



UNIVERSITÉ DE NAMUR

INFOM434 - INGÉNIERIE DES ARCHITECTURES
LOGICIELLES : MATIÈRES APPROFONDIES

Rapport du Projet

Groupe 1 :

Sébastien ANTOINE
Simon DESART
Thomas DESART

Professeur :

Vincent ENGLEBERT

Année académique 2014-2015

Table des matières

Introduction	2
1 Définition du domaine	2
1.1 Besoins	2
1.2 Contraintes	2
1.3 Rôles (Personas)	3
1.4 Problèmes	5
1.5 Ebauche de solution	5
1.6 Composants	6
1.6.1 Aide au déplacement	6
1.6.2 Monitoring passif	6
1.6.3 Alarme	6
1.6.4 Aspect social	6
2 Méta-Modèle	7
2.1 Définition du modèle	8
2.2 Diagramme des features	9
2.3 Syntaxe concrète	10
2.3.1 Définition de chaque concept	10
2.3.2 Exemple d'un modèle	13
3 Conception de la software factory	14
3.1 Patterns de variabilité	14
3.1.1 Player-Role	14
3.1.2 Strategy	14
3.1.3 Template	14
3.1.4 State	14
3.1.5 Observer	15
3.2 Contrôle du matériel	15
3.3 Communication	15
3.4 Mobilité	16
3.5 Choix de la configuration	17
3.6 Monitoring	18
3.7 Alarme	18
3.8 Connexion	19

Introduction

Ce document a pour but de modéliser l'architecture d'un système d'aide et de monitoring destiné aux personnes âgées. Le but est de créer une architecture qui sera adaptable aux problèmes spécifiques à chaque individu. Nous souhaitons porter une attention particulière sur le fait que ce système ne sera pas intrusif et sera là avant tout comme support à la vie quotidienne de la personne.

1 Définition du domaine

1.1 Besoins

Les besoins des personnes âgées peuvent être assez divers. Nous avons décidé de nous focaliser sur les suivants :

1. Monitoring passif : c'est-à-dire le suivi de la personne grâce à un ensemble de capteurs en tout genre (température, mouvement) afin d'en déduire des patterns spécifiques à la personne. On peut par exemple déduire que la personne va dans sa cuisine faire son café tous les jours entre 7h30 et 8h et si le pattern n'est pas réalisé, une alerte est lancée. Cela concerne donc spécifiquement la personne en elle-même.
2. Domotique simplifiée : la personne aura accès à une interface lui permettant d'avoir une vue d'ensemble sur différents aspects de son habitat (lampes, portes, fenêtres, appareils connectés). Cette interface sera également mise à disposition des proches si cela est souhaité. Cela concerne donc spécifiquement le contrôle de la personne âgée sur son environnement.
3. Aspect social : le système permettra aux utilisateurs isolés de prendre contact facilement avec leur proche. On pourrait également imaginer un réseau de personnes seules permettant la connexion automatique entre deux personnes désirant discuter d'un sujet en particulier.

1.2 Contraintes

Nous avons décidé de nous focaliser sur les contraintes suivantes :

1. Interaction avec l'utilisateur : il serait intéressant de minimiser l'interaction afin d'éviter le rejet par les utilisateurs réticents aux nouvelles technologies. C'est à l'outil de s'adapter à la personne et non l'inverse et donc celle-ci ne doit pas posséder de connaissances spécifiques relatives à l'informatique ou aux nouvelles technologies. Une formation rudimentaire pourrait par contre être envisagée.
2. Connectivité : pour assurer certaines des fonctionnalités, la connectivité est primordiale. Celle-ci pourrait se faire via internet ou un opérateur spécifique.
3. Coût : le coût de l'installation doit rester abordable pour les utilisateurs.
4. Vie privée : le système ne devra pas être intrusif et les données échangées devront être adéquates et détruites en temps voulu. L'intégrité des données doit donc être respectée.
5. Facilité du déploiement : le système devra être facilement déployable et adaptable aux utilisateurs.

1.3 Rôles (Personas)

Afin de pouvoir comprendre les différents problèmes que pourrait rencontrer une personne âgée, nous décidons d'utiliser la méthode Persona (personnes fictives) afin de définir des rôles utilisateurs. La liste suivante se veut non-exhaustive et tente de couvrir un maximum de situations "à problèmes", aussi bien sur le plan social que sur les aspects santé et économique. Le budget est pris en compte afin de pouvoir éventuellement proposer, en bout de course, un service adapté au porte-feuille du client (concrètement, la possibilité de choisir entre plusieurs packages correspondant à différentes solutions plus ou moins complètes). Il est à noter que nous n'avons pas pris en compte des personnes souffrant de maladie dégénérative ou neurodégénérative car la gestion de ces cas nécessite une attention beaucoup plus spécifique, aussi avons-nous décidé ne pas inclure ceux-ci dans le périmètre de ce projet.

