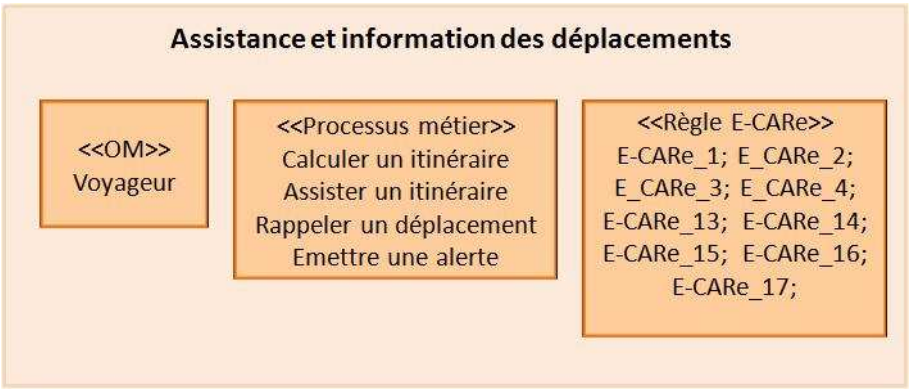


FIGURE 5.26 – Composition de l'EPU "Assistance et information des déplacements"

5.7.1 Transformation des modèles métier

Les modèles des PM sont décrits en BPMN. Pour les exécuter, il faut les transformer en BPEL (Business Process Execution Language) qui est le langage standard utilisé dans plusieurs moteurs de workflows. Le mapping de BPMN à BPEL est décrit dans les spécifications de BPMN [6] et il est supporté par la plupart des outils de modélisation des PM. Ainsi, cette tâche peut être réalisée automatiquement avec le soin de vérifier l'implémentation des règles métier. De plus, les règles d'adaptation doivent faire l'objet d'exceptions liant l'exécution métier au moteur de règles responsable de l'évaluation des conditions contextuelles et de la prise de décision.

5.7.2 Implémentation des bases de données

Dans notre approche orientée évènements où l'usage du contexte est basé sur les échanges de flux d'évènements, et pour permettre un meilleur usage du contexte et faciliter l'accès à ses données, il

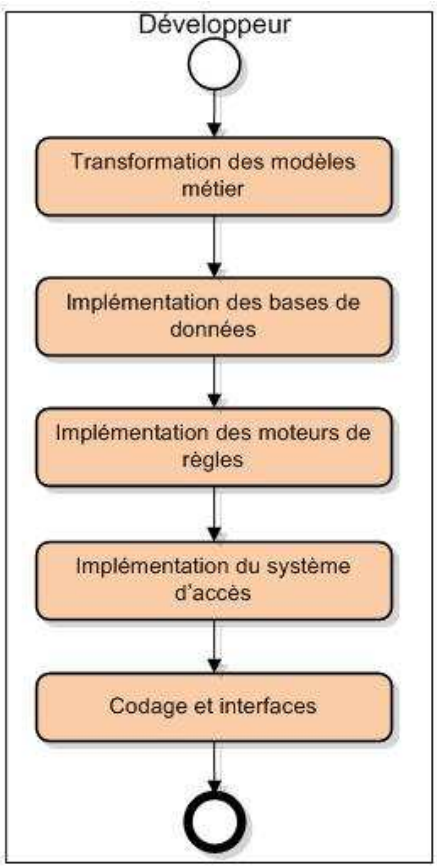


FIGURE 5.27 – Processus de la phase d'implémentation

est pertinent que les données de contexte soient considérées dans la conception des bases de données. Pour cela, plusieurs tâches sont nécessaires.

Réalisation du modèle conceptuel de données

La phase de conception fonctionnelle préliminaire a permis la construction des OM enrichis avec les données contextuelles. Les modèles d'OM doivent désormais être transformés en des modèles conceptuels prenant en compte les choix de conception tels que le typage et la visibilité des attributs, le placement des opérations et les structures de données. Cette tâche est décrite en détails dans la démarche Symphony [Hassine, 2005].

Nous ajoutons de plus que la navigabilité des relations entre entités doit être précisée. Les opérations qui permettent l'accès aux données des entités doivent être définies avec leur visibilité sachant que ces données comprennent les informations contextuelles sous forme de classes "Partie".

Définition du modèle relationnel

Comme toute démarche d'ingénierie la phase de conception doit comprendre la transformation des modèles de données en des modèles relationnels pour préparer l'implémentation des bases de données.

L'architecture technique établie dans la branche technique de la démarche permet la définition des composants de stockage de données. L'étude des sous-systèmes permet de concevoir les bases de données en définissant les OM à inclure dans chaque bases de données ainsi que les besoins de

duplication de données. Pour concevoir les bases de données contextuelles, il faut transformer ces modèles d'OM en des modèles relationnels de données.

Plusieurs approches et outils supportent les transformations d'un modèle de classes à un modèle relationnel nous ne reviendrons donc pas sur cet aspect ici.

De plus, la mise à jour des données est assurée par les évènements du modèle d'acquisition du contexte. Il faut donc relier les modèles relationnels aux évènements de mise à jour par l'intermédiaire de triggers par exemple.

Les modèles relationnels de données doivent être implémentés dans les bases de données correspondantes. Les triggers et les procédures doivent être implémentés. De plus, les mécanismes de duplication et de partage de données doivent être mis en place.

5.7.3 Implémentation des moteurs de règles

Les fonctionnalités ubiquitaires de notre système sont exprimées avec des règles E-CARe comportant un évènement de déclenchement, une condition sur le contexte et une action d'adaptation et de réaction au contexte. Chaque règle est décrite dans la branche ubiquitaire avec un diagramme de classes. Les règles, pour pouvoir s'exécuter, doivent être intégrées dans un moteur d'évènements qui permet la détection des évènements simples ou complexes par l'usage des requêtes continues. Chaque moteur de règles a son propre langage pour exprimer les règles. Nous proposons dans la phase de conception de traduire ces règles avec le langage correspondant au moteur choisi dans la branche technique. Ces règles exprimées avec le langage d'exécution sont implémentables facilement en les intégrant dans le code du moteur. De plus, elles permettent de lier les bases de données aux moteurs de règles lors de l'évaluation des conditions.

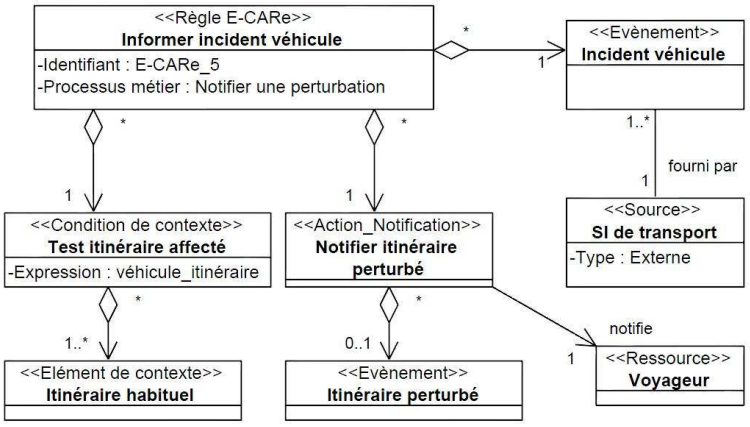


FIGURE 5.28 – Exemple d'une règle E-CARe implémentée dans le moteur de règles Esper

Dans le cadre de notre exemple, nous avons expérimenté plusieurs moteurs d'évènements afin de tester la faisabilité de nos propositions. Parmi l'ensemble des moteurs d'évènements étudiés dans l'annexe B notre choix technique a porté sur Esper [11] qui est le seul à permettre l'accès aux bases de données externes pour l'exécution des requêtes. La figure 5.28 présente un exemple d'une règle E-CARe décrite avec un diagramme de classes dans la phase de spécification évènementielle du contexte (voir section 4.6.1). Cette règle permet la détection des évènements concernant les incidents des véhicules, et leur comparaison avec les données des itinéraires des voyageurs et l'information des voyageurs concernés par cet incident le cas échéant. Cette règle est traduite ci-dessous avec le langage Event Processing Language (EPL) utilisé dans Esper. L'accès à la base de données du profil est fait explicitement dans la requête.

```
select itineraire_name, temps
from Incident_vehicule as e
sql : ProfileDB [' select v, itineraire_name, temps
from itineraire.Vehicule as v, itineraire.name as itineraire_name,
itineraire.temps_depart as temps ']
where e.vehicule_Id = v
```

Les différents moteurs de règles doivent être développés en codant l'ensemble des évènements et en intégrant les procédures correspondantes à la détection des évènements complexes. Les règles exprimées sous forme de requêtes sont intégrées dans le code des listeners. Par exemple, dans Esper les évènements sont exprimés comme des classes JAVA et les requêtes sont intégrées dans les méthodes JAVA. Une approche par "Tissage" aurait certainement permis une plus grande lisibilité et maintenance du code produit.

5.7.4 Implémentation du système d'accès

D'après l'architecture logique définie dans le chapitre 3, le module de communication est une composante primordiale de chaque SI ubiquitaire. Dans ce module, il est nécessaire de contrôler les accès et les échanges de données entre les sous-systèmes et les ressources externes. Les règles d'accès gérant ces communications ont été définies dans le réseau détaillé de traitement des évènements. Ces règles ont la structure d'une règle E-CARe puisque dans le cas d'un SI ubiquitaire, le contexte est un élément essentiel dans les décisions de contrôle d'accès. A ce stade, ces règles sont de la forme Évènement - Condition Contextuelle - Action. Un exemple d'une règle de contrôle d'accès est fourni ci-dessous.

Évènement : demande informations du véhicule
Condition : véhicule utilisé dans l'itinéraire actuel
Action : autoriser l'accès à "Etat_véhicule" et "Equipement_Vehicule"

Le langage standard de définition des règles d'accès est le langage eXtensible Access Control Markup Language (XACML) [21]. Ainsi, les règles de contrôle d'accès doivent être transformées et écrites avec ce langage qui est supporté par plusieurs plateformes de développement. Il est possible d'automatiser cette étape par l'usage d'une approche IDM en transformant les règles E-CARe de contrôle d'accès en des fichiers XML respectant le langage XACML. Les règles d'accès exprimées avec XACML sont implémentées dans un système d'accès qui doit être configuré et lié aux bases de données et aux composants métier du système.

5.7.5 Codage et interface

Le codage de l'ensemble du système doit être réalisé et les interfaces doivent être construites en transformant les Objets Interactionnels définis dans la branche fonctionnelle en des implémentations d'interfaces.

5.8 Synthèse

Ce chapitre présente la démarche complète E-CARe qui permet de spécifier les besoins fonctionnels, ubiquitaires et techniques et de les intégrer pour la conception d'un SI ubiquitaire. En effet, les modèles fonctionnels du système présentent les métiers et les fonctionnalités ubiquitaires qui leur

sont attachées. Ces fonctionnalités sont étudiées séparément pour définir les modèles structurels et évènementiels du contexte et déduire les besoins de communication à l'intérieur du système et avec son environnement. Ces besoins de communication doivent correspondre aux spécifications architecturales et techniques du système. De plus, l'usage de composants techniques supportant la gestion et l'acquisition du contexte doit être prévu et mis en place dans l'architecture logicielle du système.

La démarche proposée met en œuvre deux phases de conception préliminaire qui permettent d'intégrer les modèles ubiquitaires dans les modèles fonctionnels et techniques. Ceci prépare la mise en œuvre d'une phase de conception classique qui permet de plus de supporter la gestion et la manipulation des évènements dans une architecture de traitement des évènements.

Cette démarche permet de guider le travail de développement, de faciliter les tâches et de garantir l'organisation et le partage de travail entre les développeurs de différentes spécialités. L'usage d'outils et de techniques issus de l'approche IDM favorise la prise en charge de certaines tâches lourdes et volumineuses.

Une validation de cette démarche est présentée dans le chapitre suivant qui présente des expérimentations réalisées auprès de concepteurs d'applications ubiquitaires. La réalisation d'un prototype d'une application d'assistance au voyageur renforce le cas d'étude illustrant les différentes phases de la démarche et justifie l'usage de composants techniques ubiquitaires et leur influence sur la spécification des besoins fonctionnels et ubiquitaires.

Chapitre 6

E-CARe : validation et expérimentations

Ce chapitre a pour objectif de valider la faisabilité et l'utilité de certains des concepts mis en œuvre dans la démarche E-CARe. Nous avons axé les expérimentations sur 2 aspects :

– les aspects conceptuels d'une part,

– les aspects architecturaux de l'approche orientée évènements d'autre part.

Concernant les aspects conceptuels, notre démarche doit pouvoir s'appliquer avec tout type de SI ubiquitaire. Ainsi, il est important de valider la généricité et la complétude des concepts de base utilisés dans cette démarche pour pouvoir les instancier et les utiliser pour toute application. En particulier, les métamodèles de contexte doivent être suffisamment génériques et complets pour être utilisables dans tout type de système ubiquitaire et pour couvrir toutes les facettes contextuelles et évènementielles de tels systèmes. La validation des métamodèles passe par la réalisation d'expérimentations auprès de développeurs d'applications ubiquitaires manipulant quotidiennement la notion de contexte. L'expérimentation de ce métamodèle a permis l'identification de points de faiblesse et de force liés à la complétude et la généricité du métamodèle. Ces critiques et remarques ont été utilisées pour produire un métamodèle générique.

D'un point de vue architectural, l'approche orientée évènements exige l'usage d'une architecture de réseau de traitement des évènements avec des sous-systèmes indépendants utilisant des moteurs de traitement des évènements pour des objectifs ubiquitaires. Les tests réalisés se sont déroulés en deux temps différents en construisant deux prototypes d'applications d'assistance au voyageur. Le premier prototype permet de tester l'usage des règles ubiquitaires et les besoins d'adaptation avec le moteur d'évènements Esper. Le deuxième prototype a pour objectif de valider l'approche orientée évènements par communication Bluetooth entre systèmes indépendants.

6.1 Validation du métamodèle de contexte

Le métamodèle de contexte expérimenté est le métamodèle de couplage (voir figure 6.1) [Ben Cheikh *et al.*, 2010c] présenté en détail dans l'annexe A. Il permet de décrire le contexte sous forme de deux vues complémentaires : une vue structurelle et une vue évènementielle. L'objectif du métamodèle de contexte proposé est de supporter une démarche de développement de SI ubiquitaires. Ce métamodèle doit donc pouvoir être utilisé par des acteurs de différentes spécialités (analystes métier, experts génie logiciel, experts IHM, concepteurs et développeurs). Ainsi, il est nécessaire de tester l'usage de ce métamodèle pour valider d'une part, qu'il est compréhensible et utilisable par ces acteurs et d'autre part, qu'il est générique et complet pour s'appliquer à tout type d'application

ubiquitaire.

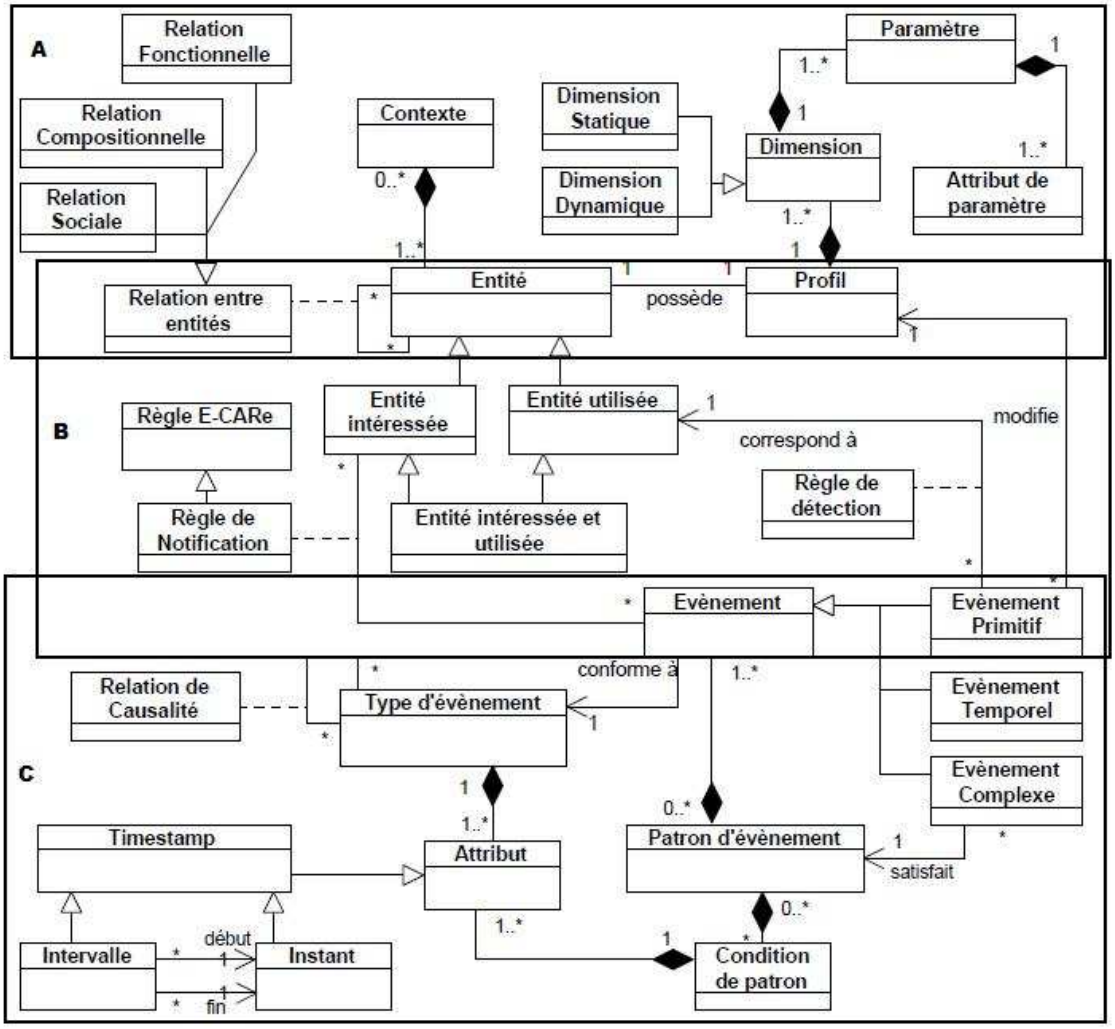


FIGURE 6.1 – Métamodèle de couplage entre contexte et évènements

Nous avons donc réalisé une série d'expérimentations sur les deux vues du métamodèle de couplage de façon séparée que nous détaillons ci-dessous.

6.1.1 Contexte des expérimentations

Afin de valider et évaluer le métamodèle proposé auprès d'utilisateurs potentiels, nous avons mis en place une expérimentation réalisée en deux sessions auprès de chercheurs du Laboratoire Informatique de Grenoble (LIG) issus de différentes disciplines informatiques.

Pour mettre en œuvre cette expérience, nous avons collaboré avec Mme Nadine Mandran, ingénieur méthode et qualité de la plateforme Marvelig. Marvelig [13], plateforme d'expérimentations scientifiques du LIG, a pour but de capitaliser les prototypes réalisés par les différentes équipes de recherche, de mutualiser les potentiels de chaque équipe, de communiquer sur les produits de la recherche du laboratoire et de mettre en œuvre des méthodes d'évaluation de concepts et d'outils.

6.1.2 Hypothèses des expérimentations

Les hypothèses des expérimentations sont les suivantes :

- *H1 : les métamodèles sont compréhensibles.* Il s'agit pour cette hypothèse de tester la compréhension des différentes métaclasses et des relations qui existent entre elles. Ceci est fait à deux niveaux : par le suivi du déroulement des expérimentations et des questions posées par les sujets et par l'évaluation des résultats du travail réalisé par les sujets.
- *H2 : les métamodèles sont utilisables.* Il s'agit de recueillir l'opinion des sujets sur ces métamodèles. L'utilisabilité définit « le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié » [ISO-9241-11, 1998].
- *H3 : les métamodèles sont génériques et complets.* Il s'agit de tester l'utilisation des métamodèles sur un exemple d'application et de recueillir l'avis des sujets sur la possibilité d'utiliser ces métamodèles sur d'autres domaines et la couverture de tous les concepts nécessaires pour la modélisation du contexte.

6.1.3 Description du protocole

L'expérimentation a été réalisée auprès de 11 experts (doctorants, post-doctorants et enseignants-chercheurs) en systèmes ubiquitaires. Ces experts appartiennent à plusieurs équipes du LIG (Sigma, Adèle, Prima, IIHM et Metah) de diverses spécialités : ingénierie des SI, intergiciels et réseaux de capteurs, génie logiciel, gestion des PM, interaction homme-machine et apprentissage humain. Le tableau 6.1 résume le profil des sujets.

TABLE 6.1 – Profil des sujets

Nombre	Moyenne d'âge	Sexe	Fonctions	Spécialités
11	32 ans	8 hommes, 3 femmes	8 doctorants, 2 enseignants-chercheurs et 1 post-doctorant	2 IHM, 2 génie logiciel, 2 ingénierie de SI, 2 gestion des PM, 2 intergiciels et réseaux de capteurs et 1 apprentissage humain

Cette expérimentation est organisée sous la forme de focus group. Le focus group, ou groupe focalisé, est une méthode qualitative permettant le recueil d'informations. Il s'agit d'un groupe de discussion semi-structuré, modéré par un animateur neutre, qui a pour but de collecter des informations sur un nombre limité de thèmes définis à l'avance [APES, 2004] .

Deux questionnaires ont été distribués :

- le premier questionnaire a été distribué avant de commencer l'expérience, il s'agissait de connaître les pratiques des sujets en matière de développement d'applications ubiquitaires et de modélisation du contexte (voir Annexe C.1) ;
- le second questionnaire a été distribué à la fin de l'expérience pour mesurer l'opinion des sujets quant à la compréhension des métamodèles, leur généricité et leur complétude sous forme de points forts et points faibles (voir Annexe C.2).

