Feuille de travaux pratiques n° 2
Projet DLC

Etude de cas

Ce document présente le cas d'étude général et une première analyse en UML. On s'intéresse à un système de contrôle automatique de la lumière d'une salle. Le projet de dévelopement se focalise sur une version simplifiée de l'analyse.

1 Le cas d'étude en bref

Le cas d'étude concerne un système automatisé de contrôle de lumière (*Dimmable Light Controller*). L'administration souhaite automatiser la gestion de la lumière dans les bâtiments de l'université de Kaiserslautern afin de réaliser des économies d'énergie. Ce cas a été posé aux participants d'un séminaire d'étude sur la capture des besoins à Dagstuhl en juin 1999. Le texte ci-dessous est une nouvelle rédaction librement inspirée du texte initial. Le cas étant relativement complexe, nous l'avons restreint au contrôle de lumière des pièces et pris quelques hypothèses.

Une pièce est organisée selon le schéma de la figure 1. La pièce communique avec le couloir par une porte (*p1*)

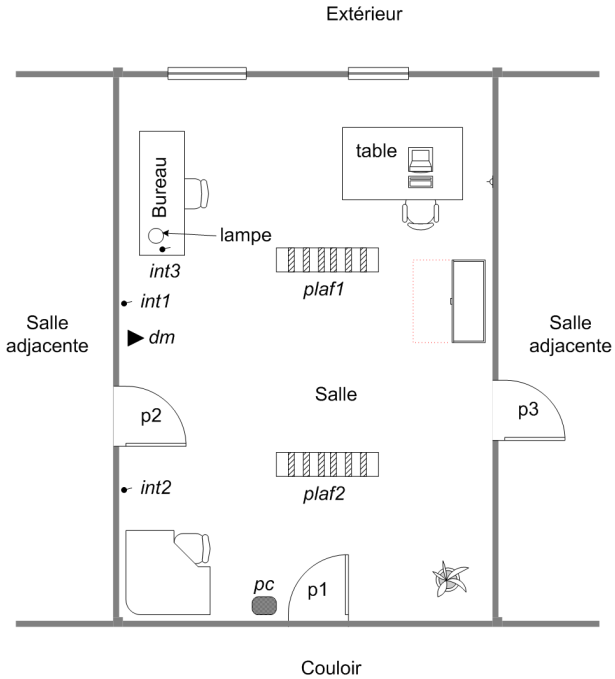

Figure 1 : *Plan d'une pièce*

et avec éventuellement des pièces adjacentes par d'autres portes (*p2* et *p3*). Une porte fait partie de la pièce si elle s'ouvre dans la pièce. Chaque porte *p* est équipée d'un capteur de contact (`cc<p>`) permettant de savoir si la porte est fermée. Le contrôle de la lumière d'une pièce utilise les équipements suivants :

– un détecteur de mouvements, composé éventuellement de plusieurs capteurs, (*dm*) qui couvre l'ensemble de la pièce,

– deux plafonniers à intensité variable (*plaf1* et *plaf2*) commandés par deux variateurs (*var1* et *var2*),
– deux interrupteurs (*int1* et *int2*) pour contrôler les plafonniers,
– une lampe de bureau et son interrupteur (*int3*),
– un panneau de contrôle amovible pour fixer l'intensité des plafonniers ou sélectionner une lumière ambiante (on suppose dans ce cas que le contrôleur sait calculer les intensités des plafonniers en fonction de l'intensité lumineuse extérieure et de l'état de la lampe de bureau),
– trois lignes d'état pour connaître l'état des sources de lumière (*le1*, *le2* et *le3*).

Les capteurs ont les caractéristiques suivantes. Le capteur de contact envoie le signal 1 (logique booléenne du courant électrique) si la porte est fermée et 0 sinon. Un interrupteur envoie le signal 1 tant qu'on appuie dessus. Le détecteur de mouvement à infra-rouge est un capteur qui renvoie le signal 1 si une personne a bougé. Il est périodiquement remis à 0. Chaque façade du bâtiment dispose d'un capteur de lumière extérieur qui mesure la luminosité entre 1 et 10000 lux. Les capteurs peuvent être actifs (impulsent des signaux) ou passifs (interrogeables). On supposera que le détecteur de mouvement et le capteur de lumière sont passifs.

La structure des plafonniers à intensité variable est organisée selon le schéma de la figure suivante. L'intensité dans le variateur varie de 0 à 100%. En dessous de 10%, la lumière est éteinte. La sortie du variateur est la ligne d'état qui fournit la valeur 1 si la lumière est allumée. Détaillons l'interface d'entrée du variateur (ce sont des variables continues qu'il faut discrétiser). La valeur d'intensité (*vali*) est le pourcentage souhaité. Elle est prise en compte avec le signal *impulsion*. L'entrée système de contrôle actif (*sca*) est positionné à 0 si le contrôleur global fonctionne. Le signal *sca* est envoyé toutes les 60 secondes. Si au bout de 60 secondes, le variateur n'a pas reçu ce signal (le système de contrôle ne communique plus), il passe en mode autonôme et sa valeur d'intensité est de 100%. Il revient en mode normal dès réception du signal *sca*. L'interrupteur fonctionne de la façon suivante. Si l'intensité est égale à 100% alors elle passe à 0 sinon elle passe à 100%.