1. *Nom* : D. Minet
Age : 80 ans
Situation sociale : Vit seule depuis la mort de son mari il y a une vingtaine d'années, très peu de contacts humains
Santé : Aucun problème de santé majeur
Budget : Limité (petite pension)
Rapport avec la technologie et/ou l'informatique : N'a jamais utilisé un ordinateur. Regarde la télévision et utilise un vieux téléphone fixe à cadran. N'a pas de GSM.
Craintes : Ne plus avoir personne à qui parler. A parfois peur pour sa sécurité.
2. *Nom* : M. Laurent
Age : 65 ans
Situation sociale : Vit seul depuis peu, réseau de connaissances peu développé (très peu de visites)
Santé : Arthrose provoquant des difficultés à se mouvoir
Budget : Dans la moyenne (pension normale)
Rapport avec la technologie et/ou l'informatique : Possède un vieux PC tournant sur Windows 95, l'utilise principalement pour échanger des mails.
Craintes : Ne plus savoir se déplacer dans sa maison comme il le souhaiterait
3. *Nom* : P. Lepers
Age : 70 ans
Situation sociale : Personne récemment veuve disposant d'un réseau de connaissances assez développé (famille, amis et voisins qui lui rendent visite fréquemment)
Santé : Problèmes cardiaques
Budget : Dans la moyenne (pension normale)
Rapport avec la technologie et/ou l'informatique : A peur de tout ce qui touche aux nouvelles technologies. Craint pour le respect de sa vie privée, et communique toujours en écrivant des lettres.
Craintes : Peur que l'on ne lui vienne pas en aide assez vite en cas de problème (infarctus ou intrusion de son domicile)

4. *Nom* : M-A. Fauville

Age : 75 ans

Situation sociale : Marié (âge du conjoint : 75 ans)

Santé : Troubles de la mémoire

Budget : Supérieur à la moyenne (pension confortable)

Rapport avec la technologie et/ou l'informatique : N'a pas d'ordinateur. Aime regarder la télévision pour se détendre et passer des coups de fil (possède un téléphone fixe récent).

Craintes : Oublier des choses importantes, comme fermer sa porte.

5. *Nom* : J. Bertrand

Age : 60 ans

Situation sociale : Marié (âge de la conjointe : 65 ans)

Santé : Diabétique

Budget : Supérieur à la moyenne (pension confortable)

Rapport avec la technologie et/ou l'informatique : Utilise un ordinateur récent pour surfer sur internet. A l'aise avec les nouvelles technologies.

Craintes : Oublier son injection d'insuline. Peur qu'on ne l'aide pas à temps en cas de gros problème.

1.4 Problèmes

A partir des différents rôles définis plus haut, nous pouvons identifier une série de problèmes récurrents, que nous classerons de la façon suivante :

1. *Mobilité* : Comprend tous les problèmes liés à la difficulté pour un utilisateur à se mouvoir dans sa maison (exemple : monter les escaliers)
2. *Sécurité* : Correspond aux différents dangers que peut craindre une personne âgée (accident domestique, cambriolage,...). A noter que nous considérons un oubli grave (exemple : partir sans fermer sa porte, ne pas éteindre un appareil électrique) comme étant également un problème de sécurité.
3. *Santé* : Les différents ennuis de santé pouvant être subis se trouvent dans cette catégorie.
4. *Social* : Cette catégorie comprend principalement les manquements en matière de contacts humains pour les personnes âgées (solitude, isolement,...).

1.5 Ebauche de solution

Au vu de la décomposition proposée ci-dessus, nous pouvons déjà fournir une première solution générale. Celle-ci sera bien sûr raffinée ultérieurement, notamment afin de prendre en compte différentes contraintes spécifiques à certains utilisateurs (budget,...). Cette solution comprendrait les fonctionnalités suivantes :

1. *Aide au déplacement* : Se traduira probablement par un monte-escaliers (principal problème de mobilité au sein d'une maison). Pourrait également comporter un système de guidage dans la maison.
2. *Monitoring* : Consisterait en une solution domotique qui permettrait, selon les principes de l'Internet des objets, de connecter et contrôler l'ensemble de la maison (environnement de l'utilisateur). Ceci permettrait notamment l'analyse de patterns dans les activités quotidiennes afin de détecter des problèmes ou des oublis (exemple : l'utilisateur n'a pas utilisé d'eau au matin comme tous les jours, cela veut dire qu'il a oublié son médicament). Ce monitoring serait passif dans ce cas-ci, c'est-à-dire qu'il ne nécessiterait pas d'interaction directe avec l'utilisateur pour fonctionner, et serait le moins intrusif possible. Une solution de monitoring actif en utilisant la domotique est également envisagée et permettrait à l'utilisateur de par exemple fermer des appareils ou lampes à distance.
3. *Alarme* : Permettrait de résoudre le problème de la sécurité. A noter que l'alarme envisagée ici serait plus générale qu'un simple système de détection d'intrusion : elle serait également capable d'alerter automatiquement une centrale, qui préviendrait à son tour les secours. Elle serait ainsi également utilisable en cas de problème de santé (infarctus ou autre). A lier au système de monitoring donc.
4. *Communication* : Un système de communication devrait être mis en place afin de permettre à l'utilisateur de conserver des contacts humains. Ce système pourrait faire emploi de logiciels déjà existants (Skype par exemple), mais devrait être le plus simple d'utilisation possible.