De plus, un ensemble d'exercices a été réalisé afin de tester les hypothèses que nous souhaitions vérifier. Les sections suivantes décrivent ces hypothèses, les exercices réalisés et les résultats obtenus pour chaque point à tester.

6.1.4 Déroulement des expérimentations

Un exemple d'application ubiquitaire est fourni aux sujets, il s'agit de la conception d'une application d'assistance au voyageur avec l'ensemble des fonctionnalités de l'application. Un ensemble de besoins d'adaptation au contexte est également fourni. Ces exercices sont consultables dans l'Annexe C.3. Les sujets doivent construire un modèle de contexte pour cette application en instanciant le métamodèle proposé, sous forme d'un diagramme de classes (usage de Post-it pour les classes et de feutres pour dessiner les liens entre classes).

FIGURE 6.2 – Photo prise au cours de l'expérience

Le deuxième exercice consiste à donner une cartographie des évènements de l'application par instanciation du métamodèle d'évènements fourni. A la fin de chaque exercice, une discussion permet de recenser les difficultés des exercices et la prise de position de chaque sujet par rapport aux métamodèles proposés. La figure 6.2 montre une illustration du travail de modélisation avec le matériel expérimental utilisé.

6.1.5 Résultats

Le questionnaire des pratiques a montré que les sujets possèdent différentes perceptions de la notion du contexte. En effet, pour certains, le contexte est un ensemble de caractéristiques dépendantes du domaine d'application (sujet 6 et sujet 11) alors que pour d'autres (sujet 8), le contexte est l'ensemble de faits pertinents pour la réalisation d'une tâche. D'autres sujets donnent des définitions plus proches de leur spécialité. Par exemple, pour les spécialistes IHM, le contexte est un triplet (plateforme, dispositif, utilisateur) (sujet 1 et sujet 3) et pour un spécialiste des réseaux de capteurs, le contexte est un ensemble de données/informations relatives à l'environnement d'exécution des requêtes/applications dans les parcs de capteurs.

En ce qui concerne les pratiques de modélisation du contexte, 10 sujets sur 11 avouent avoir rencontré des difficultés concernant l'identification des propriétés et des attributs pertinents du contexte,

la gestion de la temporalité et la dynamicité des données du contexte, le bornage des éléments de contexte et de prise en charge de leur hétérogénéité et de leur complexité et l'identification des besoins utilisateurs et des différentes réactions de l'application au contexte.

Concernant l'usage de métamodèles pour modéliser le contexte, 10 sujets sur 11 trouvent cette idée intéressante et ont manifesté une curiosité à propos de ce sujet.

Au cours de l'expérience, avec les définitions fournies des métaclasses, le travail de modélisation s'est déroulé normalement sans grande intervention de notre part, ce qui a été confirmé par le questionnaire final de Points Forts/Points Faibles qui montre que la plupart des concepts étaient clairs pour tous les sujets. Le travail de modélisation a été réussi pour 10 sujets sur 11 qui ont identifié les entités principales avec leurs profils et leurs relations entre eux : le voyageur, les véhicules de transport et les arrêts. Mais 3 sujets ont considéré de plus que les lignes du réseau et les trajets étaient des entités. Concernant le sujet 2 qui n'a pas réussi l'exercice, il n'était pas familier avec l'usage des diagrammes de classes et a interprété différemment la notion d'entité.

Concernant l'exercice de modélisation des évènements, les sujets ont rencontré certaines difficultés concernant l'identification des sources des évènements et le lien entre les évènements et les données de contexte. De plus, les deux sujets 1 et 3 ont eu besoin d'exprimer des relations entre évènements, ce qui n'est pas supporté par le métamodèle proposé. Cependant, la classification des évènements (utilisateur, métier et externe) a été claire pour tout le monde.

Lors de la discussion et dans le questionnaire, les sujets ont exprimé des critiques des concepts du métamodèle. En effet, ils trouvent que la notion de relation entre entités est intéressante mais doit être supportée par d'autres concepts qui clarifient son usage (sujets 4, 5 et 8) pour préserver la généricité du métamodèle. Pour les spécialistes IHM (sujets 1 et 3), le métamodèle mélange des notions de contexte avec les données du domaine d'application puisque pour cette spécialité, la conception utilise à la fois un modèle de domaine et un modèle de contexte. Le sujet 9 fait ressortir un besoin de relier certains éléments de contexte ensembles pour les réutiliser avec deux entités différentes.

L'évaluation de la généricité du métamodèle par les sujets montre que 9 sujets affirment que les métamodèles sont assez génériques. Par contre, le sujet 9 affirme qu'il est difficile de juger la généricité, alors que le sujet 8 pense que l'usage de la relation sociale entre entités gène à la généricité : « on force la conception, ceci restreint le domaine ». En ce qui concerne la complétude, 6 sujets sur 11 trouvent que les métamodèles sont complets alors que les autres pensent qu'il est nécessaire d'utiliser les métamodèles sur de multiples domaines pour évaluer leur complétude.

Finalement, tous les sujets trouvent que l'exercice doit être davantage guidé par des définitions et des exemples de concepts et par un exemple d'instanciation pour éviter les mauvaises interprétations des concepts et minimiser les ambiguïtés lors de la modélisation.

6.1.6 Synthèse

Les expérimentations réalisées sur les deux métamodèles structurel et évènementiel ont montré que ces métamodèles évoquent des aspects intéressants sur la modélisation structurelle et dynamique du contexte. En effet, la notion d'entité facilite le travail de modélisation et la classification des éléments de contexte en trois catégories (paramètre de profil, relation entre entité et donnée de l'environnement) a été bien comprise. Le lien entre les évènements et le contexte a été clair pour tous les sujets et la plupart ont réussi à cartographier et typer correctement les évènements.

Cependant, certaines critiques ont concerné certains concepts tels que la spécialisation des relations entre entités et l'ambiguïté du concept "source d'évènement". Nous avons décidé, par conséquent, de remédier à ce problème en apportant des modifications à ces métamodèles pour qu'ils répondent mieux aux besoins des développeurs. La réponse à ces différentes critiques et les améliorations apportés aux métamodèles sont détaillées ci dessous.

Lien entre modèle de domaine et modèle de contexte. Concernant la remarque des spécialistes IHM de séparer le modèle de domaine du modèle de contexte, nous sommes néanmoins convaincus que ces deux modèles s'entrelacent fortement et que la séparation entre les deux risque de créer des vues insuffisantes. En effet, le modèle de contexte dépend fortement du modèle de domaine et reprend des parties de ce dernier pour couvrir toutes les situations contextuelles.

Lien entre les métamodèles structurel et évènementiel La partie dynamique représente l'originalité de notre approche et il a été suggéré par certains sujets de la relier à la partie structurelle. Nous pensons, par contre, qu'il est plus pertinent de garder cette vue séparée pour pouvoir effectuer les analyses structurelles et dynamiques de façon indépendante. Cependant pour le métamodèle d'évènements, il faut revoir les liens entre évènements et éléments de contexte et les présenter de manière plus explicite. De plus, il faut envisager des liens possibles entre évènements.

Usage des sources d'évènements Pour les sources d'évènements qui n'étaient pas toujours évidentes à identifier, le problème réside dans leur hétérogénéité. Par exemple, les évènements métier proviennent essentiellement de processus métier. Or, la réalisation de l'expérimentation ne comportait pas une analyse métier pour pouvoir identifier l'origine de ce type d'évènements. Concernant les évènements utilisateur, ils sont générés par des interactions avec l'utilisateur qui ne sont pas toujours claires pour les sujets. Les évènements externes proviennent de plusieurs sources telles que d'autres systèmes, des services ou des capteurs. Afin de garder la généricité du diagramme, la notion de source d'évènement est conservée mais son instanciation devra être mieux guidée.

Guidage de la modélisation Pour faciliter le travail de modélisation, il a été recommandé par les sujets de guider le processus d'instanciation et de fournir des exemples pour simplifier la tâche et borner les choix. Ainsi, nous préconisons l'usage du métamodèle avec un ensemble de règles d'usage pour guider la modélisation.

Les expérimentations réalisées ont permis d'apporter des améliorations aux deux métamodèles pour supporter une modélisation structurelle et une modélisation dynamique du contexte. Les métamodèles finaux résultant de l'évaluation de ces expérimentations sont présentés dans le chapitre 3.

Dans la version finale du métamodèle structurel (voir figure 3.5), nous avons enlevé la spécialisation des relations entre entités. Nous avons également créé la classe “Règle” pour supporter l'expression des paramètres de profil sous forme de règle, par exemple "s'il fait chaud, je préfère prendre le vélo". La classe “Relation entre éléments de contexte” pour l'expression de relations entre éléments de contexte a également été introduite dans le métamodèle.

De même, l'analyse des résultats des expérimentations a permis de réaliser des améliorations concernant le lien entre évènements et éléments de contexte et les relations entre évènements. Ainsi, le nouveau métamodèle évènementiel est présenté par la figure 3.6.

Au delà de la finalisation des métamodèles, ces expérimentations ont été très instructives pour commencer à avoir des éléments pertinents d'une démarche d'apprentissage et de guidage de modélisation.

6.2 Prototype d'assistance basé sur un moteur de traitement des évènements

L'approche orientée évènement adoptée dans la démarche E-CARe (voir section 3.2.4) se manifeste par l'usage d'un réseau de traitement des évènements organisant le SI en un ensemble de sous-systèmes appelés "unité de traitement des évènements". Chaque unité comporte un moteur de traitement des évènements qui reçoit des évènements en entrée et produit des évènements en sortie.

L'objectif de cette section est de valider l'architecture interne d'une unité de traitement des évènement et prouver son adéquation pour le support des fonctionnalités ubiquitaires. Ainsi, nous proposons de développer un prototype d'une application d'assistance au voyageur qui comporte une unité permettant le traitement des évènements et la personnalisation des informations délivrées. Ce prototype se positionne dans le réseau de traitement des évènements d'un SI de transport présenté dans la figure 3.4 comme étant le module "Moniteur Application mobile".

6.2.1 Fonctionnalités visées

Le prototype a pour objectif de tester l'utilisabilité des moteurs d'évènements pour :
- l'adaptation et la sensibilité au contexte : filtrer des évènements en fonction du contexte et du profil. Cette action met en œuvre l'usage des règles d'adaptation (voir section 3.2.3),
- la mise à jour des données contextuelles : utiliser les évènements reçus pour mettre à jour une donnée de contexte ou de profil,
- l'usage du concept "Règle" du métamodèle structurel de contexte : définir des intérêts particuliers dépendants de certains évènements ambiants, tels que "informer le voyageur de tous les évènements de retard de train durant une période de neige".

Le prototype de l'application d'assistance au voyageur doit donc permettre :
- la gestion du contexte, en définissant une base de données contextuelles gérée par l'utilisateur qui est le voyageur même. Ce dernier a la possibilité de définir ses données de profil comme ses préférences, son âge, ses itinéraires, ses empêchements, etc. De plus, les données de contexte peuvent être définies par l'application suite à un traitement d'évènements : tel est par exemple le cas de la localisation du voyageur.
- la détection des patrons d'évènements, en utilisant le moteur d'évènements et en étant à l'écoute de tous les évènements ambiants. Ainsi, il est possible de détecter des situations particulières qui combinent plusieurs évènements et des conditions sur ces évènements.
- la personnalisation des évènements, en utilisant les données de profil pour filtrer les évènements et n'afficher à l'utilisateur que les évènements qui le concernent. Ceci est réalisé par application des règles d'adaptation.
- la définition de règles de profil, en exprimant des besoins particuliers par le voyageur qui définit des conditions pour afficher certains évènements.

6.2.2 Architecture du prototype

L'architecture générale du prototype est représentée par la figure 6.3. Le composant central du prototype est le moteur d'évènements. Plusieurs moteurs d'évènements ont été étudiés (voir annexe B) et le moteur Esper a été utilisé suite à cette comparaison. Les différents modules de l'application sont les suivants :
- **Esper Event Engine (moteur d'évènements Esper)** : permet le traitement des évènements, la détection des patrons d'évènements (par des écouteurs) et les exécutions des requêtes. Les évènements sont définis dans Esper sous forme de classes JAVA. Les requêtes sont exprimées

avec le langage Event Processing Language (EPL) et des écouteurs permettent de détecter les patrons d'évènements contenus dans les requêtes.
Le moteur Esper facilite l'usage et la gestion d'un grand nombre d'évènements puisqu'il se charge de l'évaluation continue des requêtes (représentant les règles d'adaptation ou les patrons d'évènements) et produit les actions appropriées dans le cas où la requête est satisfaite.

- **Oracle Database (base de données Oracle)** : contient les données de contexte et de profil de l'utilisateur. La base de données de contexte est conçue et développée en ORACLE. Elle contient les données de profil de l'utilisateur pouvant influencer ses déplacements en transport publics. Elle contient les itinéraires habituels, les adresses intéressantes, les empêchements, la localisation et les préférences de l'utilisateur. Afin de pouvoir notifier le voyageur en fonction de son profil, la base de données est couplée avec les requêtes Esper.
- **Context Updater (mise à jour du contexte)** : permet la mise à jour des données contextuelles par écoute des évènements ambiants qui le concernent. Dans ce module, un ensemble de requêtes est défini et permet de détecter des évènements influençant le contexte. La satisfaction de ces requêtes engendre un accès et une mise à jour de la base de données.
- **Server Interface (Interface du serveur)** : dédiée à l'administrateur, ce module permet de créer dynamiquement de nouveaux évènements et de nouvelles requêtes. Ces évènements générés sont écoutés par le prototype de l'application mobile et traités par le moteur d'évènements.
- **Client Interface (Interface Client)** : dédiée à l'utilisateur, ce module permet d'être notifié, de créer de nouvelles requêtes représentant ses besoins d'information et de visualiser et modifier son profil.

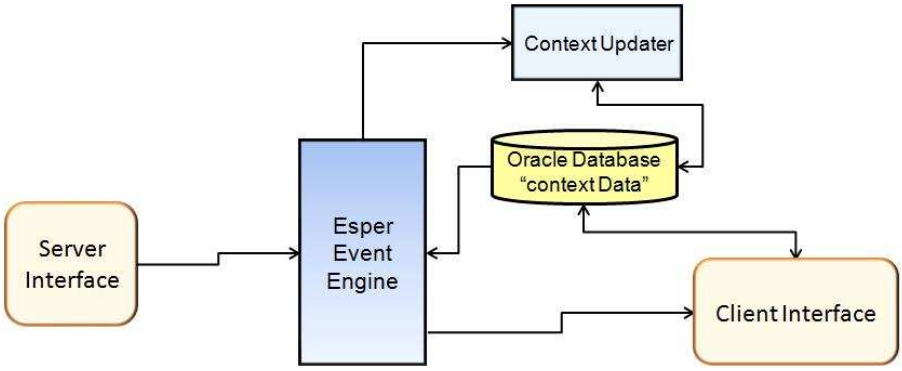


FIGURE 6.3 – Architecture du prototype

6.2.3 Le prototype développé

Cette application est un démonstrateur technique qui prouve l'usage des règles pour la notification de l'utilisateur et la mise à jour du contexte. Basée sur l'architecture présentée en figure 6.3, elle propose une interface pour l'administrateur qui représente le SIT et lui permet de générer des évènements envoyés au client, ainsi qu'une interface pour le client qui permet à l'utilisateur de recevoir des notifications personnalisées et d'exprimer ses besoins en informations.

Interface du serveur

L'administrateur peut, par l'intermédiaire de l'interface du serveur (voir figure 6.4), définir de nouveaux évènements et nommer et donner les types des attributs de ces évènements. Une fois cela effectué, l'application génère l'évènement sous la forme d'un fichier Java, compile ce fichier à la volée

et sauvegarde cet événement dans un répertoire spécifique. L'évènement est alors ajouté à la liste des évènements disponibles pour l'utilisateur. Enfin, l'administrateur peut aussi créer de nouvelles requêtes Esper pour définir des besoins d'adaptation particuliers.

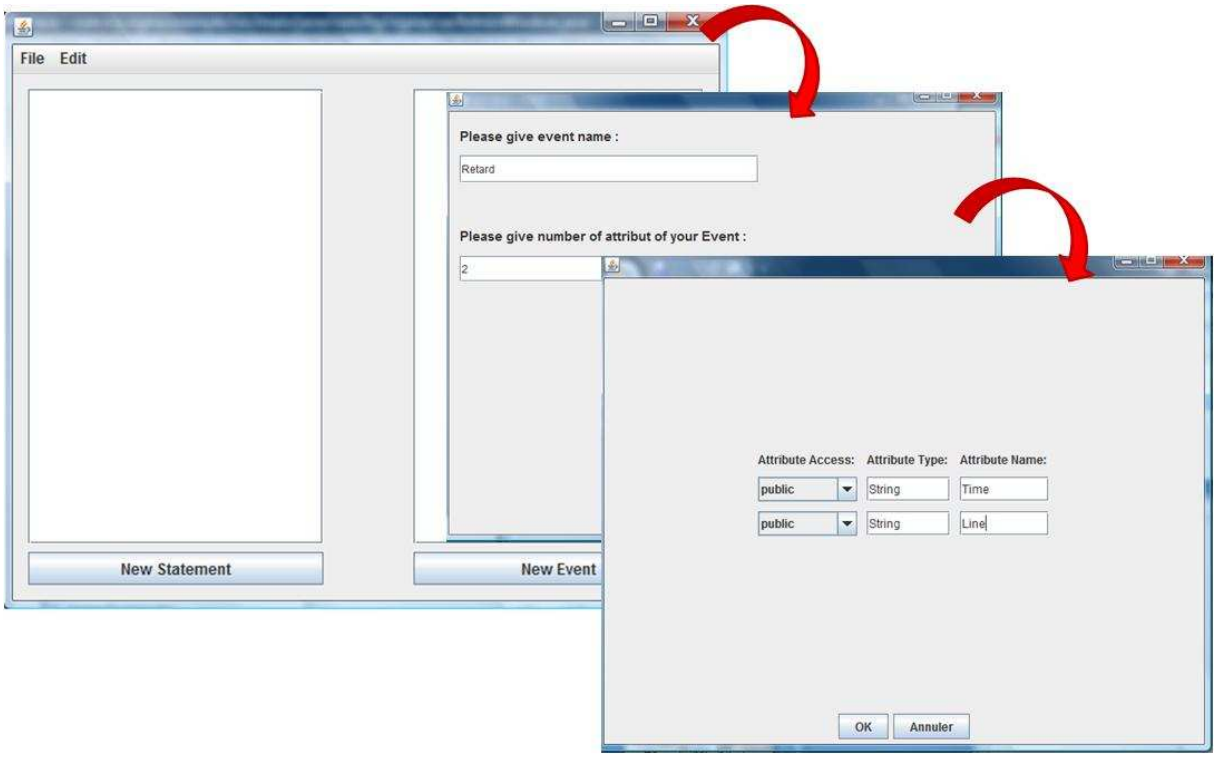


FIGURE 6.4 – Interface du serveur

Interface du client

L'interface du client (voir figure 6.5) permet au voyageur de communiquer avec l'application, de gérer son profil et de recevoir des notifications du SIT qui peuvent l'intéresser. Par son intermédiaire, le voyageur peut consulter les informations concernant son profil et les modifier. De plus, il a la possibilité de définir son itinéraire courant parmi une liste d'itinéraires habituels, ce qui lui permet d'être informé d'éventuelles perturbations en fonction du moyen de transport emprunté lors de son déplacement dans le réseau.

Exemple d'application

Pour illustrer l'utilisation de cet outil, nous prenons l'exemple d'un évènement retard de la ligne B. Tant que le voyageur n'ajoute pas cette ligne dans son itinéraire courant, il ne sera pas notifié de ce retard. Dans la figure 6.6, la rubrique notification est vide car le voyageur utilise la ligne C.

Si le voyageur change son itinéraire pour utiliser la ligne B, il sera prévenu du retard du tram. Dans la figure 6.7, le message signalant le retard du tram B est représenté dans la rubrique de texte réservée aux notifications.