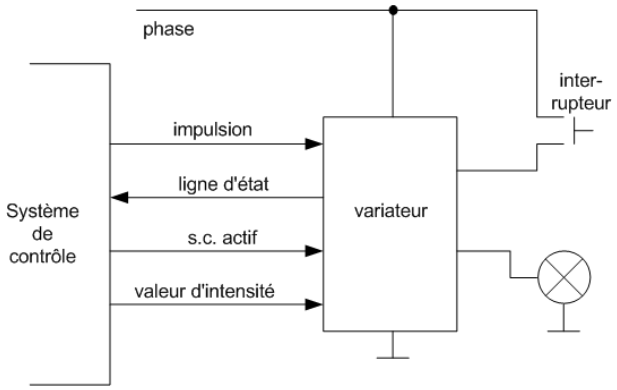

Figure 2 : *Organisation des lampes à intensité variable*

Les contraintes suivantes sont à respecter :

1. Lorsque la pièce est occupée, l'intensité lumineuse ambiante est suffisante pour pouvoir se déplacer, sauf si une intensité de lumière ambiante a été choisie.
2. La configuration de lumière ne varie pas sauf perturbation du système.
3. Si un occupant revient dans une pièce avant T1 minutes, il retrouve la même configuration de lumière que lorsqu'il a quitté la pièce.
4. Si un occupant revient dans une pièce après T1 minutes, la configuration de lumière standard est activée.
5. Lorsqu'une pièce est inoccupée depuis plus de T2 minutes, les lumières sont éteintes.
6. A partir du panneau de contrôle, l'occupant peut choisir un niveau de lumière ambiante, sélectionner la configuration standard, changer la lumière du bureau (allumer/éteindre) ou des plafonniers (allumer/ambiant/éteindre).
7. Les valeurs suivantes sont paramétrées pour chaque pièce par l'administrateur du système : T1, T2, configuration standard.
8. Le dysfonctionnement des capteurs de lumière ou de mouvement est signalé au contrôleur de la pièce.
9. Si le capteur de lumière extérieur ne fonctionne pas, on utilise sa dernière valeur correcte, la configuration de lumière standard est alors que tous les plafonniers sont allumés.

10. Si le détecteur de mouvement ne fonctionne pas correctement, on suppose que la pièce est occupée.
11. Si les contrôleurs sont en panne, les lumères doivent être allumées.

Cible et Objectifs

Le cas porte sur un équipement domotique simplifié, la gestion du contrôle de la lumière dans une salle en considérant deux dispositifs, un *client* sous forme de télécommande ou de smartphone et un *serveur*, le logiciel de contrôle de la lumière, qui pilote les différents dispositifs physique.

Il s'agit de concevoir et mettre en œuvre un logiciel de gestion de portes, dont l'analyse du domaine et les spécifications des exigences logicielles (fonctionnelles et non fonctionnelles) sont fournies.

Le prototypage se fera en utilisant des automates LEGO MINDSTORMS EV3, un ensemble d'outil de construction, programmation et de commande de robots LEGO.

2 Analyse du besoin, périmètre et hypothèses

Il s'agit d'un d'un système de contrôle de processus, ayant de nombreux événements à prendre en compte. Dans un système réactif, un événement génère presque systématiquement une réaction palpable tandis que, pour le contrôle de lumière, certaines occurrences d'événements n'ont pas d'effets immédiats ou perceptibles. Le temps de réaction n'est pas crucial et les conditions de réactions sont plus complexes.

La structure du système met plus en évidence les objets que leurs classes. Dans ce type de système, il existe souvent des classes avec une seule instance. Enfin, le comportement dynamique des objets est de première importance pour ce type d'application. C'est pourquoi nous développerons particulièrement les diagrammes d'états-transitions.

Ce système peut être qualifié de complexe car il prend en compte des aspects orthogonaux : événements (mouvements, actions), contraintes et événements temporels (T1, T2), valeurs continues (lignes d'état, valeur d'intensité) et discrètes (impulsions), informations (configurations prédéfinies, valeurs courantes), pilotage de dispositifs physiques (capteurs, plafonniers, lampe de bureau), lois physiques et standards électriques, sûreté de fonctionnement (lumière suffisante pour se déplacer, mode dégradé, défaillances), etc.

Il est important de noter que le cas initial a été simplifié car il prenait en compte des caractéristiques physiques des capteurs ou des actionneurs qui sont ignorées dans cet exercice telles que le temps de réaction, l'échantillonnage et la discrétisation de valeurs continues. La solution est développée en trois étapes : spécification globale des besoins (UC + diagramme de classes exploratoire), spécification détaillée des besoins (UC + séquences), spécification des types (diagramme de classes et diagrammes états-transitions).

2.1 Spécification globale des besoins.

La spécification des besoins permet de délimiter, d'organiser et de clarifier le cas d'étude. Deux approches sont possibles pour ce type de système :

- Analyse dirigée par les événements : l'analyste recense les événements pouvant intervenir sur le système, les classe et examine les réactions du système. C'est une méthode lourde et rébarbative, dans laquelle les risques d'oublis sont importants. C'est une méthode qui convient aux systèmes réactifs purs : événement-réponse. Dans notre cas, la réaction du système dépend de nombreux paramètres, qui forment son contexte courant.
- Analyse dirigée par les cas d'utilisation : on découpe le besoin en modules relativement indépendants et plus facilement appréhendables. Cette approche permet une meilleure abstraction.

Nous préférons la seconde approche, plus générale et plus complète, puisque l'étude des scénarios est une analyse (réduite) dirigée par les événements.

La première étape dans toute analyse de système est la délimitation du contexte du système. C'est un élément essentiel dans la structuration du besoin et toute la description logique du système en dépendra. Si l'administrateur et les usagers sont clairement des acteurs externes, on peut se poser la question pour les différents capteurs, les portes et les lampes (bureau et plafonniers). Par exemple, on peut supposer que la porte ou le plafonnier sont des systèmes indépendants, ou encore des sous-systèmes, qui communiquent avec le système de contrôle de la lumière. Les objets interfaces seraient alors des cartes d'acquisition et de pilotage. Dans notre cas, on cherche dans un premier temps à simuler le système, c'est pourquoi on va modéliser le comportement des capteurs et des lampes à l'intérieur du système.