1.6 Composants

Pour la solution énoncée plus haut, nous aurions besoin des composants suivants :

1.6.1 Aide au déplacement

1. Monte-escaliers

1.6.2 Monitoring passif

1. Gamme de capteurs (à préciser)
2. Connexion sécurisée entre les différents capteurs/composants
3. Composant de contrôle des lampes ou équipement ménager ou multimédia
4. Composant d’affichage (panneau tactile à envisager mais alors prendre en compte la facilité d’utilisation et l’acceptation)

1.6.3 Alarme

1. Bip d’ouverture de la porte
2. Détecteur de fumée
3. Sirène
4. Système de notification à la centrale

1.6.4 Aspect social

1. Réutilisation du composant d’affichage
2. Composant de connexion

2 Méta-Modèle

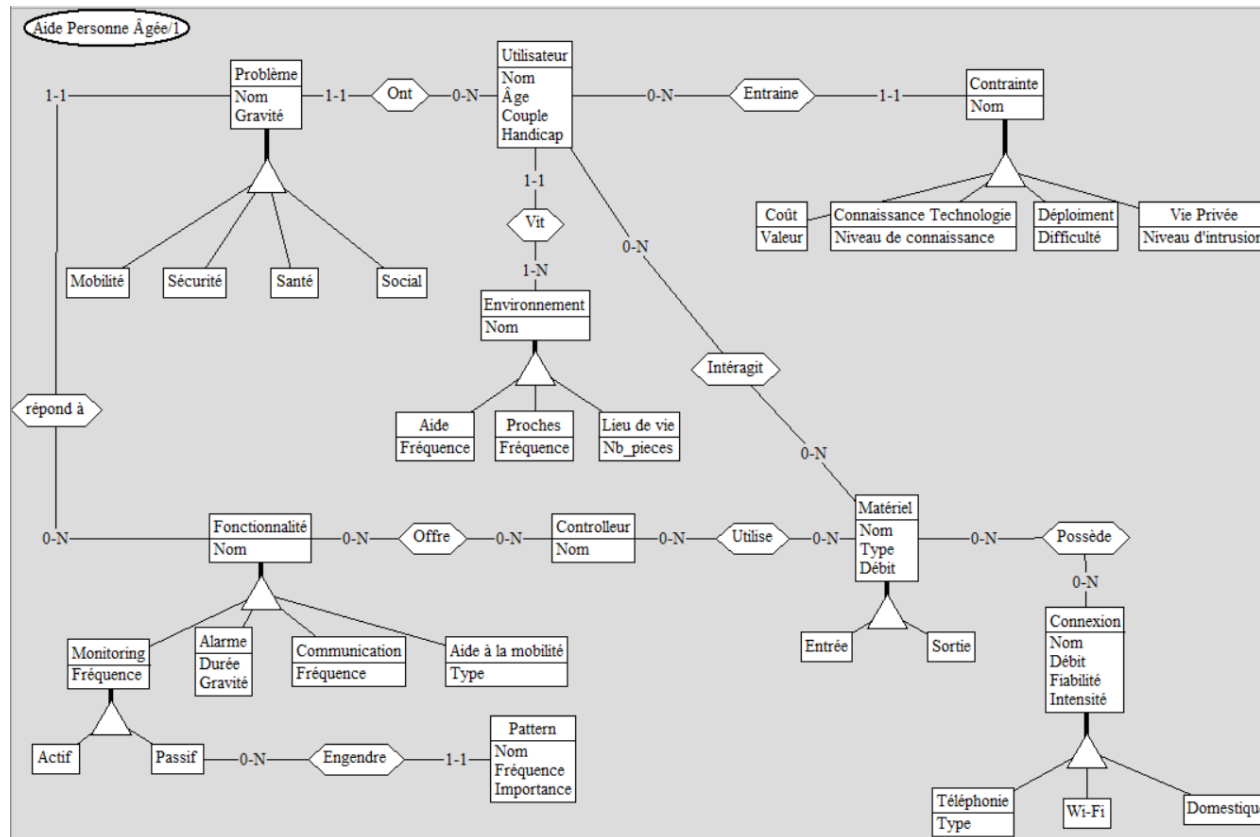


FIGURE 1 – Méta-modèle

2.1 Définition du modèle

Utilisateur : La personne qui utilisera l'application. Elle est caractérisée par son nom, son âge, par le fait qu'elle soit en couple et si elle possède un handicap ou non. Les utilisateurs ont été décrit plus précisément dans la section sur les personas (cfr section 1.3). L'utilisateur est également caractérisé par les problèmes auxquels il fait face, l'environnement dans lequel il vit et les contraintes qu'il engendre. Ces trois éléments lui sont propres et sont uniques d'où les cardinalités utilisées.

Problème : Les problèmes sont ceux explicités dans la section 1.4.

Un problème est caractérisé par son nom et sa gravité. La gravité étant une métrique allant de faible à importante. En plus d'être liés à l'utilisateur, ils vont également influencer les fonctionnalités d'où la présence d'une relation entre les deux concepts.