Pour réaliser ce traitement d'évènements, une requête Esper interroge la base de données de l'utilisateur afin de connaître les lignes de tram utilisées dans l'itinéraire courant, puis compare ces résultats avec la ligne de tram concernée par l'évènement. La requête utilisée est un "statement" défini

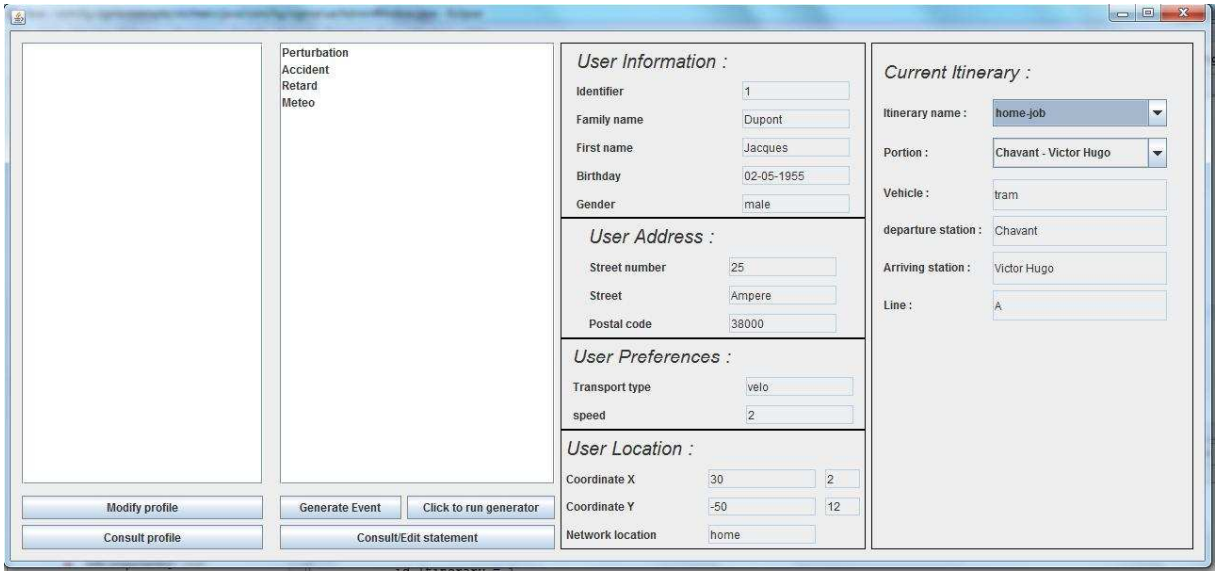


FIGURE 6.5 – Interface du client

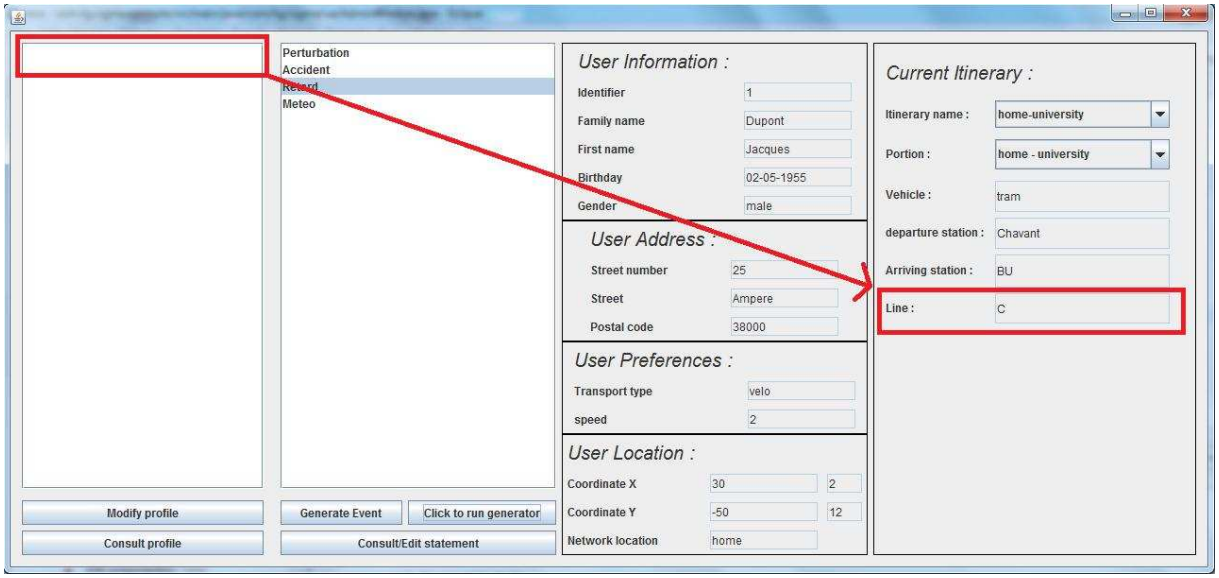


FIGURE 6.6 – Itinéraire choisi sans perturbations

dans Esper (voir figure 6.8). Le listener de cette requête indique le message à afficher à l'utilisateur et sa prise en compte dans l'itinéraire. Cette requête, qui représente en fait une règle d'adaptation de l'information, se charge de l'écoute des évènements ambiants, de leur évaluation et de la notification de l'utilisateur.

6.2.4 Bilan

L'implémentation de l'application a montré la facilité d'utiliser les moteurs de traitements des évènements pour la définition des règles ubiquitaires sous forme de requêtes. Les fonctionnalités d'adaptation au contexte sont ainsi rendues possibles. L'utilisation de ces moteurs permet d'éviter le développement long et les tests continus des évènements. De plus, ces moteurs fournissent un

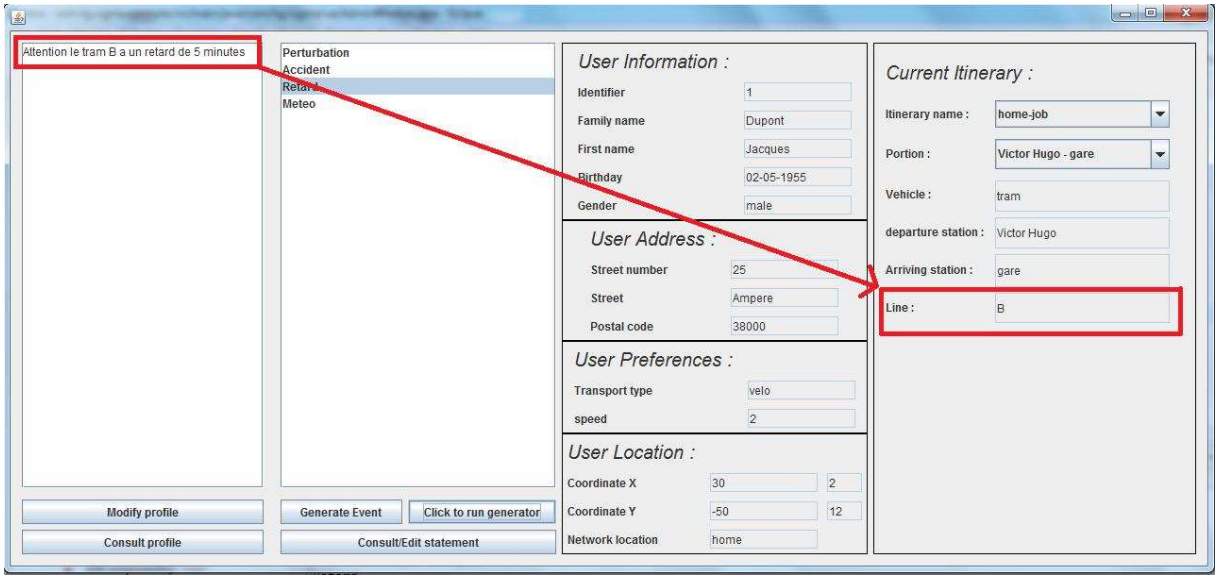


FIGURE 6.7 – Itinéraire choisi après une perturbation affectant une ligne du premier itinéraire

```
import com.espertech.esper.client.EPAdministrator;

public class RetardStatement {

    EPStatement stmt;

    public RetardStatement(EPAdministrator epAdmin){
        stmt = epAdmin.createEPL(
            "select vehicle,late,ret.line as line,LINEITINERARY " +
            "from Retard.win:length(30) AS ret,"+
            "sql:XE ['select line as lineItinerary from portions " +
            "where id_itinerary= (select id_itinerary from UsualItinerary where currentitinerary=1)'] "+
            "where ret.line =  LINEITINERARY" , "RetardStatement");
    }

    public void addListener(UpdateListener listener){
        stmt.addListener(listener);
    }

}
```

FIGURE 6.8 – Requête Esper utilisée pour signaler un retard lié à un itinéraire

moyen robuste pour définir un grand nombre d'évènements, de patrons d'évènements et de règles. L'utilisation de ces moteurs avec une application mobile simplifie ainsi le développement et fournit un large panel de services contextuels.

6.3 Prototype d'une application d'assistance au voyageur

Le but de ce prototype est de réaliser un démonstrateur technique mettant en évidence le potentiel du projet DéSIT [10] à travers une application embarquée sur un terminal mobile, permettant le guidage des personnes handicapées dans les transports en commun. Basé sur les choix architecturaux et technologiques de la démarche E-CARe, il a été réalisé par deux PFE de l'INSA de Lyon sous la tutelle de Stéphane Coulondre.

Ce prototype permet de vérifier la compatibilité des applications à architecture orientée évènements

pour fournir des fonctionnalités ubiquitaires et sensibles au contexte tout en respectant la sécurité des informations personnelles.

Les flux de données sont issus des serveurs du système d'information de transport, eux-mêmes reliés à des bornes Bluetooth implantées sur les stations et les véhicules du réseau. Muni d'un téléphone portable compatible, l'utilisateur peut alors personnaliser les informations qu'il souhaite recevoir, et profiter des services offerts par l'application. Les spécifications définies par cette application comprennent :

- un client embarqué sur le téléphone écrit en Java ME, appelé HelpMe.
- un serveur sur un ordinateur, permettant d'envoyer des messages au téléphone via Bluetooth.

6.3.1 Fonctionnalités visées

L'objectif du prototype est de valider l'usage d'une architecture orientée évènements pour la conception d'une application ubiquitaire communiquant avec son environnement par échanges de flux d'évènements avec la technologie Bluetooth. Ce prototype permet de tester les questions architecturales suivantes :

- la communication par flux d'évènements est-elle suffisante pour garantir une vigilance à l'environnement et un fonctionnement correct tout en étant supportée par des technologies les plus répandues et les moins coûteuse (cas de la technologie Bluetooth) ?
- la personnalisation et la sensibilité au contexte sont-elles possibles sans risque d'atteinte à la sécurité des données personnelles ?
- les sous-systèmes dans un réseau de traitement des évènements peuvent-ils être complètement autonomes et supportés par les dispositifs mobiles les plus basiques ?

Pour répondre à ces questions, l'application d'assistance s'intéresse aux échanges d'évènements à l'intérieur d'un réseau EPN pour fournir les services suivants au voyageur :

- sauvegarder localement au niveau du terminal mobile de l'utilisateur, ses données privées et confidentielles,
- afficher à l'utilisateur, sur son terminal mobile, les messages d'informations envoyés par le serveur ou par d'autres services,
- afficher des alertes triées et filtrées par rapport au profil de l'utilisateur. Elles concernent ainsi ses données de déplacements ; ces alertes sont des messages (évènements) concernant les données de déplacement de l'utilisateur,
- calculer des itinéraires en considérant le profil de l'utilisateur,
- guider le voyageur pendant un déplacement correspondant à un itinéraire prédéfini.

6.3.2 Architecture et choix techniques

Le prototype consiste en une application sur un terminal mobile et un serveur de système de transport qui communiquent par échanges de flux d'évènements via Bluetooth. Ceci permet de se mettre dans une situation proche de celle d'un voyageur en déplacement qui communique avec divers véhicules et arrêts du système de transport pour recevoir des informations temps réels et des alertes (voir figure 6.9).

Le développement de ce type d'application nécessite l'usage de technologies spécifiques et appropriées pour permettre de supporter les connexions Bluetooth ainsi que les données et les services de transport.

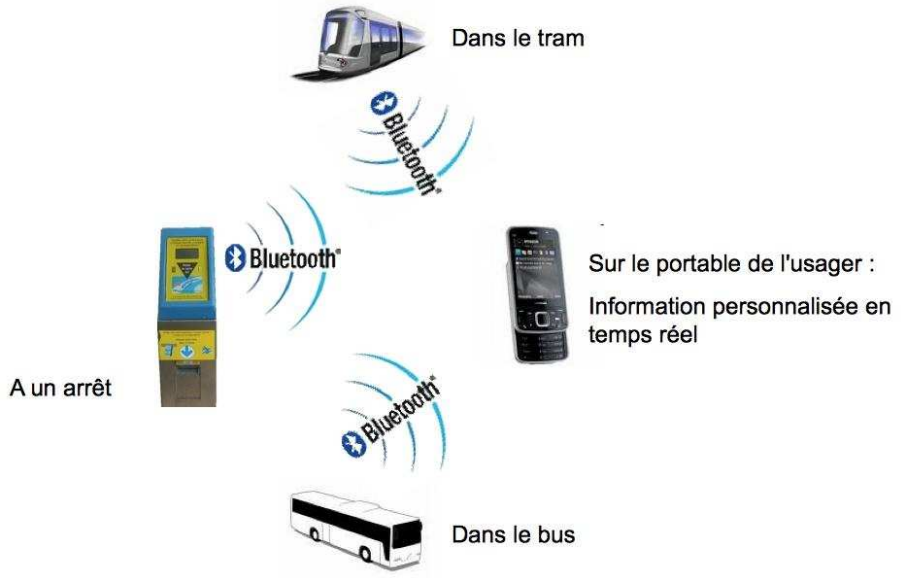


FIGURE 6.9 – Application d'assistance communiquant par échanges de flux d'évènements

Usage de la technologie Bluetooth

Pour la spécification des connexions Bluetooth, l'application utilise *jsr82* "Java Request for Specifications" qui est un système normalisé de spécifications pour le langage Java décrivant les interactions avec Bluetooth. Elle utilise également l'*API Bluecove*, une API libre pour Java permettant d'interfacer des applications avec *jsr82*. Cette API est largement utilisée ici pour toutes les fonctionnalités Bluetooth comme par exemple la gestion des connexions (recherche de services, extraction d'URL) et l'envoi des données.

Flux d'évènements

L'application est supposée récupérer des informations multiples, venant non seulement du système de transports, mais aussi d'internet et de fournisseurs de services tierces. Ces données sont formatées en XML et peuvent être par exemple :

- **des informations météo,** utilisées pour proposer des moyens transport appropriés avec l'état de la météo (par exemple, l'usage du vélo quand il fait beau). Dans cette application, ces informations sont utilisées comme de simples notifications (voir figure 6.10).
- **des informations ferroviaires,** utilisées pour informer sur les prochains départs des trains et TER. Par exemple, l'application est adaptée à un service déjà existant, le widget "Prochains départs" du site des TER e-services. Cette application fournit le service éponyme, c'est-à-dire une liste des prochains départs d'une gare donnée (voir figure 6.10).
- **des informations Transport,** utilisées pour avoir connaissance du réseau de transport et des perturbations de transport. L'utilisation de formats de données standardisés est nécessaire. Plusieurs projets antérieurs ont déjà traité ce point, notamment les projets CHOUETTE [8] et TRIDENT [19]. Le projet TRIDENT fournit une série de schémas XML (fichiers XSD) définissant avec une grande précision le format que doivent avoir les données relatives aux SIT. Le modèle de données BIDENT a été créé s'appuyant sur les standards existants et est assez abstrait et complet pour pouvoir modéliser tout type d'événements ou d'informations.

FIGURE 6.10 – Affichage des informations météo et SNCF avec l'application

Calcul d'itinéraire

Une partie conséquente du développement s'est axée autour du guidage du voyageur en déplacement à travers le réseau. Concrètement parlant, il s'agit de fournir une procédure de choix d'itinéraire entre deux points, avec un calcul du plus court chemin par l'application. La faisabilité d'une telle fonction en JavaME a été étudiée, elle est possible à condition de maintenir l'algorithme à une complexité raisonnable. En l'absence d'informations géographiques détaillées sur les stations, l'algorithme de Dijkstra [9], simple et efficace, s'est imposé pour la recherche du plus court chemin.

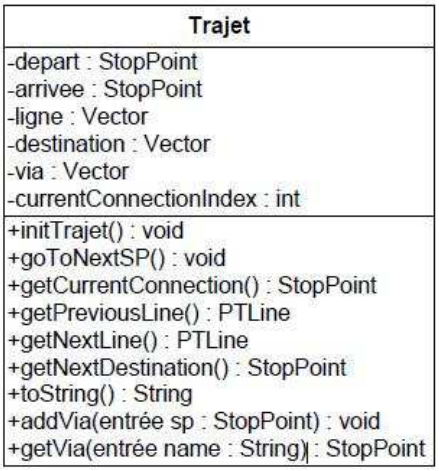


FIGURE 6.11 – La classe "Trajet"

Les informations calculées par l'algorithme sont stockées dans une classe Trajet (voir figure 6.11). Le suivi du trajet concerne surtout la possibilité d'alerter le voyageur à chaque correspondance, en lui indiquant de quitter la ligne actuelle, et de prendre une nouvelle ligne, avec sa destination pour éviter toute erreur ou incertitude fébrile. Pour cela, la classe Trajet contient les données nécessaires pour le suivi.

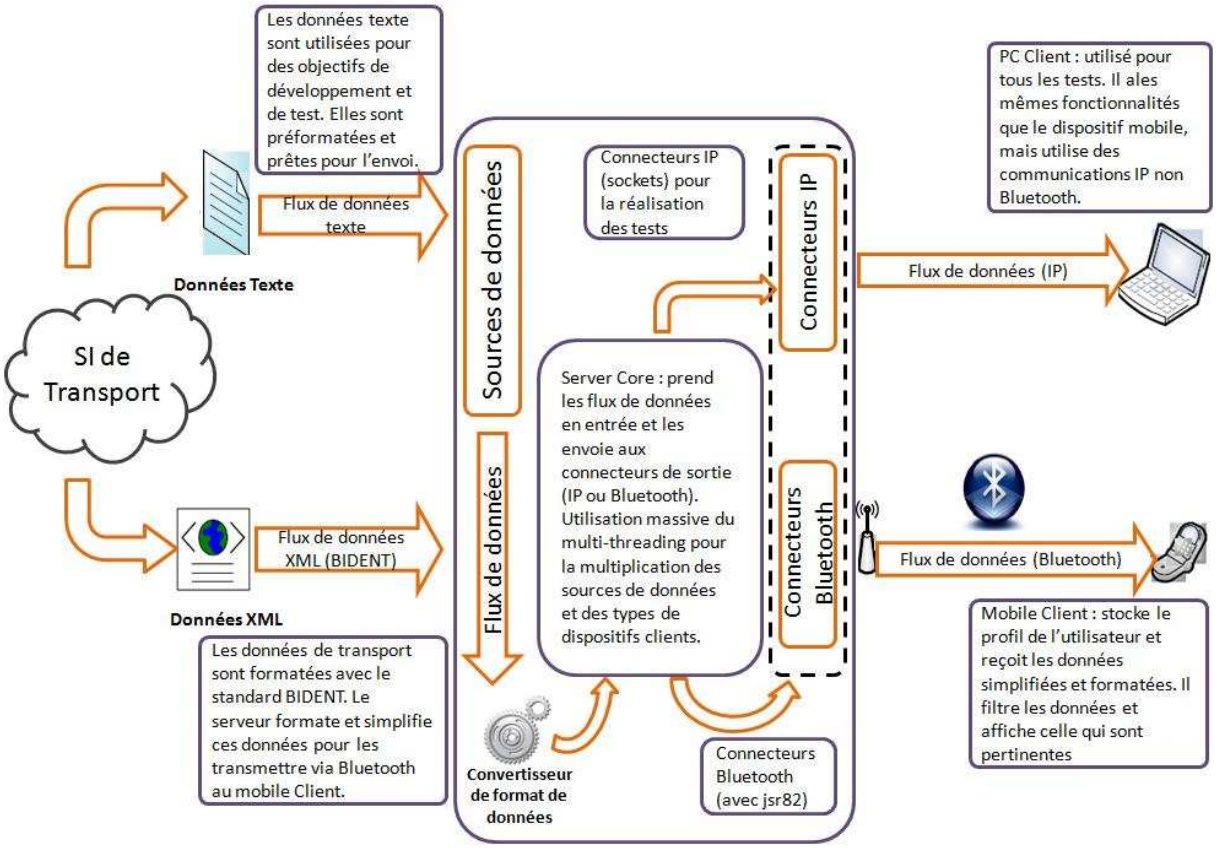


FIGURE 6.12 – Illustration de la gestion des données dans le serveur

6.3.3 Application serveur

Le serveur est l'application chargée de rassembler les informations de transport, de les centraliser puis de les envoyer à tous les terminaux connectés. Le serveur a également en charge de gérer toutes les connexions avec des terminaux allumés : dès lors qu'un utilisateur démarre l'application cliente dans une zone de couverture, celle-ci est automatiquement prise en charge par le serveur. Le serveur est basé sur la classe "Core" qui centralise la gestion des entrées, des sorties et des messages : voir figure 6.12.

Afin d'avoir une flexibilité maximale, le serveur doit pouvoir recevoir des données de différents formats et depuis différentes sources. De même, les sorties du serveur doivent pouvoir être de différentes natures. Pour y parvenir, le serveur utilise des interfaces qu'implémenteront des classes spécifiques (pour la lecture de données XML ou texte par exemple) pour la réalisation de différentes tâches. Ainsi, une interface IDataReader décrit les méthodes à implémenter pour la lecture de données et une interface IConnector liste les méthodes nécessaires à la connexion avec différents types de périphériques.

Les données d'entrée du serveur sont visibles sur l'interface graphique du serveur de la figure 6.13.