Hypothèse 1 (contexte) *Les capteurs et les lampes sont des éléments constituants du système modélisé.*

Le contexte est alors défini par trois acteurs : l'administrateur, les usagers et l'environnement. L'administrateur est chargé du paramétrage standard et du pilotage du système (initialisation/arrêt). L'usager fixe les paramètres personnalisés et les changements de configuration (mouvements, usage des interrupteurs ou du panneau de contrôle). L'environnement représente le contexte physique du système. Il est utilisé pour fixer la valeur en lux de la lumière extérieure, générer les pannes ou les dysfonctionnements. Dans une perspective de simulation, l'environnement représente une interface qui permet de générer tous les événements physiques, le contexte de la simulation.

L'architecture générale de l'application est de type capteur/contrôle/actionneur. La description des différents composants physiques ne fait pas partie des cas d'utilisation. Ils sont traités à part, lors de la description des classes de ces composants.

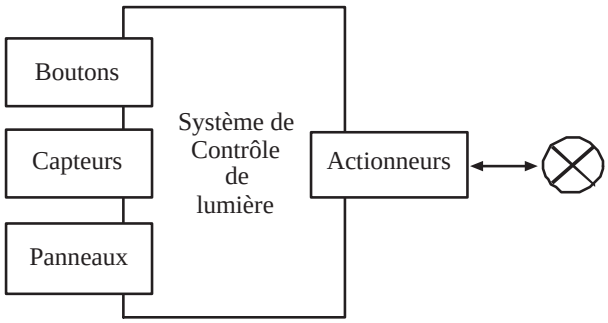

Figure 3 : *Architecture générale - Lumière*

On distingue quatre contextes de fonctionnement : les besoins de l'usager, les besoins de l'administrateur, les pertubations du système et le mode dégradé. Ils donnent lieu à quatre cas d'utilisation. Nous les traçons dans un seul diagramme, mais on peut imaginer une organisation en quatre catégories et donc quatre sous-diagrammes.

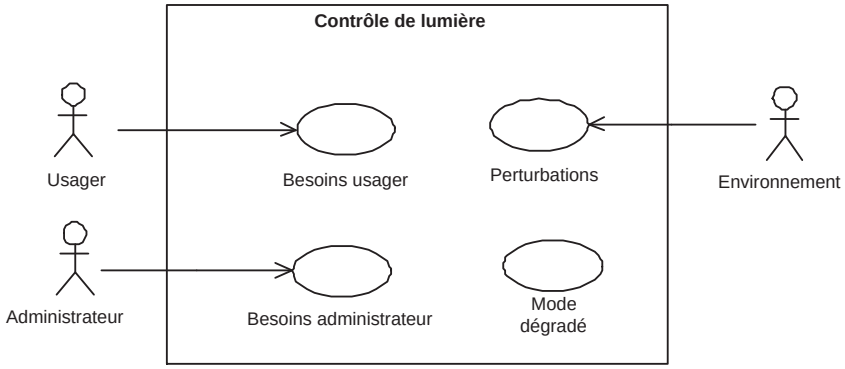

Figure 4 : *Cas d'utilisation - Lumière*

Lors de l'établissement des cas d'utilisation, nous avons été amenés à préciser certains points de l'énoncé (oublis, ambiguïtés, incohérences, précisions, etc.) sous forme d'hypothèses. Dans la réalité, ces hypothèses devront être validées par le demandeur ou les experts.

Le texte de l'énoncé semble incohérent : il indique que les capteurs de mouvement et de lumière sont passifs (dernière phrase du paragraphe sur les capteurs) et que leur dysfonctionnement est signalé au contrôleur (contrainte C8). De même la spécification du capteur indique qu'il renvoie 1 si une personne a bougé.

Hypothèse 2 (capteur) *Les capteurs de mouvement et de lumière sont passifs. La contrainte C8 est supprimée.*

L'énoncé met en évidence des pannes pour le détecteur de mouvement, ou le capteur de lumière extérieure. Il s'agit de capteurs complexes. On suppose que le capteur de contact de la porte ne tombe pas en panne. Il s'agit en effet d'un capteur simple, qui ne peut pas gérer sa défaillance.

Hypothèse 3 (défaillance capteur) *On suppose que le capteur de contact peut être défaillant ; dans ce cas, il n'émet aucun signal. Les défaillances des capteurs sont gérées par les composants eux-même.*

L'énoncé pose des contraintes sur l'état de la lumière en fonction de la présence et des délais d'absence. Il n'est pas précisé comment est définie l'occupation de la pièce : soit on se base uniquement sur le capteur de présence (si pas de mouvements au bout d'un certain temps, la pièce est supposée inoccupée), soit on couple les contacteurs de porte avec le détecteur de présence (si pas de mouvements au bout d'un certain temps après une fermeture de porte, la pièce est supposée inoccupée).

Hypothèse 4 (occupation) *L'occupation d'une pièce est définie par couplage du détecteur de mouvement et des capteurs de porte : si après un mouvement de porte (porte refermée), il n'est pas détecté de mouvement pendant un temps T3, la pièce est supposée inoccupée et les lumières sont éteintes.*

Hypothèse 5 (plafonnier) *Le plafonnier est considéré comme un tout (variateur + lampe + contrôle).*

Hypothèse 6 (lampe de bureau) *Habituellement, l'interrupteur d'une lampe de bureau est branché en série, c'est-à-dire que la lampe est pilotée en dernier ressort par l'interrupteur. L'énoncé indique qu'elle est aussi commandable depuis le panneau de contrôle, celà signifie alors qu'il y a un circuit parallèle. C'est sur ce circuit parallèle que se trouve la ligne d'état.*

Hypothèse 7 (luminosité) *Pour chaque source lumineuse (plafonnier, lampe de bureau, extérieur), on dispose d'une luminosité (captée, fixée ou calculée) qui sert au calcul de la luminosité ambiante. La luminosité de la lampe de bureau et des plafonniers est fixée lors du paramétrage par l'administrateur, en fonction de la puissance des lampes.*

Hypothèse 8 (ampoule défaillante) *L'énoncé ne précise pas ce qui se passe lorsque les ampoules sont défaillantes ("grillées"). Il semble difficile d'ajouter de l'intelligence aux lampes. On suppose que la ligne d'état de la source lumineuse permet de savoir si la source est hors d'usage.*

Hypothèse 9 (défaillance électrique) *Si l'installation électrique est en panne, le contrôleur est débranché, on passe en mode manuel (seuls les interrupteurs directs fonctionnent).*

On peut dès lors établir un diagramme exploratoire des classes à partir des éléments physiques du système.