Environnement : L'environnement se présente sous trois formes. Premièrement, l'environnement physique caractérisé par un nombre de pièces représente principalement l'habitat de l'utilisateur. Deuxièmement, les proches de l'utilisateur peuvent également avoir une influence sur l'application à déployer, la fréquence de visite étant la caractéristique majeure de cet environnement. Et troisièmement, les services d'aide comme les infirmières à domicile doivent également être pris en compte, la fréquence d'aide étant ici la caractéristique de cet environnement. Chaque environnement possède également un nom.

Contrainte : Les contraintes sont celles explicitées dans la section 1.2. Chaque contrainte possède des attributs qui lui sont propres.

Fonctionnalités : Les fonctionnalités sont les différentes propositions énoncées dans les sections 1.1 et 1.5. Chaque fonctionnalité possède des attributs qui lui sont propres.

Pattern : Les patterns sont les habitudes ou les comportements récurrents enregistrés grâce au monitoring passif. Ils sont caractérisés par un nom, une fréquence et une importance.

Contrôleur : Le contrôleur est la pièce centrale de l'application qui va utiliser le matériel afin d'offrir les différentes fonctionnalités. Cela est traduit sur les schémas par les relations entre ceux-ci. Il est caractérisé par son nom.

Matériel : Le matériel est soit vu comme une entrée (par exemple des capteurs) ou alors comme une sortie (alarme, affichage). Il est également caractérisé par un nom, un type et un débit.

Connexion : Utilisé par le matériel, elle peut-être de trois types comme présentés ; c'est-à-dire de téléphonie mobile (et donc d'un des 3 types suivant : Edge, 3G, LTE), Wi-Fi ou encore domestique (utilisation du réseau électrique de l'habitat par exemple). Une connexion est caractérisée par son nom, son débit, sa fiabilité et son intensité.

2.2 Diagramme des features

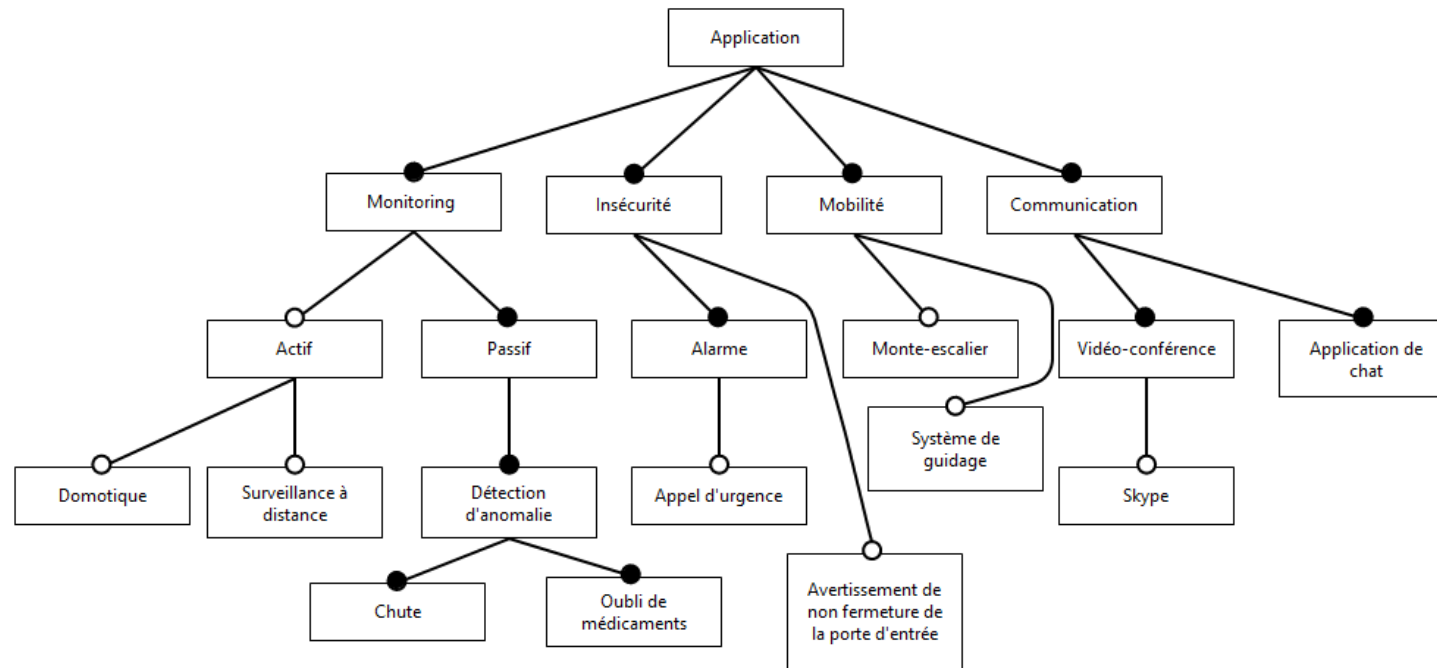


FIGURE 2 – Diagramme des features

2.3 Syntaxe concrète

2.3.1 Définition de chaque concept

Utilisateur :