6.3.4 Application client

L'application client est embarquée sur le terminal mobile de l'utilisateur, elle permet de de guider et d'informer le voyageur pendant un déplacement. Elle communique avec le serveur par échanges de messages qui représentent les données de transport, de météo ou des informations ferroviaires par le biais d'une liaison Bluetooth. L'application client doit recevoir les évènements, les traiter auto-

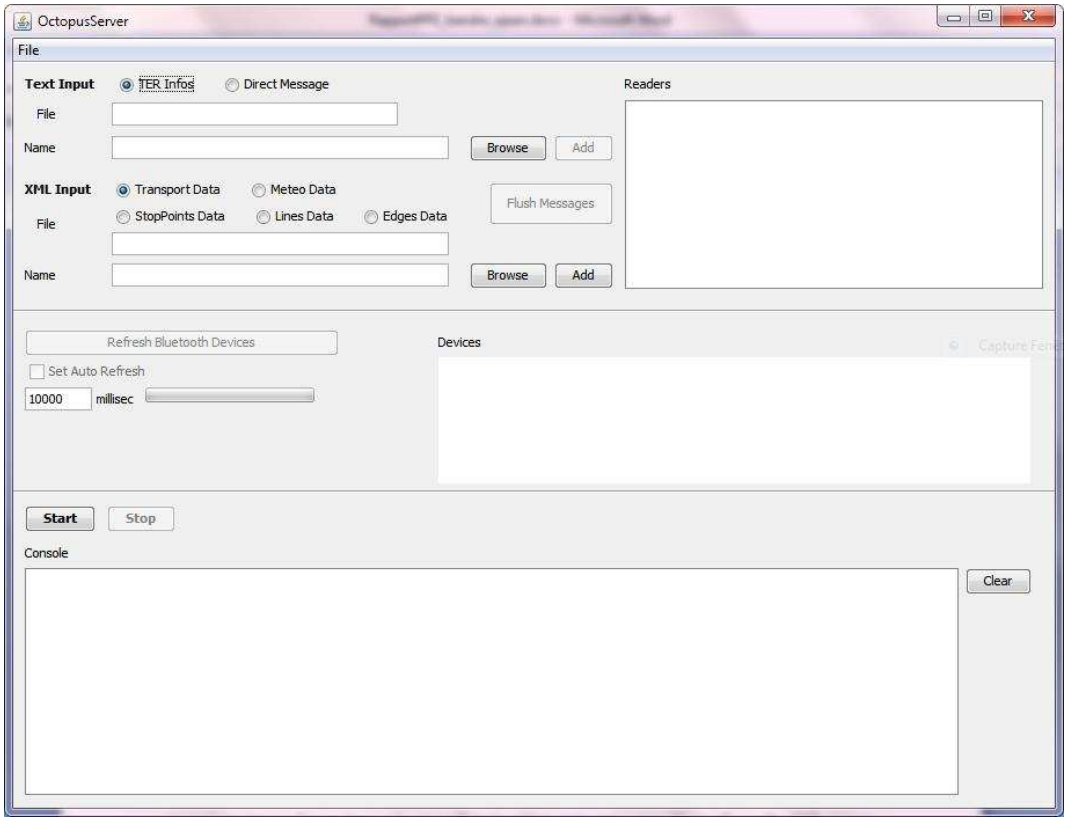
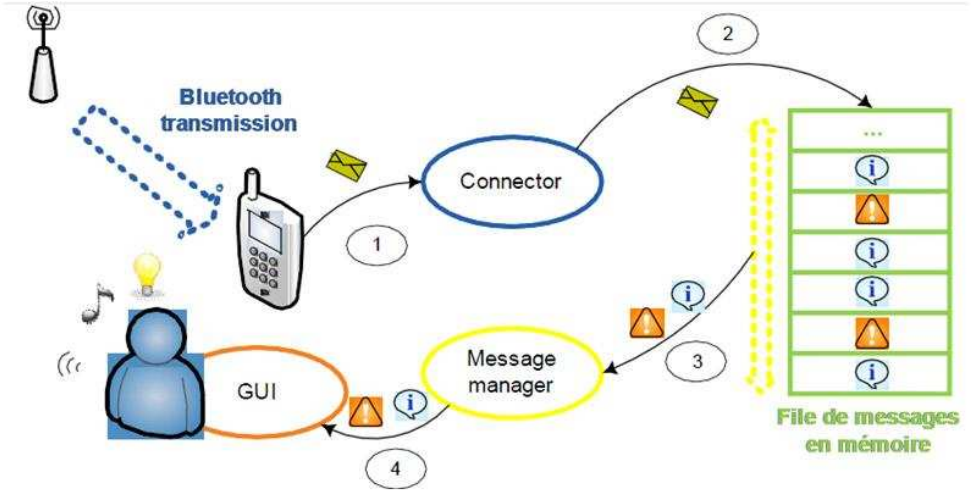


FIGURE 6.13 – Capture d'écran de l'interface du serveur

matiquement et les afficher sans aucune intervention de l'utilisateur. De plus, aucun évènement ne doit être perdu, ce qui signifie dans le cas réel que dès lors qu'un voyageur entre dans une zone de couverture du système, il est intégralement pris en charge par ce dernier et reçoit tous les évènements envoyés par le serveur. L'application client doit être capable de recevoir et de traiter un grand nombre de messages successifs.

L'application client permet au voyageur de définir son profil afin d'obtenir un service personnalisé. Les données de profil contiennent le degré de mobilité, les préférences d'affichage et les préférences en mode de transport. En plus de son profil personnel, le voyageur peut choisir le trajet qu'il veut effectuer et qui sera stocké dans la mémoire du téléphone pour être réutilisé au redémarrage de l'application. L'application client filtre ensuite les messages reçus du serveur en fonction du profil et du trajet du voyageur. Seules les informations pertinentes seront alors affichées au voyageur.

L'application client s'articule en 3 modules fonctionnant en parallèle dans des threads différents. Un message arrivant sur le téléphone est tout d'abord pris en charge par le module connector qui le stocke en mémoire en queue de file. Cette file de messages est gérée par le manager qui la parcourt et en récupère le message suivant. Il en extrait les données et si elles concernent le voyageur, il les affiche sur l'interface utilisateur. Celle-ci fait la liaison entre le cœur de l'application et le voyageur. Lors de l'affichage de nouvelles informations concernant le voyageur, celui-ci est prévenu par un signal sonore, vibratoire ou lumineux selon les préférences qu'il a défini via l'interface utilisateur et qui sont prises en compte par l'ensemble des modules de l'application. Ces différentes étapes sont résumées dans la figure 6.14.



1 : Réception du message via Bluetooth par l'entité « connector » de l'application,
2 : Stockage en queue de la file de messages du message reçu par le « connector »,
3 : Parcours de la file de message par le manager et récupération du message le plus prioritaire (alerte la plus ancienne ou information la plus ancienne si aucune alerte n'est en file),
4 : Traitement du message récupéré : vérification de son adéquation au profil usager. Si le message concerne l'usager il est alors affiché et signalé au niveau de l'interface utilisateur en fonction de son type (alerte, information).

FIGURE 6.14 – Fonctionnement de l'application client

6.3.5 Prototype final

Une série de tests réalisée sur l'application a permis l'amélioration continue de l'application sur le plan ergonomique et fonctionnel et au final l'application permet de fournir les services décrits ci-dessous.

Démarrage de l'application

Au lancement de l'application, un écran d'attente de réception du réseau de transport local est affiché (voir l'écran A de la figure 6.15). Une bonne réception des données est matérialisée par un passage automatique à l'écran d'accueil. Dans le cas contraire, l'utilisateur peut mettre fin au processus grâce à une touche "quitter".

Menu d'accueil

Un menu est utilisé pour introduire les fonctionnalités de l'application et permettre à l'utilisateur de sélectionner son choix (voir l'écran B de la figure 6.15).

Définition du profil

L'utilisateur peut éditer son profil pour définir ses préférences en mode de transport, son niveau de mobilité et ses préférences en mode d'alerte (voir figure 6.16).

FIGURE 6.15 – Écran d'accueil et menu d'accueil

FIGURE 6.16 – Édition du profil

Calcul d'itinéraire

Une succession d'écrans apparaît pour répondre à la fonctionnalité de guidage (voir figure 6.17), c'est-à-dire la recherche d'itinéraire entre deux stations quelles qu'elles soient. Les noms des arrêts de départ et d'arrivée sont entrés dans des champs texte à l'aide du clavier. Pour plus de rapidité, il suffit simplement d'entrer les premières lettres et l'application se charge de trouver la liste des arrêts correspondants. Le voyageur peut alors sélectionner la bonne option, ou retourner modifier sa saisie.

FIGURE 6.17 – Recherche d'un itinéraire

Guidage d'itinéraire

L'utilisateur peut choisir d'être guidé dans un itinéraire parmi l'ensemble des itinéraires sauvegardés dans l'application (voir écran A de la figure 6.18) ou de choisir un nouvel itinéraire. Ainsi, il est alerté à chaque correspondance pour descendre à un arrêt ou pour monter dans un véhicule de transport (voir écran B de la figure 6.18)

FIGURE 6.18 – Choix d'itinéraire et guidage

Alertes et informations

L'utilisateur reçoit uniquement les alertes qui concernent son profil, c'est-à-dire les trajets sauvegardés (voir écran A de la figure 6.19). Les autres informations sont reçues aussi, mais elles ne sont pas prioritaires (voir écran B de la figure 6.19).

FIGURE 6.19 – Alerte de retard d'un véhicule

6.3.6 Bilan

Ce prototype a montré la faisabilité technologique des systèmes orientés évènements basés sur une architecture Bleutooth et l'ensemble de leurs avantages. De tels systèmes permettent une communication facile et non coûteuse en terme d'informations et de débit. De plus, ils favorisent l'indépendance des services et leur bonne orchestration grâce à la définition de sous-systèmes indépendants en terme de données utilisées ou de fonctionnalités ubiquitaires intégrées. Ainsi, la sécurité et la confidentialité des données personnelles est garantie, ces données n'étant pas échangées entre systèmes.

Ce prototype a permis, de plus, de vérifier l'usage des technologies Bluetooth dans le cadre du projet DéSIT. Il fournit une solution pratique et faisable pour l'assistance des voyageurs dans un réseau de transport où les véhicules et les stations sont aujourd'hui équipés avec des émetteurs Bluetooth (c'est en particulier le cas du réseau TAG de l'agglomération Grenobloise).

6.4 Synthèse

Ce chapitre permet la validation de trois aspects importants de la démarche E-CARe sur les SI ubiquitaires : l'architecture orientée évènements, l'usage des moteurs de traitement d'évènements pour l'adaptation au contexte et la généricité et la complétude des modèles de contexte. En effet, les prototypes d'application d'assistance au voyageur ont montré la possibilité de réaliser des systèmes ubiquitaires indépendants et autonomes pour le traitement des évènements dans un objectif d'adaptation et de réaction au contexte. Ces systèmes communiquent par échanges d'évènements en garantissant la sécurité des données contextuelles et des données personnelles.

Les expérimentations auprès d'experts de la conception des SI ubiquitaires de différentes disciplines ont permis l'amélioration des métamodèles structurels et évènementiels utilisés pour modéliser le contexte. Ces expérimentations ont de plus permis de récupérer l'expérience des sujets en terme de

modélisation du contexte, ce qui a joué un rôle important pour l'amélioration de la démarche E-CARe.

Conclusion et Perspectives

Les systèmes ubiquitaires représentent une nouvelle tendance adoptée dans différents domaines, en particulier ceux qui permettent l'usage des dispositifs mobiles, des environnements ambiants et des systèmes interactifs. La demande croissante de telles applications et la diversification de leurs usages nécessitent d'accorder un intérêt plus conséquent à leur développement et à faciliter leur conception.

La conception des applications ubiquitaires relève de nombreux défis reliés à la modélisation des systèmes, la garantie de la sécurité, le dynamisme des fonctions et l'usage des technologies ubiquitaires. La qualité du système conçu dépend de la résolution de ces défis et l'organisation des différentes spécifications dans un tout cohérent. Chaque défi présente à lui seul un thème de recherche et une problématique pour les systèmes ubiquitaires. Ainsi, la définition d'une méthode de développement de ces systèmes est considérée comme une tâche délicate et non triviale. Elle nécessite des choix conceptuels, architecturaux et technologiques couvrant et supportant les caractéristiques ubiquitaires voulues. La combinaison de ces choix doit fournir un système cohérent, utilisable et maintenable dans son ensemble. Notre objectif dans cette thèse est de proposer une méthode de développement de ces systèmes en faisant des choix appropriés d'approches et de modèles pour répondre aux différents défis.

Résumé des contributions

Cette thèse est un guide pour le développement des SI ubiquitaires qui fournit une palette d'outils pour les développeurs des systèmes facilitant la spécification des différents besoins en liaison avec les caractéristiques ubiquitaires ou avec les autres aspects d'un SI ubiquitaire.

La priorité de ces outils de développement est de répondre de façon efficace aux besoins des clients et des utilisateurs en vue de fournir une application avec des fonctionnalités ubiquitaires appropriées et utilisables. Ainsi, nous proposons d'utiliser une approche intentionnelle dirigée par les objectifs qui analyse les objectifs des utilisateurs, les raffine en identifiant les objectifs ubiquitaires et les transforme en des règles ubiquitaires résumant les caractéristiques d'adaptation et de sensibilité au contexte. Les caractéristiques ubiquitaires non fonctionnelles du système comme les besoins de sécurité, de scalabilité et de mobilité sont considérées comme des règles techniques.

Cette approche intentionnelle est intégrée dans une démarche d'ingénierie globale qui sépare les spécification fonctionnelles, ubiquitaires et techniques. Pour les phases de spécification fonctionnelle et technique, nous avons adopté la démarche "Symphony", démarche classique d'ingénierie. Les spécifications ubiquitaires forment un ensemble de phase purement ubiquitaire intégré dans notre démarche E-CARe. Dans cette démarche, les besoins ubiquitaires représentés par les règles ubiquitaires sont utilisés pour définir le modèle de contexte en identifiant les données contextuelles nécessaires pour les fonctionnalités d'adaptation et de sensibilité au contexte. Des modèles évènementiels mettant en œuvre une approche orientée évènements, sont produits à la suite en définissant les évènements permettant l'acquisition du contexte et le déclenchement des comportements d'adaptation et de notification. Les modèles ubiquitaires obtenus à l'issue de la branche ubiquitaire (contenant l'ensemble des

phases ubiquitaires) sont les modèles contextuels structurels et évènementiels et les règles ubiquitaires.

Les modèles ubiquitaires doivent s'intégrer dans les modèles fonctionnels et être pris en compte dans les modèles techniques et architecturaux pour pouvoir être supporté par les composants du système. Ceci est réalisé lors de la conception du système en fusionnant les modèles fonctionnels, ubiquitaires et techniques.

Cette démarche de développement a été évaluée grâce à des expérimentations réalisées auprès de chercheurs manipulant des SI ubiquitaires. De plus, en suivant cette démarche, le prototype d'une application d'assistance au voyageur dans les transports publics a été développé. Le système conçu est basé sur la diffusion et la réception de flux, l'idée étant que toute information de transport sera diffusée sous la forme d'un flux grâce à la technologie Bluetooth, puis réceptionnée, traitée et affichée sur le dispositif mobile de l'usager selon son profil, ses préférences, sa localisation, etc.

Évaluation des résultats

Notre démarche de développement E-CARe permet de fournir une solution de conception qui présente un ensemble d'avantages :

- L'identification des besoins ubiquitaires se fait à la suite d'une étude des besoins des utilisateurs. Ainsi, les fonctionnalités ubiquitaires introduites sont celles souhaitées par l'utilisateur. De plus, ces fonctionnalités sont attribuées de façon organisée dépendante des catégories des usagers et de leurs préférences.
- La démarche fournit une base commune de métamodèles et d'approches pour la construction des modèles des systèmes. Ainsi, l'usage de cette démarche sur différents systèmes favorise la production de systèmes interopérables qui communiquent plus facilement étant donné qu'ils utilisent les mêmes métamodèles. L'interopérabilité est nécessaire dans les systèmes ubiquitaires qui interagissent avec plusieurs dispositifs et font appel à différents services. L'approche IDM est utilisée pour transformer des modèles.
- Les modèles ubiquitaires sont produits indépendamment des modèles techniques et fonctionnels ce qui facilite la tâche de maintenance et de modification.
- L'usage d'une approche orientée évènements assure le dynamisme de l'application et la garantie d'un temps court pour la réception de l'information et la réaction plus rapide aux différentes situations. En effet, les évènements sont reliés à leurs actions et ciblent directement les composants nécessaires en évitant la circulation classique et hiérarchique de l'information qui fait perdre du temps inutilement.
- Finalement, le travail de conception est mieux organisé et un nouveau rôle d'analyste-concepteur de contexte est créé pour prendre en charge les spécifications ubiquitaires. Jusqu'à présent ces spécifications sont prises en charge soit par les spécialistes métier, soit par les spécialistes techniques sans délimiter les répercussions du travail de chacun sur les modèles de l'autre. De plus, notre démarche s'appuie sur l'approche générale de développement de type 2TUP et plus particulièrement la démarche Symphony, ce qui en facilite l'apprentissage.

En contre partie, notre démarche présente certaines limites liées principalement à la non pratique de la démarche pour des projets différents, complets et diversifiés. En effet, les applications peuvent présenter des besoins différents qui permettent de mieux raffiner la démarche pour présenter des comportements différents en fonction des applications. Certaines applications sont mono-utilisateurs alors que d'autres sont destinées à plusieurs utilisateurs ce qui doit permettre des spécifications différentes de contexte. Dans le premier cas, l'application est privée et le contexte est unique alors que dans le deuxième cas, le contexte diffère en fonction de la personne et son stockage est inutile après la fin du service.

De même certaines applications sociales et industrielles présentent des besoins d'adaptation différents et des natures différentes de contexte. Ainsi, les modèles contextuels de ces applications peuvent varier et il est intéressant de prendre en compte les spécificités de chaque domaine pour créer des fragments ou des phases spécialisées de la démarche E-CARe.

D'un autre point de vue, cette démarche néglige l'aspect "concurrence des règles". En effet, les règles ubiquitaires sont identifiées et implémentées sans se soucier du risque de concurrence résultant de règles à effets opposés pour une même situation où des règles qui se répètent avec les mêmes effets et des formulations différentes. Par exemple, en cas d'accident, une règle peut prévoir l'envoi d'un véhicule spécialisé pour dépanner les passagers alors qu'une autre peut prévoir la mise en place d'un véhicule de remplacement pour le même trajet. Ainsi, il convient dans la démarche d'étudier ces différentes règles et de proposer un mécanisme pour la détection des règles concurrentes et la résolution de la concurrence. Ceci constitue l'une de mes perspectives de recherche.

Perspectives de recherche

Comme dit précédemment, la démarche proposée dans cette thèse peut être améliorée en prenant en compte la concurrence des règles et considérant les spécificités de certaines applications pour mieux guider le travail. Cependant, il existe de nouvelles pistes de recherche qui peuvent améliorer davantage encore les démarches d'ingénierie des SI ubiquitaires.

D'un point de vue technique, ce travail de recherche peut être amélioré et mis en valeur par la création d'une plateforme complète de développement des applications ubiquitaires contenant des outils de modélisation, de transformation des modèles et mettant en œuvre la démarche de développement.

L'apprentissage au temps d'exécution (at runtime) trouve des usages intéressants dans les systèmes ubiquitaires où il est possible de surveiller les utilisateurs, de détecter leurs réponses à certaines situations et de les mémoriser. La réaction appropriée à une situation similaire peut alors être proposée à l'utilisateur. Ainsi, le système définit par lui même ses besoins d'adaptation par apprentissage de l'utilisateur de l'application. La mise en place d'un tel mécanisme d'apprentissage doit être considérée dans la démarche en définissant les besoins d'adaptation qui peuvent présenter des points d'ouverture paramétrables en fonction de l'utilisateur. Le système doit être capable de détecter les situations et leurs différences ainsi que les réactions des utilisateurs. Ces besoins d'adaptation peuvent ensuite être transformés au moment de l'exécution en des règles d'adaptation modifiables.

L'ingénierie des besoins ubiquitaires représente aussi une thématique intéressante à développer pour améliorer la gestion des projets ubiquitaires. Jusqu'à présent les SI ubiquitaires ne bénéficient pas d'approches d'ingénierie des besoins qui leurs sont dédiées et qui permettent de se concentrer sur l'identification des caractéristiques ubiquitaires attendues d'un système. Notre démarche présente juste une introduction à ce thème qui est traité par une approche intentionnelle. Il reste un travail de recherche important à effectuer par les spécialistes d'ingénierie des besoins. Ainsi, il est nécessaires de tester différents prototypes d'applications ubiquitaires auprès d'utilisateurs de différentes catégories sociales et professionnelles pour identifier les attentes et les prises de positions par rapport à ces systèmes. Il est possible d'ailleurs d'identifier de nouvelles fonctions ubiquitaires et de nouveaux besoins d'usages.

Finalement, les méthodes Agiles connaissent aujourd'hui un succès dans la réalisation des projets de développement. Nous considérons ce domaine combiné avec le développement des applications ubiquitaires comme une perspective importante de recherche. En effet, il est possible de combiner les pratiques Agiles dans notre démarche pour construire une vision générale sur le déroulement du projet et faire une estimation temporelle de chaque phase et de l'organisation des différentes itérations.

Bibliographie

[Aasman, 2008] Aasman, J. (2008). Unification of geospatial reasoning, temporal logic and social network analysis in event based systems. *In Distributed Event Based Systems*, Rome, Italy. ACM.

[Achilleos *et al.*, 2010] Achilleos, A., Yang, K. et Georgalas, N. (2010). Context modeling and a context-aware framework for pervasive service creation : A model driven approach. *Pervasive and Mobile Computing*, 6:281–296.

[Adaikkalavan et Chakravarthy, 2006] Adaikkalavan, R. et Chakravarthy, S. (2006). How to use events and rules for supporting role based security. *In 17th international conference on database and expert Systems Applications DEXA*. IEEE.

[Adi et Etzion, 2004] Adi, A. et Etzion, O. (2004). Amit : The situation manager. *The VLDB Journal*, 13(2):177–203.

[Al-Kahtani et Sandhu, 2002] Al-Kahtani, M. et Sandhu, R. (2002). A model for attribute based user role assignement. *In 18th Annual Computer Security Application Conference*. IEEE.

[Allen, 1991] Allen, J. (1991). Time and time again : the many ways to represent time. *In International Journal of Intelligent Systems*.

[Ammon *et al.*, 2007] Ammon, R. V., Silberbauer, C. et Wolff, C. (2007). Domain specific reference models for event patterns - for faster developing of business activity monitoring applications. *In VIPSI 2007 Lake Bled*, Slovenia.

[Anand et Mobasher, 2005] Anand, S. et Mobasher, B. (2005). Intelligent techniques for web personalization. *In ITWP*, LNAI 3169, pages 1–36. Springer.

[APES, 2004] APES (2004). Les groupes focalisés. fiche méthodologique f4. Rapport technique, Université de Lièges.

[Appleton, 2000] Appleton, B. (2000). Patterns and software : Essential concepts and terminology.