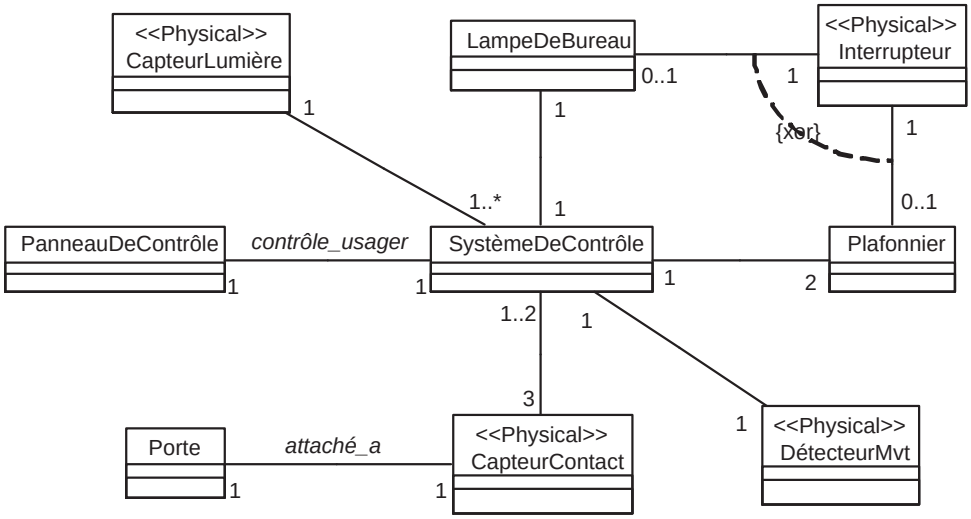

Figure 5 : *Diagramme de classes exploratoire - Lumière*

2.2 Spécification détaillée des besoins fonctionnels.

Détaillons maintenant les cas d'utilisation. Pour des raisons de place, nous ne les illustrons pas par des scénarios simples mais nous développons directement les diagrammes de séquence et de collaboration représentatifs. Les noms d'objets n'ont pas été soulignés. Les contraintes de l'énoncé sont signalées par leur numéro : C1, C2... C11.

Cas d'utilisation : `Besoins administrateur`	
invariant :	`C7, T3 <T1 < T2`
acteurs primaires :	`Administrateur`

description

Les besoins de l'administrateur décrivent le démarrage, le paramétrage de base et l'arrêt du système.

cas : `Paramétrage`

A partir d'un clavier spécifique, l'administrateur peut choisir la configuration standard, fixer les temporisations T1, T2 et T3. La configuration courante est définie par l'état de chaque lampe et d'un niveau d'intensité pour les plafonniers lorsqu'ils sont en position de lumière ambiante. L'administrateur peut tester la configuration choisie avant de la mémoriser. La configuration standard est soit une mémorisation de la configuration courante soit un calcul pour une luminosité donnée.

cas : `Mise en route, arrêt`

L'administrateur démarre ou arrête le système. Lorsque le système est arrêté, seuls les interrupteurs directs fonctionnent. Au démarrage, une vérification des composants physique est effectuée puis l'administrateur initialise les paramètres du système et le configure (voir cas `Paramétrage`).

Le démarrage et l'arrêt ne sont détaillés qu'une fois connus les composants du système. La vérification des composants physique est reportée à la conception détaillée du matériel. On supposera simplement une opération `init` d'initialisation des composants. La description peut être complétée par un dessin du clavier spécifique, si on considère que son ergonomie fait partie du besoin.

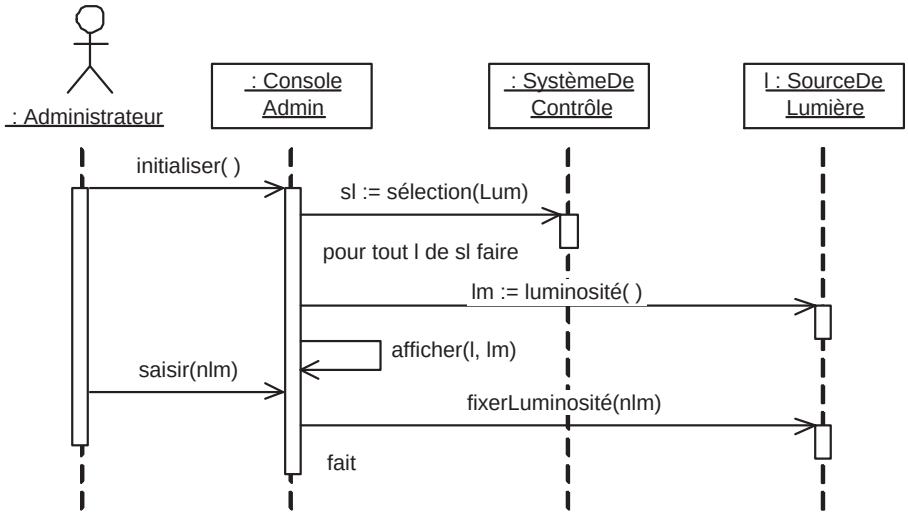

Figure 6 : *Diagramme de séquence, initialisation - Lumière*

On notera que la luminosité (maximale) des lampes est saisie, et qu'une valeur par défaut est fixée pour la source de lumière extérieure. Cette valeur par défaut est utile lorsque le capteur est défaillant.