Nom
Age
Couple
Handicapé

Problème - Santé :



Nom
Gravité

Problème - Mobilité :




Nom
Gravité

Problème - Sécurité :



Nom
Gravité

Problème - Social :



Nom
Gravité

Environnement - Aide :




Nom
Fréquence

Environnement - Proches :




Nom
Fréquence

Environnement - Lieu de vie :



Nom
Nb pièces

Contrainte - Coût :




Nom
Valeur

Contrainte - Connaissance technologique :



Nom
Niveau de connaissance

Contrainte - Déploiement :  Nom
Difficulté


Contrainte - Vie Privée :  Nom
Niveau d'Intrusion

Fonctionnalité - Monitoring Actif :  Nom
Fréquence

Fonctionnalité - Monitoring Passif :  Nom
Fréquence

Fonctionnalité - Alarme :  Nom
Durée
Gravité

Fonctionnalité - Communication :  Nom
Fréquence

Fonctionnalité - Aide à la mobilité :  Nom
Type

Pattern :  Nom
Fréquence
Importance

Contrôleur :  Nom

Matériel - Input :  Nom
Type
Débit

Matériel - Output :  Nom
Type
Débit

Connexion - Téléphonie Mobile :



Nom
Débit
Fiabilité
Intensité
Type

Connexion - Wi-Fi :



Nom
Débit
Fiabilité
Intensité

Connexion - Domestique :



Nom
Débit
Fiabilité
Intensité

Relations : Les relations sont représentées par un lien entre les éléments et possèdent un nom.

2.3.2 Exemple d'un modèle

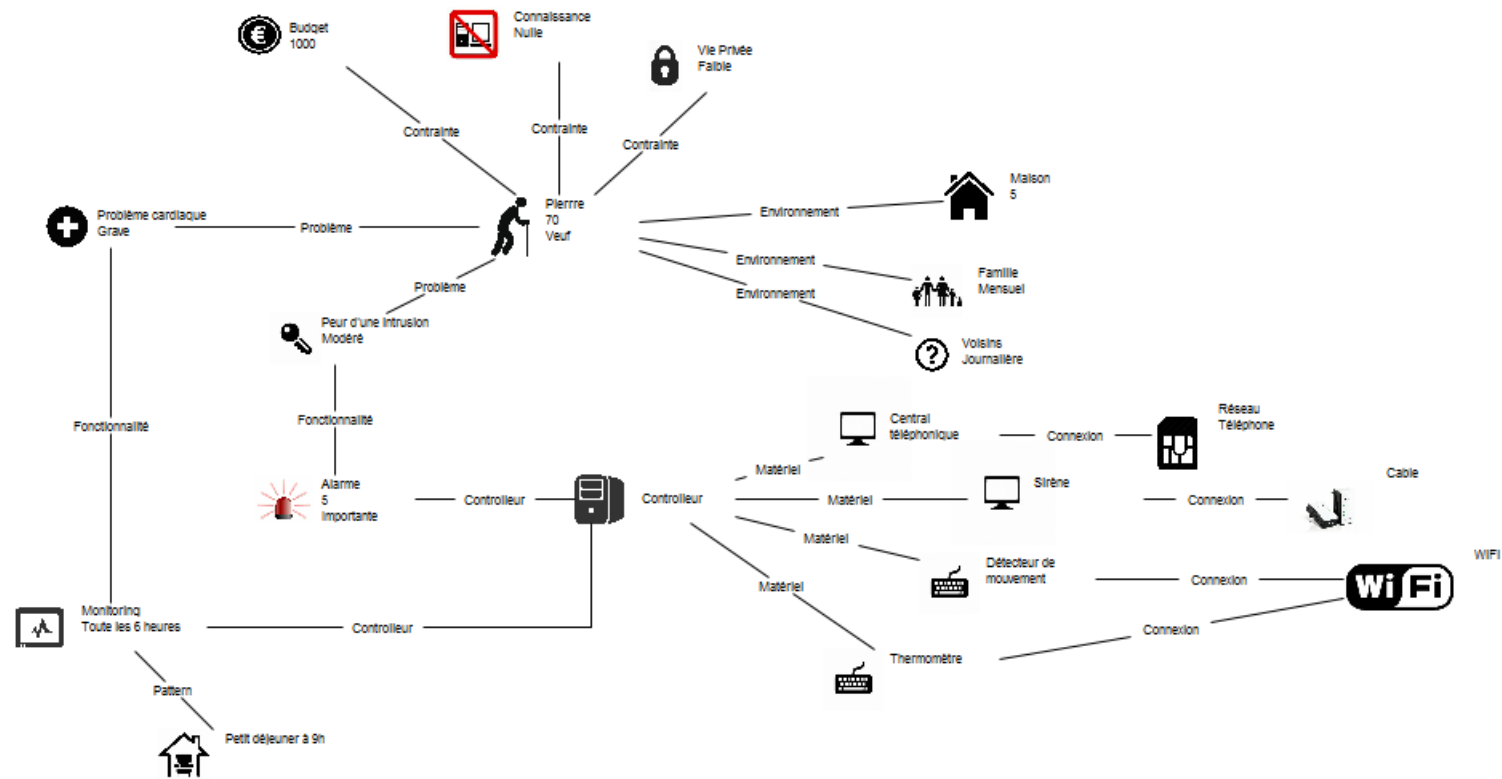


FIGURE 3 – Exemple d'un modèle réalisé avec MetaEdit

3 Conception de la software factory

Cette section présente la manière dont le middleware applicatif serait généré à partir des facteurs suivants :