[Ardissono *et al.*, 2007] Ardissono, L., Furnari, R., Goy, A., Petrone, G. et Segnan, M. (2007). *Web Engineering*, chapitre Context-Aware Workflow Management, pages 47–52. Springer Berlin.

[Augustin *et al.*, 2006] Augustin, I., Yamin, A., da Silva, L., Real, R., Frainer, G. et Geyer, C. (2006). Isamadapt : abstractions and tools for designing general purpose pervasive applications. *Software-Practice and Experience*, 36:1231–1256.

[Axel Kern, 2005] Axel Kern, C. W. (2005). Rule support for role based access control. *In SACMAT'05*. ACM.

[Ayed *et al.*, 2007] Ayed, D., Delanote, D. et Berbers, Y. (2007). Mdd approach for the development of context-aware applications. *In CONTEXT'07*. Springer.

[Balabanovic et Shoham, 1997] Balabanovic, M. et Shoham, Y. (1997). Fab : content based collaborative recommandation. *Communication of the ACM*, 40(3):66–72.

[Bellavista *et al.*, 2003] Bellavista, P., Montanari, R. et Tibaldi, D. (2003). Cosmos : A context centric access control middelware for mobile environments. *In MATA'03*, LNCS 2881, pages 77–88. Springer.

[Belotti *et al.*, 2004] Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M. et Palinginis, A. (2004). Modelling context for information environments. , springer (2004). *In UMICS'2004*. Springer.

[Ben Cheikh, 2008] Ben Cheikh, A. (2008). Une méthode de rétro-ingénierie des processus métier basée sur un métamodèle multi-vues. Mémoire de D.E.A., Université Joseph Fourier, Grenoble, France.

[Ben Cheikh *et al.*, 2010a] Ben Cheikh, A., Front, A., Coulondre, S. et Giraudin, J.-P. (2010a). Event Based Modeling for Context-Reactive Information Systems. *In 5th International Conference on Signal Image Technology and Internet Based Systems (SITIS'10)*, Kuala Lumpur, Malaysia.

[Ben Cheikh *et al.*, 2010b] Ben Cheikh, A., Front, A., Coulondre, S. et Giraudin, J.-P. (2010b). Event-based modeling for ubiquitous information systems. *In 6èmes journées francophones Ubiquité et Mobilité (UbiMob'10)*, Lyon, France.

[Ben Cheikh *et al.*, 2010c] Ben Cheikh, A., Front, A., Coulondre, S. et Giraudin, J.-P. (2010c). Une modélisation événementielle des SI ubiquitaires. *In 28ème Congrès INFORSID*, Marseille, France.

[Ben Cheikh *et al.*, 2012] Ben Cheikh, A., Front, A., Giraudin, J.-P. et Coulondre, S. (2012). An engineering method for context-aware and reactive systems. *In Sixth International Conference on Research Challenges in Information Science, RCIS*, Valence, Espagne. IEEE. Best Paper Award.

[Ben Cheikh *et al.*, 2010d] Ben Cheikh, A., Front, A. et Rieu, D. (2010d). *Enterprise Information Systems Design, Implementation and Management : Organizational Applications*, chapitre Reverse-Engineering Enterprise Business Processes, pages 98–116. Information Science Reference, IGI Global.

[Ben Cheikh *et al.*, 2009a] Ben Cheikh, A., Rieu, D. et Front, A. (2009a). A method for business process reverse-engineering based on a multi-view metamodel. *In Conference on ENTERprise Information Systems (CENTERIS'2009)*. Ofir, Portugal.

[Ben Cheikh *et al.*, 2009b] Ben Cheikh, A., Saidi, R., Front, A. et Rieu, D. (2009b). Variability integration in multi-view business process design. *In Knowledge Management and Innovation in Advancing Economies - 13th International Business Information Management Association (IBIMA)*, Marrakech, Maroc.

[Ben Cheikh *et al.*, 2009c] Ben Cheikh, A., Tourasse, Y., Front, A., Coulondre, S. et Giraudin, J.-P. (2009c). Vers une approche orientée processus métier pour les si de transport. *In CoGIST, Première Conférence Francophone sur les Technologies de l'Information, de la Communication et de la Géolocalisation dans les Systèmes de Transports*, Saint Quay Portrieux, France.

[Bettini *et al.*, 2009] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganthan, A. et Riboni, D. (2009). A survey of context modeling and reasoning techniques. *Pervasive and Mobile Computing*, 6:161–180.

[Boukhebouze, 2010] Boukhebouze, M. (2010). *Gestion de changement et vérification formelle de PM : une approche orientée règles*. Thèse de doctorat, INSA de Lyon.

[Bouzeghoub et Kostadinov, 2005] Bouzeghoub, M. et Kostadinov, D. (2005). Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. *In Actes de la*

seconde édition de la Conférence en Recherche d'Information et Applications (CORIA), Grenoble, France.

[Brown *et al.*, 1997] Brown, P., Bovey, J. et Chen, X. (1997). Context-aware applications : from the laboratory to the marketplace. *IEEE Personal Communication*, 4(5):58–64.

[Brésillon et Pomerol, 1999] Brésillon, P. et Pomerol, J. (1999). Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain, Rubrique "Théories et Méthodologies"*, 62(3):223–246.

[Business Rule Group, 2000] Business Rule Group, B. (2000). *Defining Business Rules, what are they really ?*

[Bézivin et Jouault, 2006] Bézivin, J. et Jouault, F. (2006). Using atl for checking models. *Electronic Notes in Theoretical Computer Science*, 152:69–81.

[Calvary *et al.*, 2003] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. et Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15:289–308.

[Chaari *et al.*, 2007] Chaari, T., Ejigu, D., Laforest, F. et Scuturici, V. M. (2007). A comprehensive approach to model and use context for adapting applications in pervasive environments. *The Journal of Systems and Software*, 80:1973–1992.

[Chakravarthy et Jiang, 2009] Chakravarthy, S. et Jiang, Q. (2009). *Stream Data Processing : A Quality of Service Perspective*, chapitre NFMi : an inter-domain network fault management system, pages 167–186. Springer, US.

[Chakravarthy *et al.*, 1994] Chakravarthy, S., Krishnaprasad, V., Anwar, E. et Kim, S. (1994). Composite events for active databases : semantics, contexts and detection. *In 20th VLDB conference Santiago, Chile*.

[Chen *et al.*, 2004a] Chen, H., Finin, T. et Joshi, A. (2004a). Semantic web in the context broker architecture. *In Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications PerCom'04*. IEEE Computer Society.

[Chen *et al.*, 2004b] Chen, H., Perich, F., Finin, T. et Joshi, A. (2004b). Soupa : Standard ontology for ubiquitous and pervasive applications. *In 1st Annual International Conference on Mobile and Ubiquitous Systems, MobiQuitous 2004*. IEEE Computer Society.

[Chevalier *et al.*, 2007] Chevalier, M., Julien, C., Soulé-Dupuy, C. et Vallès-Parlangeau, N. (2007). personalized information access through flexible and interoperable profiles. *In WISE Workshops*, LNCS 4832, pages 374–385. Springer.

[Cilia *et al.*, 2001] Cilia, M., Bornhovd, C. et Buchmann, A. (2001). Moving active funtionality from centralized to open distributed heterogeneous environments. *In* et al., C. B., éditeur : *CoopIS'01*, pages 195–211. Springer.

[Cipriani *et al.*, 2011] Cipriani, N., Wieland, M., Grobmann, M. et Nicklas, D. (2011). Tool support for the design and management of context models. *Information Systems*, 36:99–114.

[Coad, 1992] Coad, P. (1992). Object oriented patterns. *Communication Of ACM*, 35(9):152–159.

[Coutaz *et al.*, 2005] Coutaz, J., Crowley, J. et S. Dobson, D. G. (2005). Context is key. *Communication of the ACM*, 48(3):49–51.

[Dardenne *et al.*, 1993] Dardenne, A., Lamsweerde, A. et Fickas, S. (1993). Goal directed requirements acquisition. *Science of computer Programming*, 20:3–50.

[de Farias *et al.*, 2007] de Farias, C. R. G., Leite, M. M., Calvi, C. Z., Pessoa, R. M. et Pereira Filho, J. G. (2007). A MOF metamodel for the development of context-aware mobile applications. *In SAC'07*, Seoul, Korea. ACM.

[Dey, 2001] Dey, A. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7.

[D'Souza et Wills, 1998] D'Souza, D. et Wills, A. C. (1998). *Objects, Components and Frameworks with UML : the Catalysis Approach*. Addison-Wesley.

[Filho et Martin, 2009] Filho, J. et Martin, H. (2009). A generalized access control model for pervasive environments. *In SPRINGL'09*. ACM.

[Fuchs *et al.*, 2005] Fuchs, F., Hochstatter, I. et Krause, M. (2005). A metamodel approach to context information. *In 3rd International conference on Pervasive Computing and Communication Workshop (PerCom 2005 workshops)*.

[Giannakopoulos et Palpanas, 2009] Giannakopoulos, G. et Palpanas, T. (2009). Adaptivity in entity subscription services. *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, Computation World*, 0:61–66.

[Godet-Bar, 2009] Godet-Bar, G. (2009). *Spécification et outillage d'une méthode de conception des systèmes de réalité mixte*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble.

[Goedertier et Vanthienen, 2007] Goedertier, S. et Vanthienen, J. (2007). Declarative process mining with business vocabulary and business rules. *In OTM workshops*, volume 4805 de *LNCS 4805*, pages 603–612. Springer.

[Golden et Powell, 2000] Golden, W. et Powell, P. (2000). Towards a definition of flexibility : in search of the holy grail ? *Omega*, 28(4):373 – 384.

[Gu *et al.*, 2004] Gu, T., Wang, X., Pung, H. et Zhang, D. (2004). An ontology-based context model in intelligent environments. *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, California, USA.

[Guerrero García *et al.*, 2009] Guerrero García, J., Lemaigre, C., Vanderdonckt, J. et González Calleros, J. (2009). *Computer-Aided Design of User Interfaces VI*, chapitre Model-Driven Engineering of Workflow User Interfaces. Springer-Verlag London.

[Hariri *et al.*, 2009] Hariri, M., Lepreux, S., Tabary, D. et Kolski, C. (2009). Principes et étude de cas d'adaptation d'ihm dans les si en fonction du contexte d'interaction de l'utilisateur. *Revue ISI : Networking and Information Systems*, 14:141–162.

[Hassine, 2005] Hassine, I. (2005). *Spécification et formalisation des démarches de développement à base de composants : la démarche Symphony*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble.

[Hassine *et al.*, 2005] Hassine, I., Rieu, D., Bounaas, F., Jausseran, E., Front, A. et Giraudin, J. (2005). *Ingénierie des composants*, chapitre Méthodes de développement centrées composants. Vuiberts.

[Henricksen, 2003] Henricksen, K. (2003). *A Framework for context-aware pervasive computing applications*. Thèse de doctorat, University of Queensland, Queensland, Australia.

[Henricksen et Indulska, 2006] Henricksen, K. et Indulska, J. (2006). Developing context-aware pervasive computing applications : models and approach. *Pervasive and Mobile Computing*, 2.

[Hong *et al.*, 2009] Hong, J., Suh, E. et Kim, S. (2009). Context-aware systems : A literature review and classification. *Expert Systems with Applications*, 36:8509–8522.

[Hull *et al.*, 1997] Hull, R., Neaves, P. et Bedford-Roberts, J. (1997). Towards situated computing. *In The First International Symposium on Wearable Computers*, Cambridge.

[ISO-9241-11, 1998] ISO-9241-11 (1998). *Exigences ergonomiques pour travail de bureau avec terminaux à écrans de visualisation (TEV) - Partie 11 : lignes directrices relatives à l'utilisabilité*.

[Jausseran, 2005] Jausseran, E. (2005). démarche symphony étendue, formalisation et expérimentation sur un système d'information hospitalier. Mémoire d'ingénieur cnam, Conservatoire National des Arts et des Métier de Grenoble.

[Joshi *et al.*, 2005] Joshi, J., Bertino, E., Latif, U. et ghafoor, A. (2005). A generalized temporal role-based access control model. *IEEE Transactionon Knowledge and Data Engineering*, 17(1).

[Kapitsaki *et al.*, 2009] Kapitsaki, G., Kateros, D., Prezerakos, G. et Venieris, I. (2009). Model-driven development of composite context-aware web applications. *Information and Software technology*, 51:1244–1260. Elsevier.

[Kappel *et al.*, 2003] Kappel, G., Proll, B., Retschitzegger, W. et Schiwger, W. (2003). Customization for ubiquitous web applications - a comparaison of approaches. *International Journal of Web Engineering Technology*.

[Kappel *et al.*, 2001] Kappel, G., Retschitzegger, W. et Schwinger, W. (2001). Modeling ubiquitous web applications - the wuml approach. *In Proceedings of the International Workshop on Data Semantics in Web Information Systems*, Kyoto, Japan.

[Kim *et al.*, 2005] Kim, Y., Mon, C., Jeong, D., Lee, J., Song, C. et Baik, D. (2005). Context -ware access control mechanism for ubiquitous applications. *In AWIC*, LNAI 3528, pages 236–242. Springer.

[Klyne *et al.*, 2004] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. et Tran, L. (2004). Composite capability/preference profiles (cc/pp) : Structure and vocabularies. Rapport technique, W3C Recommendation.

[Kolos-Mazuryk *et al.*, 2005] Kolos-Mazuryk, L., Poulisse, G. et van Eck, P. (2005). Requirements engineering for pervasive services. *In Workshop on Building Software for Pervasive Computing Held in conjunction with OOPSLA'05*, San Diego, USA.

[Kong *et al.*, 2009] Kong, J., Jung, J. et Park, J. (2009). Event-driven service coordination for business process integration in ubiquitous enterprise. *Computers and Industrial Engineering*.

[Kruchten, 2003] Kruchten, P. (2003). *The Rational Unified Process : An introduction*. Addison Wesley Professional, third edition édition.

[Kulkarni et Tripathi, 2008] Kulkarni, D. et Tripathi, A. (2008). Context aware role based access control in pervasive computing systems. *In SACMAT'08*. ACM.

[Kumar *et al.*, 2002] Kumar, A., Karnik, N. et Chafle, G. (2002). Context sensitivity in role based access control. *ACM SIGOPS operating system review*.

[Kumar et Yao, 2009] Kumar, A. et Yao, W. (2009). Process materialization using templates and rules to design flexible process models. *In RuleML 2009*. Springer.

[Lakshmanan *et al.*, 2009] Lakshmanan, G., Rabinovich, Y. et Etzion, O. (2009). A stratified approach for supporting high throughput event processing applications. *In Distributed Event Based Systems*.

[Levina et Stantchev, 2009] Levina, O. et Stantchev, V. (2009). realizing event driven SOA. *In Fourth International conference on Internet and Web Applications and Services*. IEEE.

[Lieberman, 1995] LIEBERMAN, H. (1995). Letizia : An agent that assists web browsing. *In 14th International Joint Conference on Artificial Intelligence*, pages 924–929, Montreal Canada.

[List et Korherr, 2006] LIST, B. et KORHERR, B. (2006). An evaluation of conteptual business process modeling languages. *ACM*.

[Luckham, 2002] LUCKHAM, D. (2002). *The power of events : an introduction to complex event processing in enterprise systems*. Addison Wesley.

[Michelson, 2006] MICHELSON, B. M. (2006). Event-driven architecture overview : Event-driven soa is just part of the eda story. Rapport technique, Seybold Group.

[Michlmayr *et al.*, 2008] MICHLMAYR, A., ROSENBERG, F., LEITNER, P. et DUSTDAR, S. (2008). Advanced event processing and notifications in service runtime environments. *In Distributed Event Based Systems*, Rome, Italy. ACM.

[Mladenic, 1996] MLADENIC, D. (1996). personal web watcher : Implementation and design. Rapport technique, department of Intelligent Systems, J. stefan University, Slovenia.

[Modafferi *et al.*, 2005] MODAFFERI, S., BENATALLAH, B., CASATI, F. et PERNICI, B. (2005). *Mobile Information Systems II*, chapitre A Methodology for Designing and Managing Context-Aware Workflows, pages 91–106. Springer.

[Morley *et al.*, 2005] MORLEY, C., HUGUES, J., LEBLANC, B. et HUGUES, O. (2005). *Processus Métiers et SI*. Dunod ; Paris.

[Mossakowski *et al.*, 2003] MOSSAKOWSKI, T., DROUINEAUD, M. et SOHR, K. (2003). A temporel-logic extension of role based access control covering dynamic separation of duties. *In 10th International Symposium on Temporel Representation and Reasoning and Fourth international Conference on Temporal Logic*. IEEE.

[Moukas, 1997] MOUKAS, A. (1997). Amalthaea : Information discovery and filtering using a multi-agent evolving ecosystem. *Applied Artificial Intelligence*, 11(5):437–457.

[Muhl *et al.*, 2006] MUHL, G., FIEGE, L. et PIETZUCH, P. (2006). *Distributed Event Based Systems*. Springer-Verlag Berlin Heidelberg.

[Muñoz *et al.*, 2006] MUÑOZ, J., VALDERAS, P., PELECHANO, V. et PASTOR, O. (2006). Requirements engineering for pervasive systems. a transformational approach. *In RE'06*, pages 344–345.

[Nagargadde *et al.*, 2005] NAGARGADDE, A., VARADARAJAN, S. et RAMAMRITHAM, K. (2005). Semantic characterization of real world events. *In International Conference on Database Systems for Advanced Applications*, Berlin Heidlberg. Springer-Verlag.

[Narendra *et al.*, 2005] NARENDRA, N., UMESH, B., NANDY, S. et KALAPRIYA, K. (2005). Functional and architectural adaptation in pervasive computing environments. *In Proceedings of the Third international workshop on Middleware for pervasive and ad-hoc computing*, Grenoble, France. ACM.

[Norrie, 1993] NORRIE, M. (1993). An extended entity-relationship approach to data management in object- oriented systems. *In 12th International Conference on the Entity-Relationship Approach*.

[Otzturk et Aamodt, 1997] OTZTURK, P. R. et AAMODT, A. (1997). Towards a model of context for case-based diagnostic problem solving. *In Proceedings of the interdisciplinary conference on modeling and using context*, pages 198–208, Rio de Janeiro.

[Paschke et Vincent, 2009] PASCHKE, A. et VINCENT, P. (2009). A reference architecture for event processing. *In Distributed Event Based Systems*, Nashville USA. ACM.

[Perich *et al.*, 2004] PERICH, F., JOSHI, A., FININ, T. et YESHA, Y. (2004). On data management in pervasive computing environments. *IEEE transaction on Knowledge and Data Engineering*, 16(5).

[Pichler, 2007] Pichler, M. (2007). *A Novel View on Requirements Engineering for Ubiquitous Computing : The Innovation Perspective*. Thèse de doctorat, Johannes Kepler Universitat Linz, Autriche.

[Pietzuch et Bacon, 2002] Pietzuch, P. et Bacon, J. (2002). Hermes : A distributed event middleware architecture. *In 1st Int. Workshop on on Distributed Event Based Systems*, pages 611–618.

[Pietzuch, 2004] Pietzuch, P. R. (2004). *Hermes : A Scalable Event-Based Middleware*. Thèse de doctorat, Queens' College University of Cambridge.

[Ploesser *et al.*, 2010] Ploesser, K., Recker, J. et Rosemann, M. (2010). Building a methodology for context-aware business processes : insight from an exploratory case study. *In 18ème European Conference on Information Systems*.

[Priego Roche, 2011] Priego Roche, L. M. (2011). *Modélisation intentionnelle et organisationnelle des systèmes d'information dans les organisations virtuelles*. Thèse de doctorat, Université de Grenoble, Grenoble, France.

[Regev et Wegmann, 2005] Regev, G. et Wegmann, A. (2005). A regulation-based view on business process and supporting system flexibility. *In CAISE workshop*, pages 91–98.

[Rey, 2005] Rey, G. (2005). *Contexte en Interaction Homme-Machine : le contexteur*. Thèse de doctorat, Université Joseph Fourier, Grenoble.

[Rocques et Vallée, 2002] Rocques, P. et Vallée, F. (2002). *UML 2 en action : de l'analyse des besoins à la conception J2EE*. Eyrolles.

[Rosemann *et al.*, 2008] Rosemann, M., Recker, J. et Flender, C. (2008). Contextualisation of business processes. *International Journal Business Process Integration and Management*, 3(1):47–60.

[Rozsnyai *et al.*, 2007] Rozsnyai, S., Schiefer, J. et Schatten, A. (2007). Concepts and models for typing events for event based systems. *In Distributed Event Based Systems*, Toronto Canada. ACM.

[Schafer *et al.*, 1999] Schafer, J., Konstan, J. et Riedl, J. (1999). Recommender systems in e-commerce. *In Conference on Electronic Commerce*. ACM.

[Schmidt *et al.*, 1999] Schmidt, A., Beigl, M. et Gellersen, H. (1999). There is more to context than location. *Comput Graphics*, 23(6):893–901.

[Schonenberg *et al.*, 2008] Schonenberg, H., Mans, R., Russell, N., Mulyar, N. et van der Aalst, W. M. P. (2008). Towards a taxonomy of process flexibility. *In CAiSE Forum*, pages 81–84.

[SelectPerspective, 2006] SelectPerspective (2006). An Agile process v 1.2 (whitepaper). Rapport technique, Select Business Solutions, Inc.

[Serral *et al.*, 2010] Serral, E., Valderas, P. et Pelechano, V. (2010). Towards the model driven development of context-aware pervasive systems. *Pervasive and Mobile Computing*, 6:254–280.

[Sharon et Etzion, 2008] Sharon, G. et Etzion, O. (2008). Event processing network model and implementation. Rapport technique, IBM.