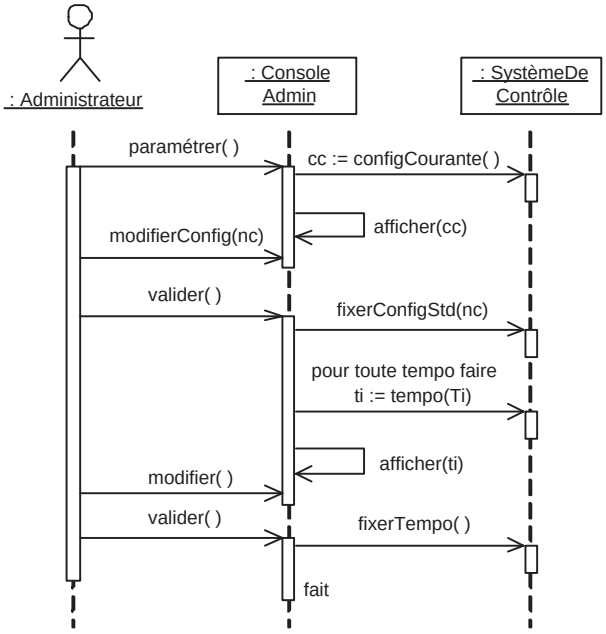

Figure 7 : *Diagramme de séquence, paramétrage - Lumière*

Cas d'utilisation : `Besoins usager`	
acteurs primaires :	`Usager`
invariant :	`C6`
Les contraintes portent sur les actions de l'usager.	
description	
Les besoins de l'usager décrivent l'interaction entre l'usager et le système de contrôle : actions sur les interrupteurs ou le panneau de contrôle. Les mouvements sont supposés être des pertubations.	
cas :	`Configuration`
A partir du panneau de contrôle, l'occupant peut choisir un niveau de lumière ambiante, sélectionner la configuration standard, changer la lumière du bureau (allumer/éteindre) ou des plafonniers (allumer/ambiant/éteindre), éteindre toutes les lumières. Dans le cas d'une lumière ambiante, le panneau de contrôle permet d'augmenter ou de diminuer l'intensité de chaque plafonnier. Une configuration de lumière (on parle aussi de scène de lumière ou de lumière ambiante) est la définition d'une intensité pour chaque plafonnier. La configuration est soit une mémorisation de la configuration courante soit un calcul pour une luminosité donnée. Dans ce dernier cas, le calcul est fonction de l'état et de la luminosité de chaque source lumineuse.	
cas :	`Intervention`
L'occupant peut à tout moment agir sur les interrupteurs ; l'effet est le même qu'à partir du panneau de contrôle (allumer/éteindre), mais sans passer par le système de contrôle puisque le schéma de fonctionnement montre que l'interrupteur est lié directement au variateur. L'interrupteur pour un plafonnier fonctionne de la façon suivante. Si l'intensité est égale à 100% alors elle passe à 0 sinon elle passe à 100%.	

Cette interaction met en évidence la commande des lampes par le système de contrôle. Le parallélisme a été introduit uniquement pour signifier que la séquencialité n'est pas impérative. Le message `état` d'une lampe consulte la ligne d'état.