- Les problèmes rencontrés par l'utilisateur
- L'environnement dans lequel vit l'utilisateur
- Des contraintes tel que le coût, le respect de la vie privée, le niveau de connaissance technologique ou encore la facilité du déploiement
- Le matériel déjà présent ainsi que celui à intégrer

Ces facteurs étant explicités dans la section relative au Méta-Modèle.

3.1 Patterns de variabilité

Cette section a pour but de présenter les différents design patterns utilisés dans la solution imaginée. Ces patterns doivent permettre le développement d'une solution qui soit facilement configurable afin de répondre à des besoins spécifiques.

3.1.1 Player-Role

Le Player-Role est un pattern structurel qui permet à un objet de participer à un ou plusieurs rôles différents dynamiquement. Ce pattern est particulièrement adéquat ici afin d'adapter le comportement du système en fonction de l'utilisateur et de ses contraintes/-problèmes. Par ailleurs, il permet également d'encapsuler les fonctionnalités liées à un rôle en particulier.

3.1.2 Strategy

Le Strategy permet de définir plusieurs implémentations différentes indépendamment et de les encapsuler de telle sorte que les modifications apportées à une implémentation n'influencent pas les autres. Dans le cadre de ce projet, ce pattern permet de proposer plusieurs configurations différentes indépendamment, de telle sorte que l'utilisateur peut choisir la configuration qui convient le mieux à ses besoins et/ou à son budget sans que cela ne porte à conséquence.

3.1.3 Template

Le template pattern permet de fournir le squelette d'un objet, dont le comportement concret se trouvera dans les sous-classes implémentant les opérations. Ceci nous permet de fixer des fonctionnalités génériques qui devront être partagées par toutes les configurations, et de d'encapsuler les fonctionnalités plus ciblées dans des sous-classes.

3.1.4 State

Le design pattern State permet à un objet de modifier son comportement en fonction de l'état dans lequel il se trouve. Ce pattern nous permettra donc de modifier dynamiquement les réponses du système. De plus, le système ne peut-être que dans un seul état à la fois.

3.1.5 Observer

Le design pattern observer est utilisé en programmation pour envoyer un signal à des modules qui jouent le rôle d'observateur. En cas de notification, les observateurs effectuent alors l'action adéquate en fonction des informations qui parviennent depuis les modules qu'ils observent (les « observables »). Il nous permet dans le cadre de notre système d'observer les différents capteurs pour ensuite traiter l'information et réagir de manière adéquate.

3.2 Contrôle du matériel

Le contrôle du matériel est une fonctionnalité transversale à l'application. En effet, il est nécessaire d'interagir avec l'ensemble du matériel présent afin de fournir les fonctionnalités métiers présentées auparavant.

La programmation par aspect semble donc idéale pour réaliser ce module de contrôle du matériel puisque ce paradigme permet de traiter séparément les fonctionnalités transversales.

3.3 Communication

Nous nous concentrons ici sur la fonctionnalité "communication". L'intérêt pour les utilisateurs est de pouvoir choisir la ou les méthodes de communication les plus adaptées à leurs besoins grâce à l'utilisation d'un pattern player-role. Ce pattern permet également d'encapsuler la gestion des différents moyens de communication et, par extension, de pouvoir rajouter facilement des technologies par la suite. La figure ci-dessous montre comment les différentes fonctionnalités de communication pourraient être gérées :

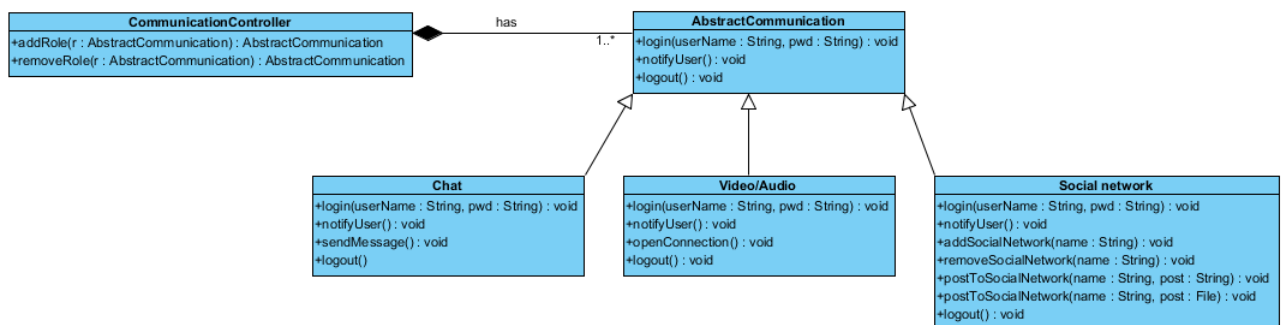


FIGURE 4 – Diagramme de classe - Communication

La classe "CommunicationController" permet de gérer dynamiquement les moyens de communications en ajoutant ou en retirant des rôles (= moyens de communication). Ici, les technologies considérées sont la messagerie instantanée ("Chat"), la téléconférence ("Vidéo/Audio") et les réseaux sociaux ("Social network").