[Sheng et Benatallah, 2005] Sheng, Q. et Benatallah, B. (2005). Contextuml : A uml based modeling language for model driven developement of context aware web services. *In International Conference on Mobile Services ICMB'05*, pages 206–212.

[Simons et Wirtz, 2007] Simons, C. et Wirtz, G. (2007). Modeling context in mobile distributed systems with the uml. *Journal of Visual Languages and Computing*, 18:420–439.

[Smanchat *et al.*, 2008] SMANCHAT, S., LING, S. et INDRAWAN, M. (2008). A survey on context-aware wokflow adaptation. *In MoMM 2008*, Linz, Austria. ACM.

[Sottet *et al.*, 2008] SOTTET, J.-S., CALVARY, G., COUTAZ, J. et FAVRE, J.-M. (2008). A model driven engineering approach for the usability of plastic user interfaces. *In* et AL., J. G., éditeur : *EIS 2008*, LNCS, pages 140–157. IFIP International Federation for Information Processing.

[Sousa *et al.*, 2007] SOUSA, K., MENDONÇA, H. et VANDERDONCKT, J. (2007). Towards method engineering of model-driven user interface development. *In* WINCKLER, M., JOHNSON, H. et PALANQUE, P., éditeurs : *TAMODIA 2007*, LNCS 4849, pages 112–125. Springer-Verlag Berlin Heidelberg.

[Strang et Linnhoff-Popien, 2004] STRANG, T. et LINNHOFF-POPIEN, A. (2004). A context modeling survey. *In Ubicomp, First International Workshop On Advanced Context Modeling, Reasoning and Management*.

[Strembeck et Neumann, 2004] STREMBECK, M. et NEUMANN, G. (2004). An integrated approach to engineer and enforce context constraints in rbac environments. *ACM transactions on Information and System security*, pages 392–427.

[Tchienehom, 2005] TCHIENEHOM, P. (2005). Profile semantics for personalized information access. *In Workshop on Personalization in the Semantic Web (PerSWeb'05)*, Edinburgh UK.

[Tchienehom, 2006] TCHIENEHOM, P. (2006). *Modélisation et Exploitation de Profils : Accès Sémantique à des Ressources*. Thèse de doctorat, Université de Toulouse I.

[Traetteberg et Krogstie, 2008] TRAETTEBERG, H. et KROGSTIE, J. (2008). Enhancing the usability of bpm-solutions by combining process and user-interface modelling. *In* STIRNA, J. et PERSSON, A., éditeurs : *PoEM 2008*, LNBIP, pages 86–97. IFIP International Federation for Information Processing.

[Valatkaite et Vasilecas, 2005] VALATKAITE, I. et VASILECAS, O. (2005). On business rule automation : The br-centric is development framework. *In* et AL., J. E., éditeur : *ADBIS 2005*, 3631, pages 349–364. Springer-Verlag.

[Vanderdonckt *et al.*, 2008] VANDERDONCKT, J., MENDONCA, H. et MASSO, J. M. (2008). Distributed user interfaces in ambient environment. *In AmI workshops*, pages 121–130. Springer-Verlag.

[Vieira *et al.*, 2010] VIEIRA, V., TEDESCO, P. et SALGADO, A. (2010). Designing context sensitive systems : An integrated approach. *In Expert Systems with Applications*. Elsevier.

[Wagner, 2005] WAGNER, G. (2005). Rule modeling and markup. *Reasoning Web*, 3564:251–274.

[Wapforum, 2001] WAPFORUM (2001). User agent profile. Rapport technique.

[Weiser, 1991] WEISER, M. (1991). The computer for the 21st century. *Scientific American*, pages 94–104.

[Wilikens *et al.*, 2002] WILIKENS, M., FERITI, S. et MASERA, M. (2002). A context related authorization and access control method based on rbac : A case study from the health care domain. *In SACMAT'02*. ACM.

[Xu *et al.*, 2008] XU, K., ZHU, M., ZHANG, D. et GU, T. (2008). Context-aware content filtering and presentation for pervasive and mobile information systems. *In Ambi-sys*.

[Yu, 1997] YU, E. (1997). Towards modeling and reasoning support from early-phase requirements engineering. *In IEEE 3th International Symposium on Requirements Engineering*, pages 226–235, Annapolis MD.

[Zambrano *et al.*, 2004] ZAMBRANO, A., GORDILLO, S. et JAUREGUIBERRY, I. (2004). Aspect-based adaptation for ubiquitous software. *In Mobile and ubiquitous info*, pages 215–226. Springer.

[Zang *et al.*, 2008] Zang, C., Fan, Y. et Liu, R. (2008). Architecture, implementation and application of complex event processing in enterprise information systems based on rfid. *In Information System Frontier*, volume 10, pages 543–553. Springer.

[Zhang *et al.*, 2005] Zhang, D., Gu, T. et Wang, X. (2005). Enabling context-aware smart home with semantic technology. *International Journal of Human-friendly Welfare Robotic Systems*, 6(4): 12–20.

[Zimmermann *et al.*, 2007] Zimmermann, A., Lorenz, A. et Oppermann, R. (2007). An operational definition of context. *In Modeling and using context*, pages 558–571. Springer Berlin Heidelberg.

Webographie

[1] ACTIF. (http ://www.its-actif.org/).

[2] Aleri. (http ://www.sybase.com/products/financialservicessolutions/complex-event-processing).

[3] ATL. (http ://eclipse.org/atl/).

[4] Aurora. (http ://www.cs.brown.edu/research/aurora/).

[5] Borealis. (http ://www.cs.brown.edu/research/borealis/).

[6] BPMN. (http ://www.omg.org/spec/BPMN/2.0/PDF).

[7] Cayuga. (http ://www.cs.cornell.edu/bigreddata/cayuga/).

[8] CHOUETTE. (http ://www.certu.fr/fr/IMG/pdf/NotePresentationOutilChouetteAvril08.pdf).

[9] IREM de Lyon. Introduction à l'algorithme de dijkstra appliqué à la recherche d'itinéraire. (http ://irem-fpb.univ-lyon1.fr/feuillesprobleme/feuille6/enonces/courseazero/dijkstra.html).

[10] DéSIT. (http ://projet-desit.imag.fr).

[11] Esper. (http ://esper.codehaus.org/).

[12] IBM Websphere Business Event. (http ://www-01.ibm.com/software/integration/wbe/).

[13] Marvelig. (https ://marvelig.imag.fr).

[14] Objectiver. (http ://www.objectiver.com/).

[15] TIBCO Rendezvous. (http ://www.tibco.com/products/soa/messaging/rendezvous/default.jsp).

[16] RuleCore. (http ://www.rulecore.com/).

[17] sigma. (http ://sigma.imag.fr).

[18] STREAM. (http ://infolab.stanford.edu/stream/).

[19] trident. (http ://www.ertico.com/en/activities/completed_projects/trident_website.htm).

[20] Umanis. (http ://www.groupeumanis.com).

[21] XACML. (http ://www.oasis-open.org/committees/xacml/).

Liste des acronymes

2TUP 2 Track Unified Process

ATL Atlas Transformation Language

BAM Business Activity Monitoring

BPDM Business Process Definition Metamodel

BPEL Business Process Execution Language

BPM Business Process Management

BPMN Business Process Modeling Notation

CCL Continuous Computation Language

CC/PP Composite Capabilities/ Preference Profile

CEP Complex Event processing

CML Context Modeling Language

CMP Context Modeling Profile

DAC Discretionary Access Control

DéSIT Démarche d'ingénierie des Systèmes d'Information de Transport pervasifs, sécurisés et personnalisables

ECA Event Condition Action

ECAPE Evènement- Condition - Action - Post condition - post Evènement

E-CARe Engineering - Context Aware and Reactive systems

EDA Event Driven Architecture

EdBPM Event driven Business Process Management

EPA Event Processing Agent

EPL Event Processing Language

EPN Event Processing Network

EPU Event Processing Unit

GL Génie Logiciel

IDM Ingénierie Dirigée par les Modèles

IHM Interaction Homme-Machine

KAOS Knowledge Acquisition in automated specification

LIG Laboratoire Informatique de Grenoble

LIP / PC2S Laboratoire Interuniversitaire de Psychologie-Personnalité, Cognition, Changement Social - Grenoble

LIRIS Laboratoire d'infoRmatique en Image et Systèmes d'Information - Lyon
MAC Mandatory Access Control
MOF Meta Object Facility
MVC Model View Controller
OCL Object Constraint Language
OM Objet Métier
OMG Object Management Group
ORM Object-Role Modeling
OSGi Open Services Gateway initiative
OWL Web Ontology Language
PervML Pervasive system Modeling Language
PIM Platform Independent Model
PM Processus Métier
RBAC Role Based Access Control
RDF Resource Description Framework
RDFS Resource Description Framework Schema
RFID Radio Frequency IDentification
RM Règle Métier
RUP Rational Unified Process
SBVR Semantics of Business Vocabulary and Business Rules
SI Système d'Information
SII Système d'Information Interactif
SIM Système d'Information Mobile
SIT Système d'Information de Transport
SIV Système d'Information des Voyageurs
SOA Service Oriented Architecture
SPEM Software Process Engineering Metamodel
TBAC Task Based Access Control
UML Unified Modeling Language
WS-ECA Web Service - Event Condition Action
WUML Web Unified Modeling Language
XACML eXtensible Access Control Markup Language
XML eXtensible Markup Language
XPDL XML Process Definition Language
YAWL Yet Another Workflow language

Annexes

Annexe A

Couplage entre contexte et évènements [2]

Les évènements dans les SI permettent de réagir rapidement et en temps pratiquement réel aux changements du contexte du système. Pour mieux réagir à ces évènements il est important d'identifier d'une façon la plus complète possible ces évènements, leurs sources et leurs effets. Il est également indispensable de souligner l'utilisation des données contextuelles pour deux buts différents :

- pour permettre la personnalisation, en fournissant à l'utilisateur des services et des informations adaptés à son contexte et son profil.
- pour être proactif, en étant vigilant aux changements se produisant dans le contexte et réagir de façon appropriée aux évènements.

Dans cette perspective, cette annexe définit un métamodèle couplant évènements et contexte [Ben Cheikh *et al.*, 2010c]. Ce métamodèle, représenté par la figure A.1, est composé par trois vues représentant trois parties focalisant chacune sur des aspects différents : le contexte, les évènements et le couplage entre contexte et évènements.

Métamodèle de contexte

Le contexte d'une application ou d'un système est l'ensemble des ressources avec leurs profils (voir la vue A de la figure A.1). Dans le cas des SI de transport, les bus, les trams, les voyageurs, les arrêts, les informations météorologiques, les informations trafic et les fiches horaires sont des ressources. Chaque ressource possède un profil qui regroupe ses métadonnées. Seules les données dynamiques du profil sont susceptibles de changer alors que les données statiques sont rarement modifiées.

Les relations entre ressources sont nécessaires pour déduire suite à la production d'un évènement les différentes ressources qui peuvent être impactées ou intéressées par l'évènement. Zimmermann et al. [Zimmermann *et al.*, 2007] considèrent que les relations entre ressources sont importantes pour caractériser le contexte. Notre vision du contexte est différente de celle de [Zimmermann *et al.*, 2007], qui considère le contexte de chaque entité séparément des autres alors que nous considérons le contexte comme un groupement d'entités dont chacune contribue avec ses données de profil. Contrairement à cette approche, les relations entre ressources ne font pas partie du profil pour permettre au système de gérer plus facilement ces relations sans modifier les profils qui sont dans la plupart des cas maîtrisés par des systèmes distribués couplés aux ressources.

2. Cette combinaison de métamodèles avec le métamodèle de couplage était une base de travail pour initialiser cette recherche et a été utilisée par les expérimentations liées à notre approche. Suite à ces expérimentations, nous avons opté pour la séparation des deux métamodèles, le couplage étant repoussé à la phase de conception (section 5.6).

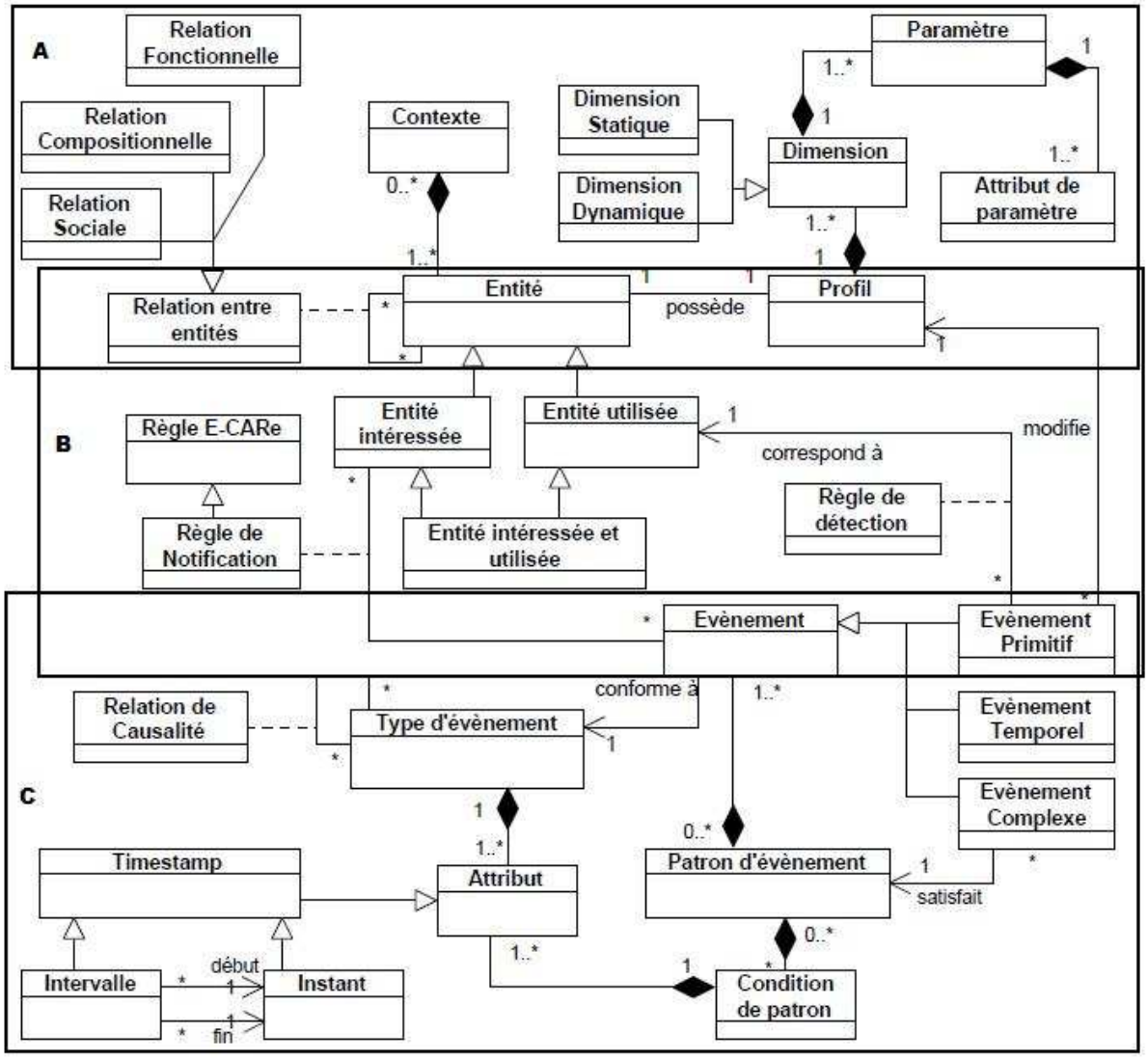


FIGURE A.1 – Métamodèle de couplage entre contexte et évènements

Trois types de relations peuvent exister entre les ressources : sociale, fonctionnelle et compositionnelle [Zimmermann *et al.*, 2007]. Les relations sociales concernent plutôt les ressources humaines qui sont représentées par les différentes personnes interagissant avec le SI. Elles permettent d'identifier pour une personne ses amis, ses collègues, ses voisins, etc. Les relations fonctionnelles mettent en évidence des liens d'utilisation : une ressource utilise une autre ressource. Finalement, les relations compositionnelles montrent des liens de type "partie à ensemble". Par exemple, la relation entre voyageur et dispositif est une relation fonctionnelle, alors que la relation entre un voyageur et ses amis est une relation sociale et la relation entre un arrêt et un réseau de transport est une relation compositionnelle. Dans chacun de ces cas, un évènement qui survient sur une ressource peut intéresser les ressources qui lui sont liées par une des relations citées ci-dessus.

Métamodèle d'évènements

La vue structurelle des évènements, représentée en figure A.1 vue C, permet d'assembler les différents concepts adoptés pour caractériser les évènements. Chaque évènement correspond à un type d'évènements (utilisation du patron Item-Description défini dans [Coad, 1992]).

La forme d'un évènement est mise en évidence par l'utilisation des types et des attributs d'évènements. Les relations entre évènements sont considérées grâce à l'utilisation des relations de causalité et d'agrégation (utilisation des évènements complexes pour composer des évènements). Ces relations sont prédéfinies dans le système entre types d'évènements. D'autres relations dépendent des instances des évènements, autrement dit, de la valeur des attributs des évènements. Ces relations sont définies par l'utilisation des règles de traitement qui permettent de détecter des situations particulières (par identification de patrons d'évènements) et de déduire les évènements qui en découlent. Les patrons d'évènements sont à leur tour prédéfinis dans le système. Les conditions sur un patron portent sur les attributs des évènements.

Le « timestamp » d'un évènement est un attribut commun à tous les évènements qui peut être soit un intervalle soit un instant. Il fait référence au temps d'occurrence de l'évènement ou à son temps de détection (selon la nature du système). L'identification des relations temporelles entre évènements se fait par comparaison de leurs timestamps pour déduire les ordonnancements et les séquences d'évènements dans le temps. Le traitement des évènements basés intervalle nécessite l'utilisation d'opérateurs temporels qui s'appuient sur l'utilisation des primitives d'intervalles définies par Allen dans sa logique temporelle [Allen, 1991].

Métamodèle de couplage

La vue B de la figure A.1 représente le modèle combiné de contexte et d'évènements sur lequel s'appuie notre approche. Un évènement du contexte correspondant à une ressource modifie l'état de la ressource, autrement dit modifie son profil. Cette approche classifie les ressources en trois catégories :

- les ressources utilisées sont les ressources supervisées par le SI et dont le changement d'état intéresse le système.
- les ressources intéressées sont des ressources avec lesquelles le SI communique pour livrer des évènements.
- les ressources intéressées et utilisées sont les ressources qui peuvent être à la fois supervisées et notifiées.

Ce modèle met en évidence les règles de notification entre les évènements et les ressources intéressées. Ces règles sont utilisées pour la livraison des évènements. Les règles de détection font le lien entre les évènements primitifs et les ressources qui leurs correspondent. L'identification des ressources intéressées se fait de manière automatique en utilisant les relations entre ressources ou les profils des ressources ou de manière semi-automatique en appliquant des règles E-CARe.

Le métamodèle de couplage ne permet pas seulement de lier l'aspect structurel à l'aspect dynamique, mais il permet aussi de lier les aspects conceptuels de l'application aux aspects techniques. En effet, les évènements sont fortement liés aux infrastructures physiques de l'application et mettent en évidence l'usage de composants physiques comme des capteurs et des systèmes de localisation. Ainsi, le lien entre contexte et évènements facilite la prise en compte des techniques ubiquitaires pour la collection et l'usage du contexte. Ce métamodèle est expérimenté dans l'annexe C.

Annexe B

Moteurs de traitement des évènements

Cette annexe a pour objectif de comparer deux moteurs d'évènements Esper et Cayuga que nous avons évalués afin d'implémenter le prototype d'une application d'assistance au voyageur.

B.1 Esper

Esper [11] est un projet Open Source développé en java par l'entreprise EsperTech et disponible sous la licence GNU GPL (à noter que le même projet est disponible en .NET sous le nom de NEsper). Esper, comme Cayuga, a pour objectif de détecter des patrons d'évènements complexes parmi un (ou plusieurs) flux d'évènements. Il a été conçu afin de répondre aux besoins toujours croissants de traitement de flux de données dans le monde de la finance, des réseaux, ou encore les applications utilisant des capteurs. Le langage d'interrogation est basé lui aussi sur le format SQL avec toutes les clauses standards. Esper utilise des outils très utiles pour l'expression des patrons d'évènements. Par exemple, il introduit la notion de fenêtre qui permet de définir une limite en nombre ou en temps des évènements à considérer lors d'une requête, ou encore la notion de pattern qui permet de définir un ordre d'arrivée des évènements (ex : "l'évènement B doit être précédé par l'évènement A, se traduit par le pattern "A $\rightarrow$ B"). Grâce à tous ces outils il devient relativement simple de définir, dans le langage d'interrogation d'évènements d'Esper, intitulé EPL, des patrons d'évènements.

B.2 Cayuga

Cayuga [7] est un système développé en C++ par l'université de Cornell située aux États Unis. C'est un logiciel libre disponible sous la licence BSD. Il a été conçu dans le but de « découvrir » des patrons d'évènements dans un flux important d'évènements. La particularité de Cayuga concerne l'utilisation d'automates non-déterministes pour faire correspondre les patrons avec les évènements reçus. Il utilise un langage d'interrogation appelé CEL (Complex Event Language). CEL est basé sur le format SQL en utilisant les clauses standards de ce dernier (Select, from, where ...) et en intégrant des clauses spécifiques (NEXT, FOLD) permettant de traiter des séquences d'évènements. Les requêtes sont écrites en CEL et elles sont ensuite traduites sous la forme d'automates non-déterministes.

B.3 Comparaison entre moteurs

Nous comparons ici un ensemble de moteurs de traitement des évènements avec Esper et nous faisons apparaître les avantages d'Esper.