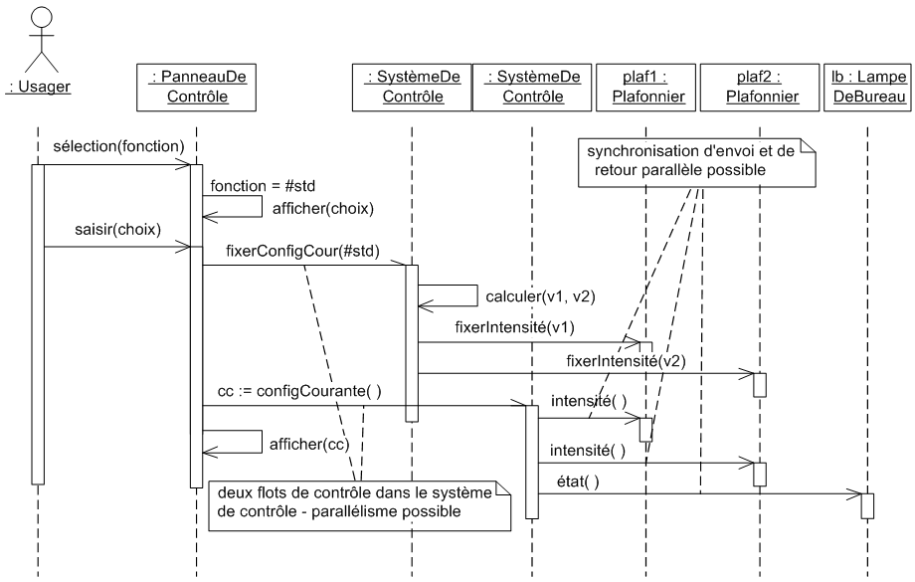

Figure 8 : *Diagramme de séquence, choix d'une configuration standard - Lumière*

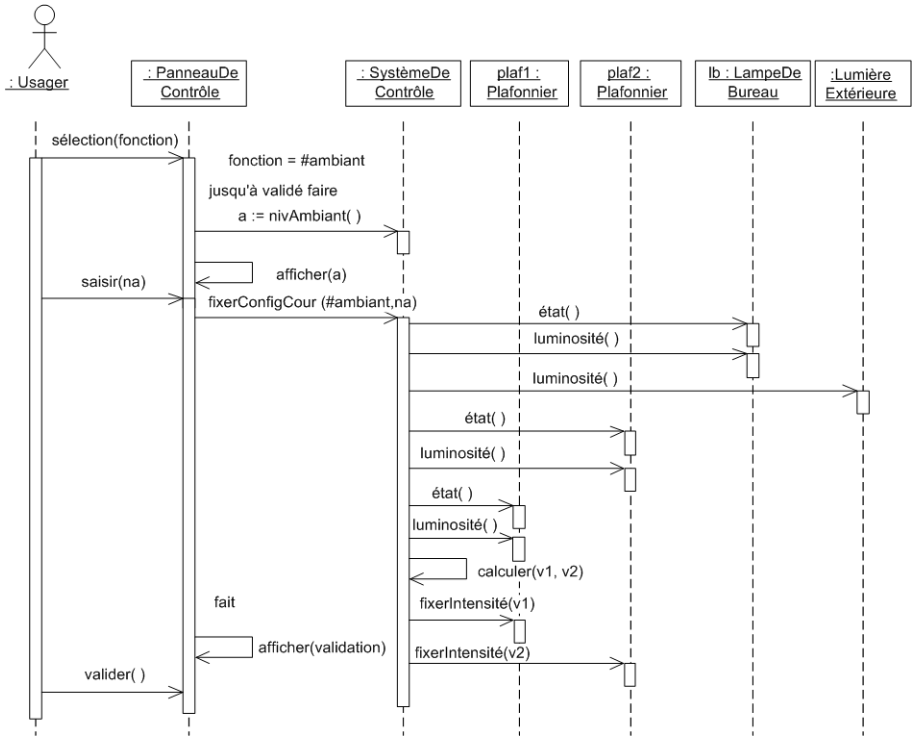

Figure 9 : *Diagramme de séquence, choix d'une lumière ambiante - Lumière*

Pour fixer l'intensité lumineuse, on teste l'état de chaque source lumineuse, le calcul prend en compte l'état des dispositifs (normal, défaillant) et la valeur de leur luminosité. Lorsque l'état est normal, la lumière ambiante est calculée à partir de la luminosité réelle (capteur extérieur) ou maximale des installations (lampes). Si le capteur est dans un état défaillant, la dernière valeur mémorisée est reprise (voir aussi UC `Pertubations`).

Examinons maintenant l'usage des interrupteurs et du panneau pour changer la lumière d'un plafonnier.

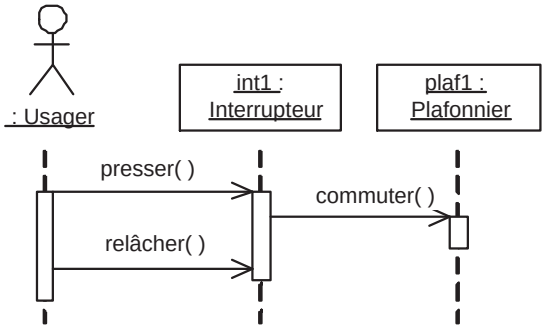

Figure 10 : *Diagramme de séquence, interrupteur direct de plafonnier - Lumière*

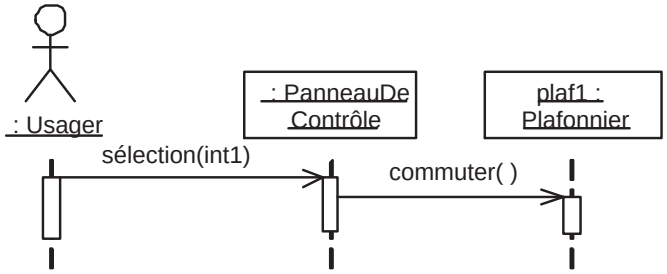

Figure 11 : *Diagramme de séquence, interrupteur panneau de plafonnier - Lumière*

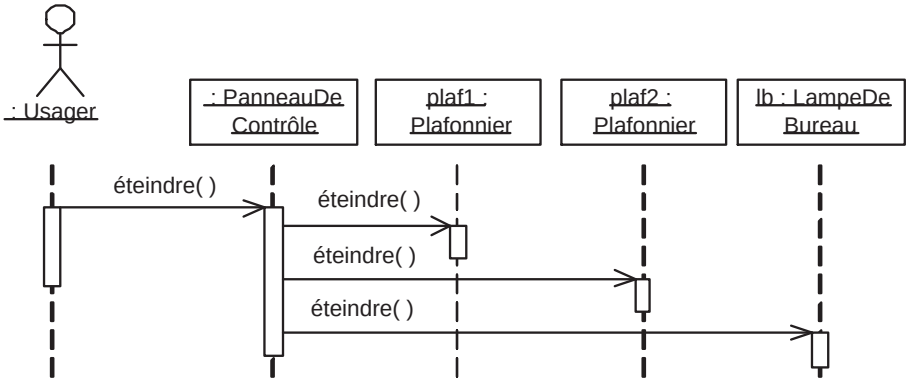

Figure 12 : *Diagramme de séquence, extinction - Lumière*

Cas d'utilisation : `Perturbations`	
invariant :	`C1, C2, C3, C4, C5, C9, C10`
acteurs primaires :	`Environnement`

description

Les perturbations sont engendrées par des événements de l'environnement physique autres que les commandes des usagers et des administrateurs : événements périodiques, mouvements, défaillances diverses.

cas : `Lumière extérieure`

Chaque façade du bâtiment dispose d'un capteur de lumière extérieur qui mesure la luminosité entre 1 et 10000 lux. Le capteur de lumière est passif car il ne répercute aucun événement sur le système de contrôle. Il peut tomber en panne. La luminosité extérieure est collectée périodiquement et le contrôleur est informé.

cas : `Mouvement`

Le détecteur de mouvement détermine la présence d'un usager dans la pièce. L'occupation d'une pièce est définie par couplage du détecteur de mouvement et des capteurs de porte : si après un mouvement de porte (porte refermée), il n'est pas détecté de mouvements pendant un temps T3, la pièce est supposée inoccupée et les lumières sont progressivement éteintes. Le détecteur de mouvement est un capteur passif.