Chaque classe concrète qui hérite de `AbstractCommunication` la spécialise en implémentant les opérations spécifiques à la communication considérée.

A noter que la classe "Social Network" consisterait en un module de post vers les réseaux sociaux, qui centraliserait les comptes de l'utilisateur sur différents sites (Facebook, Twitter, Google+, LinkedIn...) et lui permettrait ainsi d'accéder à ceux-ci via une interface unique. Cette classe contient donc une méthode afin d'ajouter un réseau social et une autre afin d'en retirer, afin de gérer dynamiquement les sites que l'utilisateur veut utiliser (un `Player-Role` pourrait également être utilisé pour cela, mais les opérations étant quasiment identiques, ce n'est pas obligatoire). L'usage du polymorphisme pour la méthode `postToSocialNetwork` permet de gérer les différents types de média envoyés (texte, image, vidéo,...).

3.4 Mobilité

La fonctionnalité "mobilité" comprend la gestion des moyens mis en oeuvre afin de faciliter les déplacements d'un utilisateur au sein de son domicile. Dans l'ébauche de solution proposée ici, elle se limite à un monte-escaliers (d'autres éléments comme des barres d'appui et un fauteuil roulant peuvent être intégrés, mais ces derniers sont principalement des éléments mécaniques et ne sont donc pas considérés ici). Cette fonctionnalité doit pouvoir être optionnelle (les personnes suffisamment aptes à se déplacer ne doivent pas être forcées d'acheter un monte-escaliers). La figure suivante montre une possibilité d'implémentation, utilisant le template pattern :

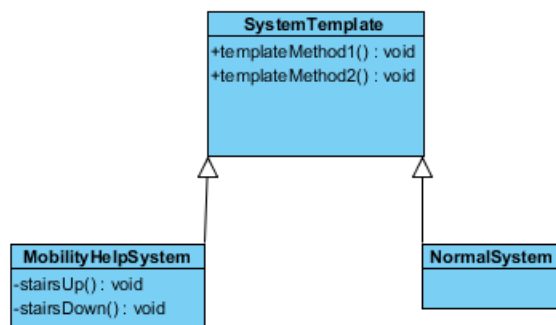


FIGURE 5 – Diagramme de classe - Mobilité

La classe "TemplateSystem" fournit des méthodes génériques héritées par tous les sous-types, correspondant aux opérations de base à devoir effectuer peu importe la configuration (ces méthodes doivent être déclarées avec le modifier "final" pour éviter l'overriding). La classe "MobilityHelpSystem" implémenterait des méthodes privées afin de gérer le système d'aide à la mobilité (dans ce cas-ci les méthodes "stairsUp()" et "stairsDown()" pour monter et descendre à l'aide du monte-escaliers respectivement). La classe "NormalSystem" quant à elle n'implémente pas ces méthodes (dans un contexte sans

monte-escaliers) et se contente d'hériter des opérations génériques.

Dans un contexte où on aurait plusieurs possibilités de systèmes d'aide à la mobilité (par exemple, on pourrait imaginer une sorte de fauteuil roulant intelligent contrôlé par le système), le pattern Template pourrait être combiné avec le pattern Strategy afin de fournir plusieurs solutions différentes, et l'utilisateur choisirait celle qui lui conviendrait le mieux.

3.5 Choix de la configuration

Comme énoncé plus haut, l'utilisateur devrait avoir le choix entre plusieurs packages de coûts différents, de telle sorte qu'il puisse choisir une configuration adaptée à son budget. La différence entre les packages se situerait principalement au niveau du matériel fourni, mais la présence ou non de certains composants est un facteur suffisamment important pour nécessiter une implémentation différente (par exemple, l'algorithme de reconnaissance de patterns comportementaux devra être adapté en fonction du nombre de capteurs et de données à sa disposition). Cette exigence semble se prêter à l'utilisation du pattern Strategy, qui permet d'encapsuler des implémentations différentes pour de mêmes opérations. Le schéma ci-dessous illustre l'utilisation de ce pattern dans le cadre de la reconnaissance de patterns :