Langage EPL/ License	Spécification des évènements			Spécification des règles					implémentation	
	représentation	héritage	attributs	types	opérateurs	condition	relation	livraison	stockage	outils
ESPER Open source	Objet Java, Map et XML	Oui, supertypes par Map	Simple, indexé, dictionnaire (Mapped)	Requête SQL like (fenêtre)	Comparaison, logique, arithmétique	Attributs d'évènements, accès BD	corrélation	souscription	Cache (résultats), In-memory BD	Moteur ESPER
CAYUGA	Tuple : schéma relationnel	non	Simple	Requête SQL like (fenêtre) Automates non-deterministes	Comparaison, logique, arithmétique	Attributs d'évènements	Corrélation, composition	souscription	In-memory BD	Moteur de requête Cayuga
RuleCore Open source	XML	non	Valeur unique (string, number, date)	ECA (script, event, change state)	Logique, séquence, temporel	Attributs d'évènements	composition	souscription	BD relationnelle (PostgreSQL)	Python, Qt, XML, Moteur ruleCore
DEAL	Objet Java	Oui	Valeur unique	ECA	Logique, comparaison, temporel	Attributs d'évènements		souscription		Serveur évènements Cassius
AMIT	XML Objet Java	Oui	Valeur unique (string, number, date)	Requête SQL like	Logique, temporel, conparaison, séquence, ...	Attributs d'évènements	Transformation (référence)	souscription		WebSphere Message Broker - CEP Detector Nodes
XChange	XML	Oui	Valeur unique et composée	ECA	Logique, temporel, conparaison, séquence, ...	Données persistentes		Souscription + filtrage par données persistante	Pas de sauvegarde	Xcerpt system

FIGURE B.1 – Table de comparaison entre moteurs d'évènements

Annexe C

Expérimentations

Les expérimentations réalisées sur les métamodèles de contexte comportent un ensemble de questionnaires à remplir par les sujets qui sont présentés dans cette annexe.

C.1 Questionnaire des pratiques

Le questionnaire des pratiques permet aux sujets de définir leurs visions des SI ubiquitaires et leurs habitudes de conception.

E-CARe

1. Avez vous l'habitude de concevoir des applications sensibles au contexte ?

○ 1. Oui ○ 2. Non

2. si oui, pouvez vous décrire une de ces applications ?

3. Si oui, comment modélisez vous le contexte d'une application ? quelles étapes suivez vous ?

4. Quelle définition donneriez vous au concept de "contexte" ?

5. A votre avis, pourquoi utiliser le contexte ? dans quels buts ?

6. A votre avis, quelles sont les principales difficultés pour modéliser le contexte ?

7. Pour vos applications, vous utilisez des méta-modèles ?

○ 1. pour chaque application ○ 2. parfois ○ 3. jamais

8. Globalement, que pensez vous des méta - modèles ?

9. Quels sont les avantages des méta - modèles ?

10. Quels sont les inconvénients des méta - modèles ?

11. Que pensez vous de l'utilisation d'un méta modèle pour modéliser le contexte ?

12. Prénom

13. année de naissance

14. Spécialité de recherche

C.2 Questionnaire Points Forts/Points Faibles

Le questionnaire des Points Forts/Points Faibles permet de récapituler les avis des sujets sur les métamodèles et de leur donner la liberté de s'exprimer.

TABLE C.1 – Comparaison entre les métamodèles de contexte

Donnez les	Points forts	Points faibles
sur les concepts du métamodèle de contexte		
sur les liens entre les concepts		
sur les concepts du métamodèle d'évènements		
sur la complétude des métamodèles		
sur la généricité des métamodèles		
sur l'utilisation des métamodèles proposés		

C.3 Exercices de l'expérience

Scénario : Application d'assistance au voyageur conçue pour un SI de transports publics Il s'agit d'une application mobile interagissant avec son environnement (SI de transport, bus, arrêts) et permettant de :

1. Effectuer des opérations classiques comme la consultation des horaires, la consultation des cartes du réseau ou le calcul d'itinéraire,
2. Consulter des horaires en temps réel,
3. Recevoir des alertes adaptables à ses besoins ou à sa situation géographique,
4. Exprimer des préférences sur les moyens et les lignes de transport ou même exprimer des habitudes de transport ou d'éventuelles situations d'empêchement (handicap, femme avec poussette, etc),
5. Être assisté pendant un trajet (signaler quand le tram arrive ou quand il faut monter ou descendre).

L'application doit fournir des services personnalisés en prenant en compte le contexte de l'usager tels que son profil, ses empêchements (handicap, bagage, malvoyant), ses préférences de mode de transport, sa localisation. Le système doit raisonner de manière intelligente pour pouvoir détecter des perturbations en raisonnant sur des évènements externes (météo, trafic, retard, manifestation, etc). Ensuite, il doit adapter l'assistance au contexte.

C.3.1 1er exercice

Construire un modèle de contexte en instanciant le métamodèle de la figure C.1.

1. Identifier les entités (une entité est toute partie interagissant avec le système pour consommer un service ou une information),

2. Identifier le profil de chaque entité (l'ensemble des paramètres caractéristiques utiles pour adapter le service d'information ou d'assistance),
3. Identifier les relations entre entités,
4. Identifier les données de l'environnement à prendre en compte pour adapter l'information délivrée.

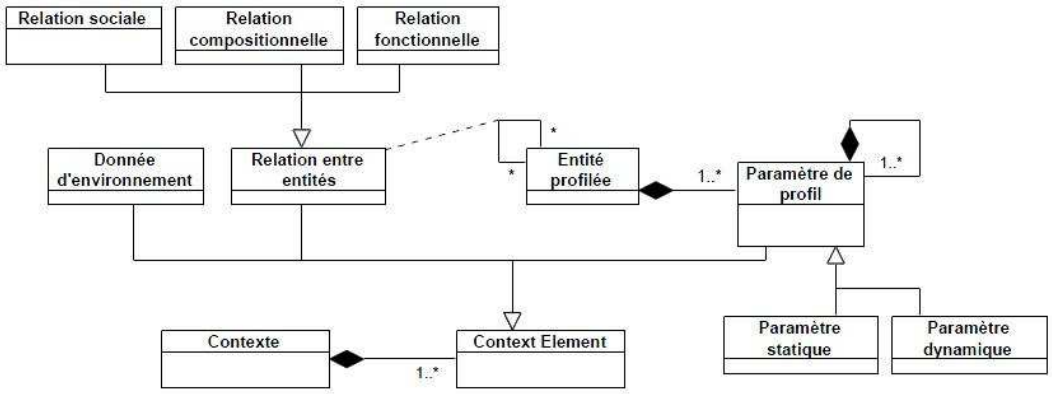


FIGURE C.1 – Métamodèle de contexte à instancier

C.3.2 2ème exercice

Construire un modèle d'évènements en instanciant le métamodèle de la figure C.2.

1. Identifier l'ensemble des évènements que peut recevoir l'application de son entourage ou du SI de transport (TAG),
2. Qualifier ces évènements (externes : proviennent d'un autre système, métier : proviennent d'une interaction avec d'autres composants du SI, utilisateur proviennent d'une interaction avec l'utilisateur),
3. Lier ces évènements aux ressources modifiées.

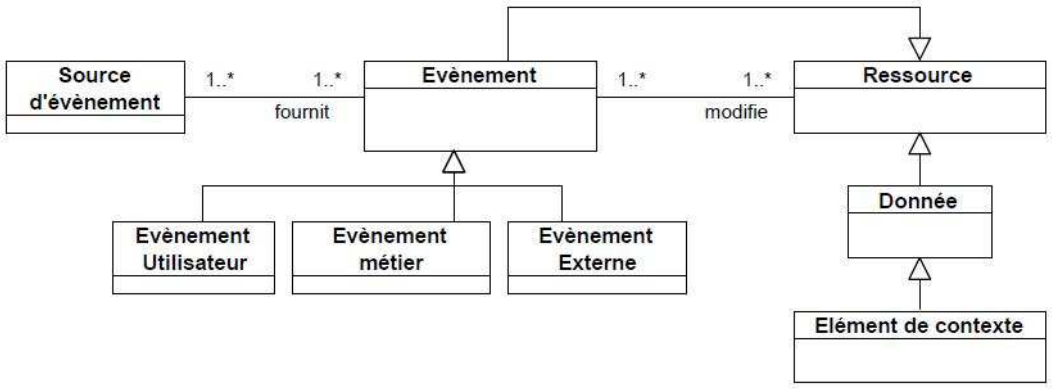


FIGURE C.2 – Métamodèle d'évènements à instancier

C.4 Exemples de questionnaires des pratiques remplis

E-CARe

Avez vous l'habitude de concevoir des applications sensibles au contexte ?

Ø 1.Oui ○ 2.Non

si oui, pouvez vous décrire une de ces applications ?

Un Système d'interrogation de parcs de capteurs hétérogènes!

Si oui, comment modélisez vous le contexte d'une application ? quelles étapes suivez vous ?

le contexte est un tupl de métadonnées. e.g. localisation, type de capteur...

Quelle définition donneriez vous au concept de "contexte" ?

Le contexte est un ensemble de données/informations relatif à l'environnement d'exécution des requêtes/applications dans les parcs de capteurs.

A votre avis, pourquoi utiliser le contexte ? dans quels buts ?

Dans le but d'adapter l'exécution des applications à l'environnement.

A votre avis, quelles sont les principales difficultés pour modéliser le contexte ?

- Identifier le besoin de l'utilisateur
- L'aspect continu!

Pour vos applications, vous utilisez des méta-modèles ?

○ 1.pour chaque application ○ 2.parfois ✓ 3.jamais

Globalement, que pensez vous des méta - modèles ?

Un méta modèle permet une abstraite et globale du système à modéliser.

Quels sont les avantages des méta - modèles ?

Simplifier les problèmes similaire!

Quels sont les inconvénients des méta - modèles ?

Trop abstrait dans certain cas, ils ne facilitent pas la modélisation de problème spécifique

Que pensez vous de l'utilisation d'un méta modèle pour modéliser le contexte ?

E-CARe

Avez vous l'habitude de concevoir des applications sensibles au contexte ?

☑ 1.Oui ○ 2.Non

si oui, pouvez vous décrire une de ces applications ?

Batiments intelligent : controle des dispositifs, éclairage, chauffage, etc...

Si oui, comment modélisez vous le contexte d'une application ? quelles étapes suivez vous ?

- modelisation dynamique, pas de model connu a l'avance, fusion d'information.
- entité, relation, faculte, interpretation (ma thèse)

Quelle définition donneriez vous au concept de "contexte" ?

Ensemble de faits qui sont pertinent pour la réalisation d'une tache donné.

A votre avis, pourquoi utiliser le contexte ? dans quels buts ?

Pour adapter les services proposé par un système, dans le but d'améliorer la "vie" de l'usagé et le décharger de certaines taches

A votre avis, quelles sont les principales difficultés pour modéliser le contexte ?

Le coté dynamique, (services de perception qui apparaissent/disparaissent), la difference d'interpretation de ce contexte

Pour vos applications, vous utilisez des méta-modèles ?

○ 1.pour chaque application ○ 2.parfois ☑ 3.jamais

Globalement, que pensez vous des méta - modèles ?

- La même chose que les meta-meta modèle.
- plutôt exper que end-user.

Quels sont les avantages des méta - modèles ?

Je ne les utilise pas vraiment, je ne pense pas être objectif.
La simplicité lors de la phase de conception, et d'adaptation (system plus souple)

Quels sont les inconvénients des méta - modèles ?

- modèle → meta-modèle → → meta meta-model.
- intractable pour les utilisateur.
- Souvent indique un mauvais model.

Que pensez vous de l'utilisation d'un méta modèle pour modéliser le contexte ?

Je suis curieux, cela m'interesse, cela a un interet coté concepteur.

C.5 Exemples de questionnaires Points Forts/Points Faibles remplis

Donnez les	Points Forts	Points Faibles
sur Les concepts du meta modèle de contexte	COMPLET - EXHAUSTIF.	PAS INTUITIF SUR CERTAINS CONCEPTS Evt temporel ? Relation sociale ?
sur Les concepts relatifs aux vues		
sur Les liens entre les concepts	BIEN	
Sur La complétude du méta modèle	COMPLET.	
sur La généricité	SUFFISAMMENT GENERIQUE	NOTALITE D'OBTENTION DU CONTEXTE D'EXECUTION.
sur Utilisation des méta modèles proposés	BONNE REPRESENTATION DU CONTEXTE	COMPLEXE - NECESSITE D'UNE FORMATION AU META-MODELE GUIDE D'UTILISATION

FIGURE C.3 – Questionnaire Points forts/points faibles rempli par le sujet 4

Donnez les	Points Forts	Points Faibles
sur Les concepts du meta modèle de contexte	Je pense je se peut être une bonne guide pour le concepteur, car il est assez complet	Il m'ont manqué des liens entre les différent méta-modèles. "Cognitive Fit" Superposer des concepts pourra aider.
sur Les concepts relatifs aux vues	Beaucoup de concepts intéressantes et variées.	Difficiles à comprendre de prime abord. Il faut accentuer sur la définition et clarification des concepts
sur Les liens entre les concepts	Assez claires en général.	J'aurais mis des éléments UML 2 qui clarifié le sens de s relations. L'heritage été forcé dans certains cas.
Sur La complétude du méta modèle	Je le trouve complet mais je pense pas que je puisse avoir les compétences pour le analyser à profondeur	le terme contexte reste très générique
sur La généricité	Ok. mais abuse de hierarchie à mon avis.	Certain termes comme "Relation Sociale" ou "Données environnements" sont trop détaillé.
sur Utilisation des méta modèles proposés	la première partie été beaucoup plus intuitive.	la partie Analyse de communication été compliqué. Peut être se serait bien de le diviser en plusieurs concepts.
Sur l'expé	→ Bien géré → Le travail en binome -> les cacahuèt !!	→ des post-it ont été difficile a gérer. (ordinateur c'est mieux) → des exercices ont été un peu trop compliqué (le deuxième surtout) → Il manquais un dictionnaire des termes ~~→ pas mêmes~~ → (mêmes exercises => pas de confrontation ->

FIGURE C.4 – Questionnaire Points forts/points faibles rempli par le sujet 7

Annexe D

Modèles ACTIF

De manière à éviter que chaque application ne se développe de manière isolée, la démarche ACTIF (Aide à la Conception de Systèmes de Transports Interopérables en France) a été lancée en 1999 par le Ministère de l'Environnement, du Développement et de l'Aménagement Durables. L'objectif du projet est de fournir aux maîtres d'ouvrages et aux équipes techniques un ensemble permettant de concevoir des SIT interopérables. Le projet ACTIF tente de traiter tous les types de transports (individuels, collectifs, personnes et biens), et tous les acteurs en interaction avec le SIT. Le site Web du projet (www.its-actif.org) propose :

- une méthode de conduite de projet SIT en environnement complexe ;
- un modèle proposant un référentiel des métiers des transports et de leurs interfaces : la version 5 est en ligne depuis le 5 octobre 2007 ;
- un outil logiciel permettant d'appliquer la méthode et d'utiliser le modèle ;
- un ensemble de rapports d'études présentant des applications concrètes d'ACTIF.

Le modèle ACTIF (V5 octobre 2007) est un référentiel des métiers des transports accessible et consultable librement sur Internet. L'objectif du modèle est de « faciliter l'échange d'informations, la collaboration et la recherche de solutions mutualisées, entre des structures et organisations ayant des missions et des périmètres d'activités différents » [1]. La représentation des métiers y est donc faite du point de vue du SI. Le modèle tente de couvrir tous les types (transports de fret ou de personnes) et tous les modes de transports. Les métiers des transports ont été séparés en neuf domaines fonctionnels principaux. Ces domaines fonctionnels sont reconnus comme un découpage valide au niveau international. Chaque domaine fonctionnel présente les chaînes fonctionnelles et détaille les fonctions de recueil, traitement, stockage et diffusion de l'information. Les interfaces, en termes de flux d'information sont décrites. Les acteurs externes sources ou destinations de l'information manipulée sont indiqués le plus exhaustivement possible. Les normes techniques et/ou règlementaires applicables aux différents niveaux modélisés sont indiquées.

Les neuf domaines fonctionnels définis par ACTIF sont les suivants :

- « Fournir des moyens de paiement électronique » ;
- « Gérer les services d'urgence et de sécurité » ;
- « Gérer les infrastructures de transports et leurs trafics » ;
- « Exploiter les transports publics » ;
- « Fournir des systèmes d'assistance à la conduite » ;
- « Gérer et Informer sur les déplacements multimodaux » ;
- « Faire appliquer la réglementation » ;
- « Exploiter les marchandises et les flottes » ;
- « Gérer les données partagées ».

Le domaine fonctionnel « Gérer et Informer sur les déplacements multimodaux » nous intéresse plus particulièrement dans le cadre de cette étude car il comprend un ensemble de fonctions proches de la problématique traitée dans le projet DéSIT [10]. Il est structuré autour des activités de mise à disposition de l'information sur l'offre de transport, de préparation du déplacement (consultation de l'offre, planification, personnalisation), d'information sur les offres connexes, de l'historisation. Le site Internet ACTIF propose de « naviguer dans les modèles » et d'accéder aux diagrammes logiques les composant. La représentation logique du modèle ACTIF est issue d'une décomposition fonctionnelle de type hiérarchique présentée par la figure D.1. Le plus haut niveau (en dessous du niveau représentant le STI) est constitué des domaines fonctionnels qui approximativement cohérents à l'échelle européenne. Ces domaines constituent un découpage reconnu des métiers. Les domaines fonctionnels sont ensuite découpés, selon les différentes logiques métier, en fonctions agrégées (sous-domaines fonctionnels), puis en fonctions élémentaires au niveau le plus fin de la décomposition. Les flux logiques représentent les échanges de données entre des éléments du modèle. Les Stocks de données représentent des éléments destinés à la conservation et à la mise à disposition de l'information traitée dans un système STI. L'architecture logique est représentée selon trois types de diagrammes. Les arbres fonctionnels D.2 rendent compte du lien de dépendance " hiérarchique " des fonctions.

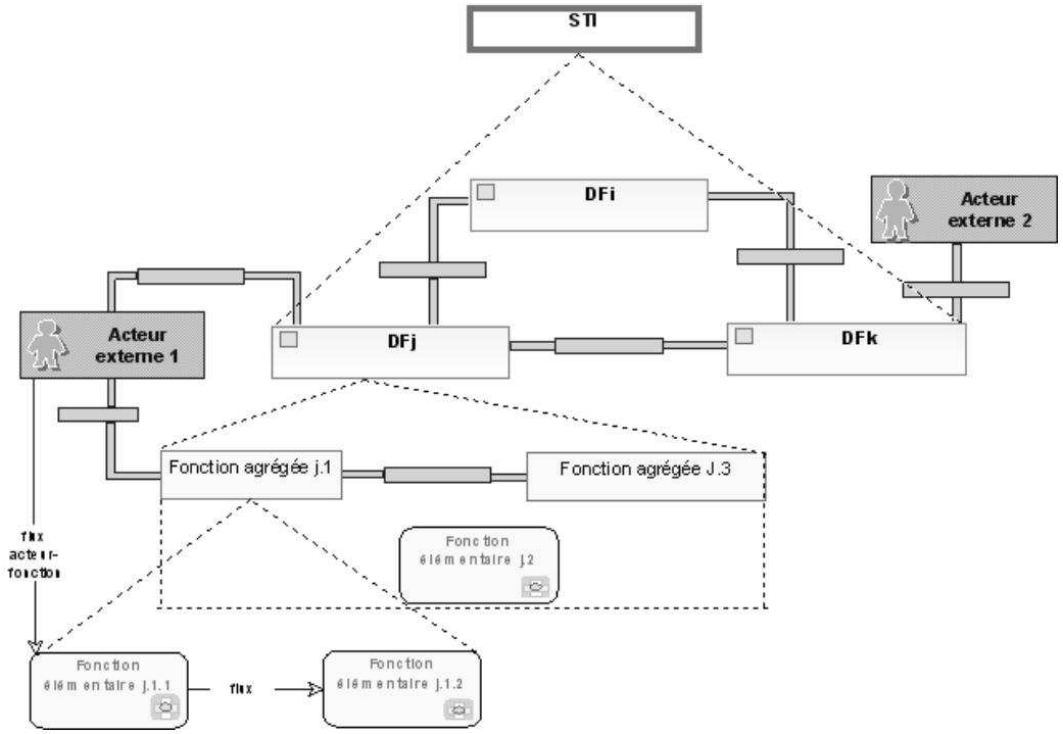


FIGURE D.1 – ACTIF : décomposition en domaines fonctionnels

Les diagrammes de contexte (voir figure D.3) offrent des vues agrégées des échanges existants entre les différents domaines et sous-domaines fonctionnels et leur environnement.

Les diagrammes de flux de données (DFD) (voir figure D.4) montrent le fonctionnement d'une chaîne fonctionnelle logique d'un sous-domaine fonctionnel au travers des flux de données échangés.