Lorsque la pièce est occupée, l'intensité lumineuse ambiante est suffisante pour pouvoir se déplacer, sauf si une intensité de lumière ambiante a été choisie. Si un occupant revient dans une pièce avant T1 minutes, il retrouve la même configuration de lumière que lorsqu'il a quitté la pièce. Si un occupant revient dans une pièce après T1 minutes, la configuration de lumière standard est activée. Lorsqu'une pièce est inoccupée depuis plus de T2 minutes, les lumières sont éteintes.

. . .

	...
exceptions	
cas :	`C9: dysfonctionnement du capteur de lumière.`
précondition :	Le capteur de lumière extérieur ne fonctionne pas.
résultat :	On utilise sa dernière valeur correcte, la configuration de lumière standard est alors que tous les plafonniers sont allumés.
cas :	`C10: dysfonctionnement du détecteur de mouvement`
précondition :	Le détecteur de mouvement ne fonctionne pas correctement.
résultat :	La pièce est occupée.

Un objet `LumièreExtérieur` est responsable de l'acquisition de la luminosité. Il collecte périodiquement la luminosité sur le capteur. Si la luminosité change, le contrôleur est informé (le traitement effectué alors est celui de la configuration d'une lumière ambiante (test et commande des lampes - cf figure 9). Le test d'état sera à affiner une fois connu le comportement dynamique du capteur.

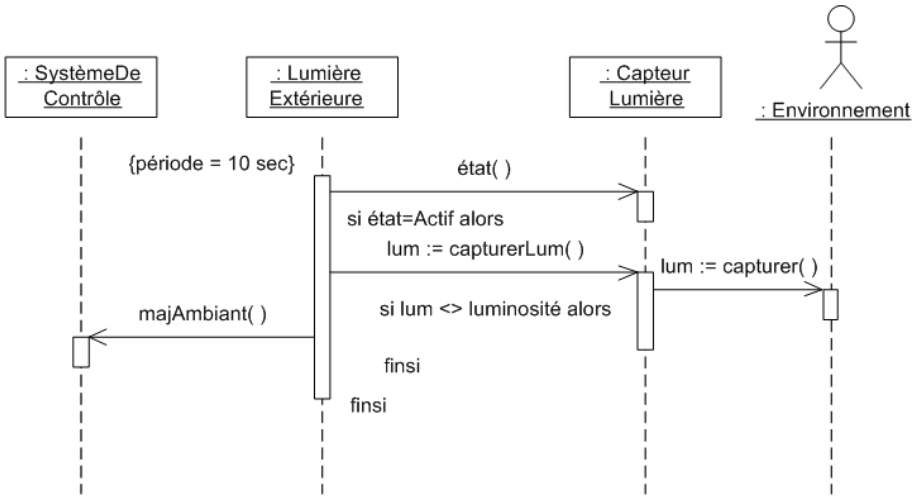

Figure 13 : *Diagramme de séquence, acquisition luminosité extérieure - Lumière*

Ce capteur peut être défaillant.

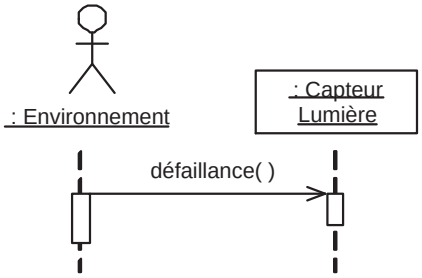

Figure 14 : *Diagramme de séquence, défaillance capteur lumière extérieure - Lumière*

La panne de capteur de contact est identique mais elle n'est pas gérée par le contrôleur, selon les hypothèses de départ.

Le mouvement est perçu par l'environnement et non un usager, car il peut s'agir d'autre chose (feuille déplacée par le vent, animal...). On peut coupler le capteur de mouvement avec les contacteurs de porte pour définir l'occupation de la salle.

Ce capteur peut être défaillant.

La gestion des temporisations peut être effectuée par le système de contrôle ou déléguée à un composant spécifique, à des réveils (armer/réveiller). Dans l'étude du déplacement, le contrôleur délègue les temporisations à un objet spécifique. Plusieurs scénarios de sortie sont examinés, selon les contraintes de temps.

La sortie peut être définitive.

Sous contrôle du système, les plafonniers évoluent après réception d'une impulsion.