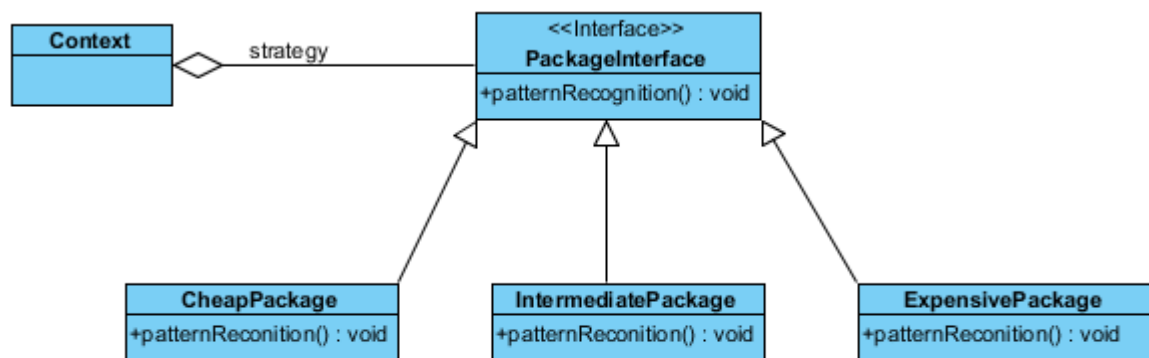


FIGURE 6 – Diagramme de classe - Configuration

L'implémentation de l'algorithme de pattern recognition diffère en fonction du package, mais ça ne change rien aux yeux du système qui peut toujours y faire appel via la même signature de méthode. Par ailleurs, si une implémentation pour un package doit être changée, alors le changement est limité à ce package uniquement.

Le pattern strategy peut être utilisé de la même façon pour toutes les opérations nécessitant une implémentation différente selon la configuration choisie par l'utilisateur.

3.6 Monitoring

Au niveau de la fonction monitoring qui permet pour rappel de surveiller de manière intelligente non seulement la maison de l'utilisateur mais également l'utilisateur lui même.

Au niveau du monitoring *passif*, consistant à surveiller l'utilisateur sans qu'il ne s'en rende compte. Il faudrait configurer le système par des patterns que l'utilisateur reproduit dans sa vie de tous les jours à l'aide de différents capteurs non-intrusifs. Le pattern observer est donc ici une solution nous semblant idéal afin d'observer l'état des différents capteurs.

Le pattern strategy peut être lui aussi mis en place dans cette fonctionnalité afin de produire des réactions différentes en fonction des données obtenues.

Au niveau du monitoring *actif*, permettant à l'utilisateur de vérifier certaine chose de son quotidien. Par exemple vérifier qu'une lampe est bien éteinte ou que l'alarme est enclenchée, etc ...

Ici aussi le pattern observer pourrait être utiliser afin de permettre la visualisation des données sur les différents supports visuels présent dans l'environnement de l'utilisateur (Smartphone, tablettes,TV,...)

3.7 Alarme

La fonctionnalité "Alarme" est ici explicitée. Comme le nom l'indique cette fonction représente l'alarme mise en place chez l'utilisateur. Cette alarme est principalement une alarme afin d'éviter les intrusions permettant ainsi de résoudre les problèmes d'insécurité des personnes âgées.

Le pattern state pourrait être utilisé afin d'implémenter cette fonctionnalité, il permet entre autre d'activer ou de désactiver la surveillance du domicile selon que l'alarme soit armée ou désarmée. Armé et désarmé serait les états principaux de l'alarme, il pourrait être envisager d'ajouter des états intermédiaire afin de convenir au mieux au besoin de l'utilisateur.

Le pattern Strategy pourrait être ici également implémenter au niveau de la puissance de l'alarme. De fait l'alarme mise en place peut varier selon les besoins de l'utilisateur. Cela peut aller de l'alarme complète à l'alarme beaucoup plus simple (exemple : simple détecteur d'ouverture de la porte ou des fenêtres) .

3.8 Connexion

Au niveau de la gestion des connexions disponibles, le pattern player role semble le plus pertinent pour répondre au différents besoins de conception. En effet, le système peut avoir accès à plusieurs types de connexions comme explicité dans la section concernant le Méta-modèle.

Cette implémentation permet donc à un ConnexionController d'avoir accès aux différents types de connexions disponibles et d'utiliser certaines connexions de façon prioritaire. Typiquement, si une connexion Wi-Fi est présente, elle sera préférée au réseau GSM.

Cela est représenté sur le diagramme suivant :

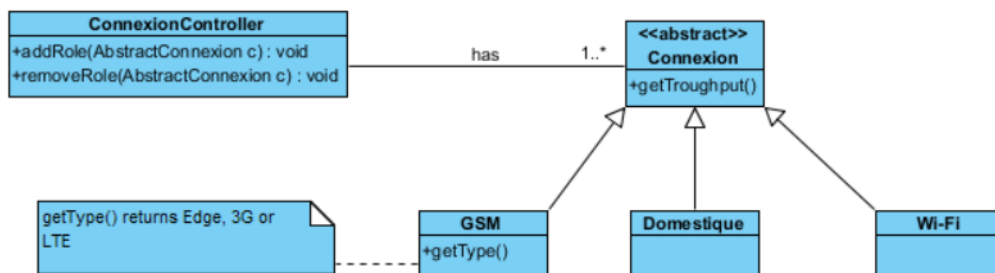


FIGURE 7 – Diagramme de classe - Connexion