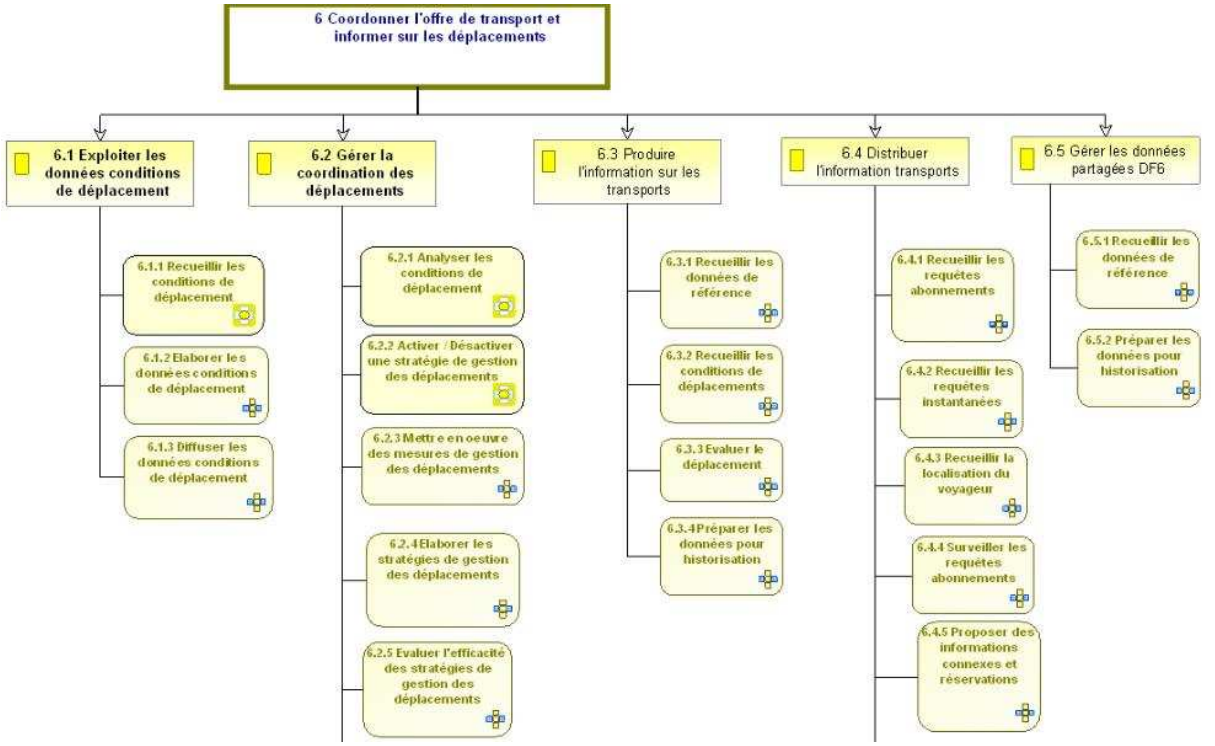


FIGURE D.2 – Extrait d'un arbre fonctionnel d'ACTIF

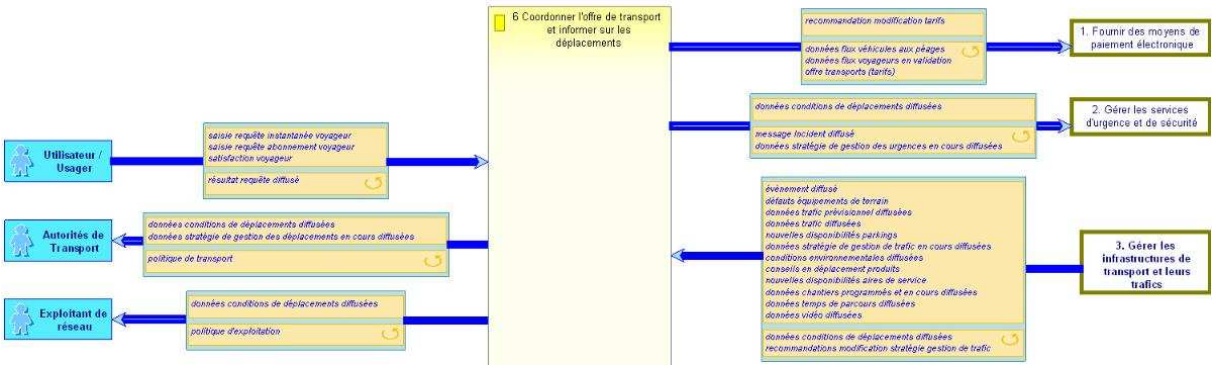


FIGURE D.3 – Extrait d'un diagramme de contexte d'ACTIF

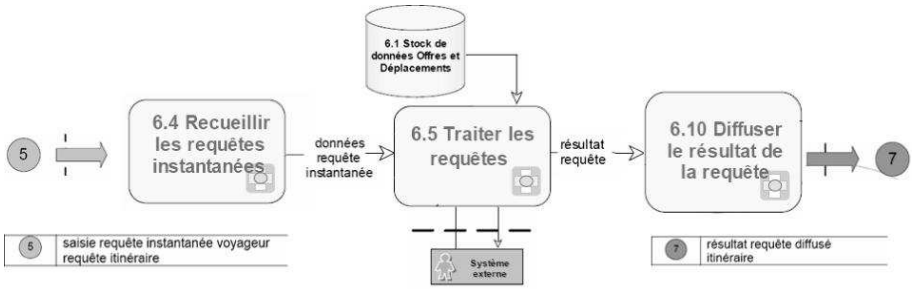


FIGURE D.4 – Extrait d'un diagramme de flux de données d'ACTIF

Annexe E

Transformations ATL

E.1 Transformation des modèles de règles et d'acquisition en une cartographie d'évènements

L'atelier ATL est utilisé dans l'étape de "Spécification de la cartographie d'évènements" 4.6.3 pour fournir au concepteur de contexte un framework acceptant en entrée un modèle de règles et un modèle d'acquisition et fournissant en sortie une cartographie d'évènements. Il s'agit d'une transformation ATL utilisant les métamodèles d'entrée et de sortie sous forme de fichiers "ecore" comme le montre la figure E.1.

Les règles de transformation décrites dans la section 4.6.3 sont exprimées avec le langage ATL. Un fragment du code de la transformation est représenté dans la figure E.2. Cette transformation utilise des modèles d'entrée et de sortie sous forme de fichiers XML utilisant le langage XMI (XML Metadata Interchange) et représentant les règles E-CARe et les sources des données contextuelles.

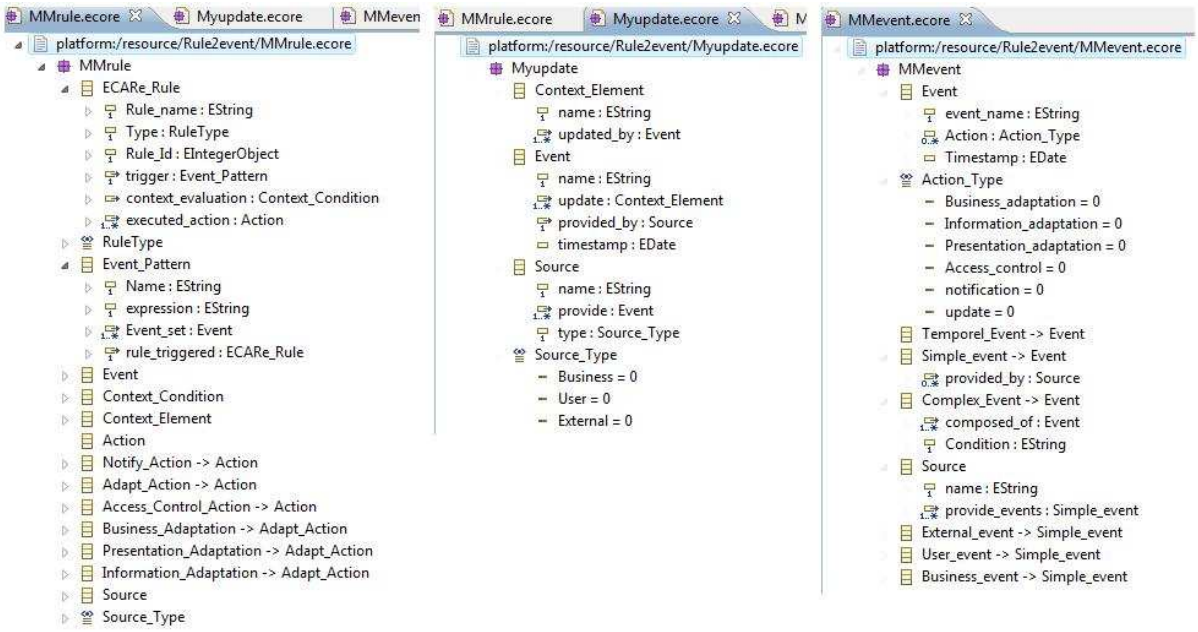


FIGURE E.1 – Métamodèles de règles, d'acquisition et d'évènements exprimés avec le langage ecore

L'application de ces transformations sur le modèle de règles exprimé en XMI de la figure E.3 et

MMRule2MMevent.atl

```
-- @path MMrule=/Rule2event/MMrule.ecore
-- @path MMevent=/Rule2event/MMevent.ecore
-- @path Myupdate=/Rule2event/Myupdate.ecore

module MMRule2MMevent;
create OUT : MMevent from IN : MMrule, IN1 : Myupdate;

helper def : enumMap : Map(MMrule!RuleType,MMevent!Action_Type) =
    Map{(#Business_Adaptation,#Business_adaptation),(#Presentation_Adaptation,#Presentation_

helper context MMrule!Event_Pattern def: action_type: MMrule!RuleType =
            self.rule_triggered.Type;

rule eventpattern2complexevent {
    from
        g: MMrule!Event_Pattern

    to
        t: MMevent!Complex_Event (
            Action <- thisModule.enumMap.get(g.action_type),
            Condition <- g.expression,
            event_name <- g.Name
            )
}

rule updateevent2simple {
    from
        g: Myupdate!Event (g.timestamp.oclIsUndefined())
    to
        t: MMevent!Simple_event  (
            Action <- #update,
            event_name <- g.name
            )
}
```

FIGURE E.2 – Fragment du code ATL pour générer la cartographie d'évènements

le modèle d'acquisition en XMI de la figure E.4 produit une cartographie d'évènements en XMI de la figure E.5.

example_sortie.xmi | example_sortie.xmi | example_event.xmi | example_event.xmi | example_update.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="MMrule">
    <ECARe_Rule Rule_name="notification personnelle" Type="Notification" Rule_Id="1" xmi:id ="rule1" >
        <trigger href="#pattern1"/>

    </ECARe_Rule>
    <ECARe_Rule Rule_name="adapter guidage" Type="Business_Adaptation" Rule_Id="12" xmi:id ="rule2" >

        <trigger href="#pattern2"/>
    </ECARe_Rule>
    <Event_Pattern xmi:id ="pattern1" Name="alerte" expression="retard.time>3 and traffic.level>2">
        <Event_set href="#event1"/>
        <Event_set href="#event2"/>
        <rule_triggered href="#rule1"/>

    </Event_Pattern>
    <Event_Pattern xmi:id ="pattern2" Name="alerte2" expression="retard.time>3 and traffic.level>2">
        <Event_set href="#event1"/>
        <Event_set href="#event2"/>
        <rule_triggered href="#rule2"/>
    </Event_Pattern>
    <Event xmi:id ="event1" name="retard"/>
    <Event xmi:id ="event2" name="traffic"/>
</xmi:XMI>
```

FIGURE E.3 – Modèle de règles en XMI utilisé en entrée

MMevent.ecore | MMRule2MMevent.atl | example_update.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="Myupdate">
    <Context_Element name="Localisation"  xmi:id ="CE1" >
        <updated_by href="#event1"/>

    </Context_Element>
    <Context_Element name="Preferences"  xmi:id ="CE2" >
        <updated_by href="#event2"/>

    </Context_Element>
    <Context_Element name="Jours_repos"  xmi:id ="CE3" >
        <updated_by href="#event3"/>

    </Context_Element>
    <Event xmi:id ="event1" name="loca_GPS" >
        <update href="#CE1"/>
        <provided_by href="#source1"/>
    </Event>
    <Event xmi:id ="event2" name="preference exprimée" >
        <update href="#CE2"/>
        <provided_by href="#source2"/>
    </Event>
    <Event  xmi:id ="event3" name="jours férier" timestamp = "2011-12-25">
    </Event>
    <Source xmi:id ="source1" name="GPS" type= "External"/>
    <Source xmi:id ="source2" name="passenger" type= "User"/>
</xmi:XMI>
```

FIGURE E.4 – Modèle d'acquisition en XMI utilisé en entrée

MMevent.ecore | MMRule2MMevent.atl | example_sortie.xmi | example_sortie.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="MMevent">
  <Complex_Event event_name="alerte" Condition="retard.time>3 and traffic.level>2">
    <Action>notification</Action>
  </Complex_Event>
  <Complex_Event event_name="alerte2" Condition="retard.time>3 and traffic.level>2">
    <Action>Business_adaptation</Action>
  </Complex_Event>
  <Simple_event event_name="loca_GPS">
    <Action>update</Action>
  </Simple_event>
  <Simple_event event_name="preference exprimée">
    <Action>update</Action>
  </Simple_event>
  <Temporel_Event event_name="jours férier" Timestamp="2011-12-25T00:00:00.000+0100">
    <Action>update</Action>
  </Temporel_Event>
</xmi:XMI>
```

FIGURE E.5 – Cartographie d'évènements obtenue en sortie

E.2 Vérification de cohérence entre modèles d'évènements et modèles métier

L'objectif de l'étape "Vérification de la cohérence entre évènements et PM" à la section 5.4.1 est de vérifier la cohérence entre la cartographie d'évènements obtenue dans la branche ubiquitaire et les modèles de PM obtenus dans la branche fonctionnelle. Les métamodèles d'entrée sont utilisés sous forme de fichiers "ecore" et correspondent au métamodèle d'évènements et au métamodèle de BPMN (figure E.6).

Le métamodèle de sortie est le métamodèle de problème sous la forme d'un fichier "ecore" (figure E.7).

La figure E.2 présente un extrait du code ATL utilisé pour la vérification de cohérence.

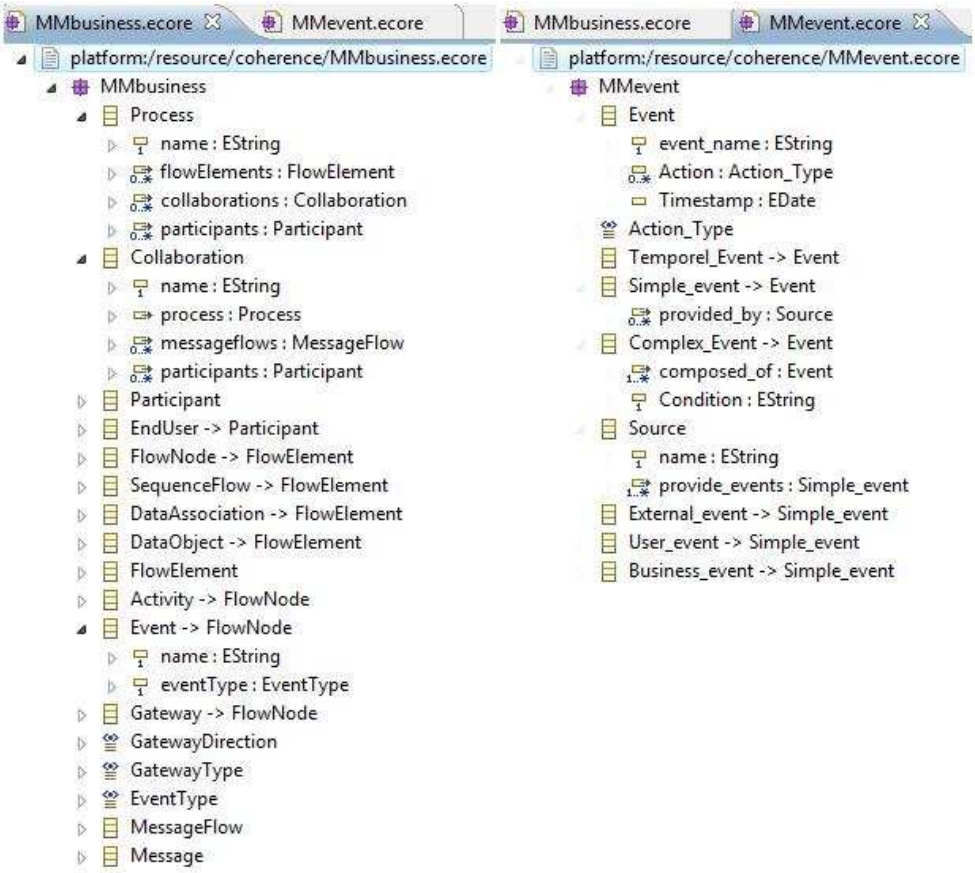


FIGURE E.6 – Métamodèles d'évènements et de BPMN exprimés en "ecore"

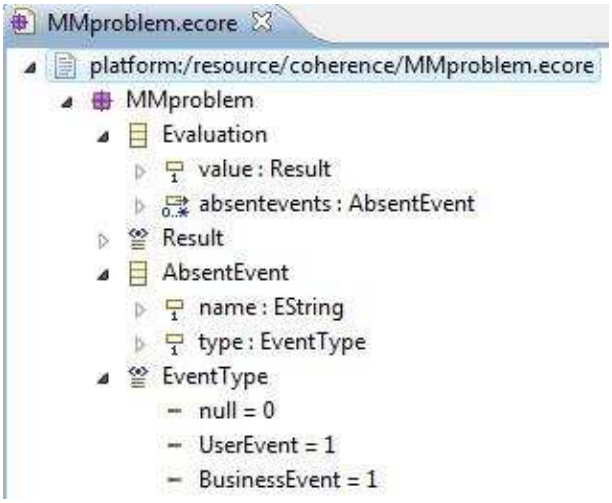


FIGURE E.7 – Métamodèle de problème exprimé avec le langage ecore

eventcoherence.atl

```
module eventcoherence;
create OUT : MMproblem from IN : MMbusiness, IN1 : MMevent;

helper  def: EvaluationResult: MMproblem!Result =
            #succeed;

helper  def: TBusinessEventSet(): Set(String) =
            MMbusiness!Event.allInstances() -> collect(e|e.name).asSet();

helper  def: TUserEventSet(): Set(String) =
            MMbusiness!MessageFlow.allInstances() -> collect(e|e.name).asSet();

rule comparingbusiness_events {
    from
        s : MMevent!Business_event (not (thisModule.TBusinessEventSet() -> includes(s.event_name)))
    to
        tt : MMproblem!AbsentEvent ( )
        do {
                tt.type <- #BusinessEvent;
                thisModule.EvaluationResult <- #fail;
                tt.name <- s.event_name;
                }
    }

rule comparinguser_events {
    from
        ss : MMevent!User_event (not (thisModule.TUserEventSet() -> includes(ss.event_name)))
    to
        tt : MMproblem!AbsentEvent ()
        do {
                thisModule.EvaluationResult <- #fail;
                tt.name <- ss.event_name;
                tt.type <- #UserEvent;
                }
```

FIGURE E.8 – Extrait du code ATL pour la vérification de cohérence entre modèles d'évènements et modèles de PM

E-CARe : une méthode d'ingénierie des Systèmes d'Information ubiquitaires

Abstract : Ubiquitous Information Systems appeared as a consequence to emerging and evolving communication and mobile technologies providing the system with information on its environment, the environment of its users and their profiles. These data constitute the application context and are used to provide personalized, targeted and relevant services. However, ubiquitous services face some difficulties and challenges concerning specially needed contextual data, adaptation degree and computerized decision making. This is due to the gap between advanced ubiquitous services and their applications, and methods and processes for developing and engineering ubiquitous systems.

Our goal in this thesis is to propose an engineering method for ubiquitous Information Systems considering different requirements resulting from the mobile and high scalable nature of these systems. The proposed method is based on a development process and a set of generic metamodels and languages facilitating a complete system specification and implementation. The proposed process separates functional, technical and ubiquitous specifications. Ubiquitous specifications enable the structural and event based context models definition while considering user requirements and security requirements. Adaptation and context awareness functionalities are supported by structural and dynamic context models. The proposed event oriented approach is enhanced by the adoption of an event processing architecture. Ubiquitous specifications are integrated into a classical information systems engineering process to constitute the E-CARe process including functional and technical specifications. Our propositions are used to design a user assistance application in the transport domain, highly dependent on the ambient environment and events.

Keywords : Ubiquitous IS, event flow, context awareness, engineering method.

Résumé : L'apparition des Systèmes d'Information ubiquitaires ou pervasifs est issue de l'émergence de nouvelles technologies fournissant au système une vision de son environnement, de l'environnement de ses utilisateurs ainsi que de leurs profils. Grâce à ces données formant le contexte de l'application, il est possible de fournir des services personnalisés, pertinents et ciblés. Mais, le problème qui se pose à ce niveau concerne le degré d'adaptation, de prise de décision à la place de l'utilisateur et de l'identification des données contextuelles nécessaires et suffisantes pour ces services. Ceci est dû à un déséquilibre entre les avancées des technologies et de leurs applications (qui reçoivent un grand intérêt de la part de la recherche et de l'industrie) et les méthodes et démarches de développement et d'ingénierie spécifiques aux systèmes ubiquitaires.

Notre objectif dans ce travail de thèse est de proposer une méthode d'ingénierie des SI ubiquitaires en considérant les différentes exigences reliées à la nature mobile et grande échelle de ces systèmes. Cette méthode est basée sur une démarche de développement qui fait usage d'un ensemble de métamodèles et de langages génériques favorisant la spécification complète de ces systèmes. Cette démarche sépare les spécifications fonctionnelles, techniques et ubiquitaires. Les spécifications ubiquitaires forment la démarche E-CARe qui permet de définir des modèles structurels et événementiels du contexte respectant les exigences des utilisateurs et les contraintes de sécurité et supportant les fonctionnalités d'adaptation et de sensibilité au contexte. Cette approche orientée évènements est consolidée par l'adoption d'une architecture de traitement des évènements. Notre démarche E-CARe est intégrée dans une démarche classique de conception des SI pour garantir la couverture des spécifications fonctionnelles et techniques. Les applications d'assistance représentent un cas d'étude idéale pour cette démarche qui s'intéresse au domaine des transports, fortement dépendant de l'environnement et des évènements ambiants.

Mots Clés : SI ubiquitaire, flux d'évènements, adaptation au contexte, méthode d'ingénierie.