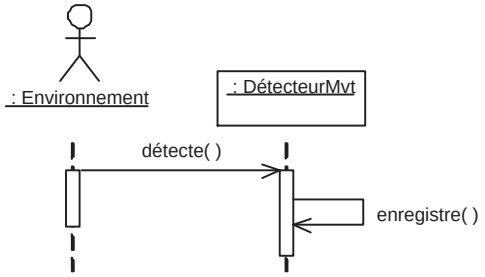

Figure 15 : *Diagramme de séquence, mouvement - Lumière*

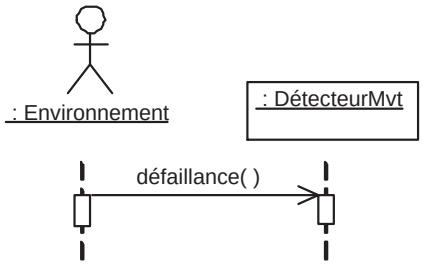

Figure 16 : *Diagramme de séquence, défaillance capteur mouvement - Lumière*

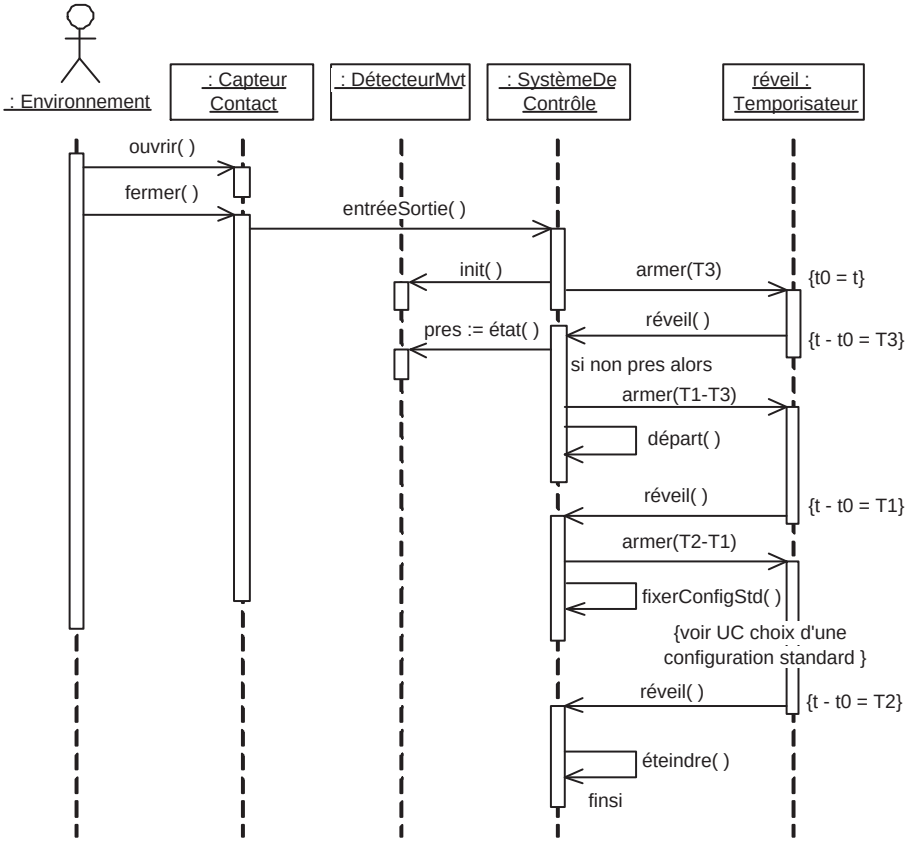

Figure 17 : *Diagramme de séquence, déplacement/sortie définitive - Lumière*

Le retour après sortie se fait avant le temps T2.
Le retour après sortie se fait avant le temps T1.

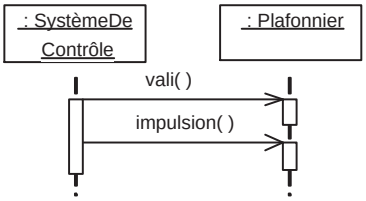

Figure 18 : *Diagramme de séquence, signaux du contrôleur - Lumière*

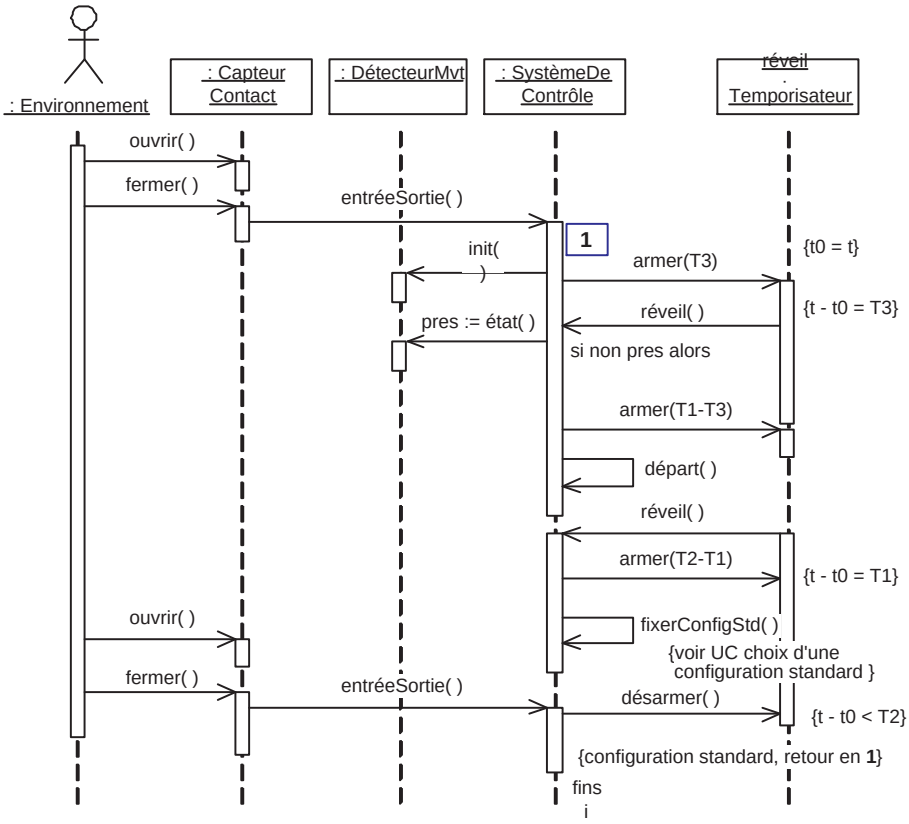

Figure 19 : *Diagramme de séquence, déplacement/retour avant T2 - Lumière*

Cas d'utilisation : `Mode dégradé`
invariant : `C11`
description
Si les contrôleurs sont en panne, les lumières doivent être allumées. Tous les événements liés au contrôleur sont ignorés (ils peuvent intervenir). Par contre l'usage des interrupteurs directs est accepté, mais cela sort de notre contexte. Le contrôleur est remis en route par l'administrateur.

Il n'y a pas d'interaction spécifique pour ce cas.

NB : il est important de noter que la conception va transformer certains comportements : on doit tenir compte des spécifications d'interface des composants physiques.

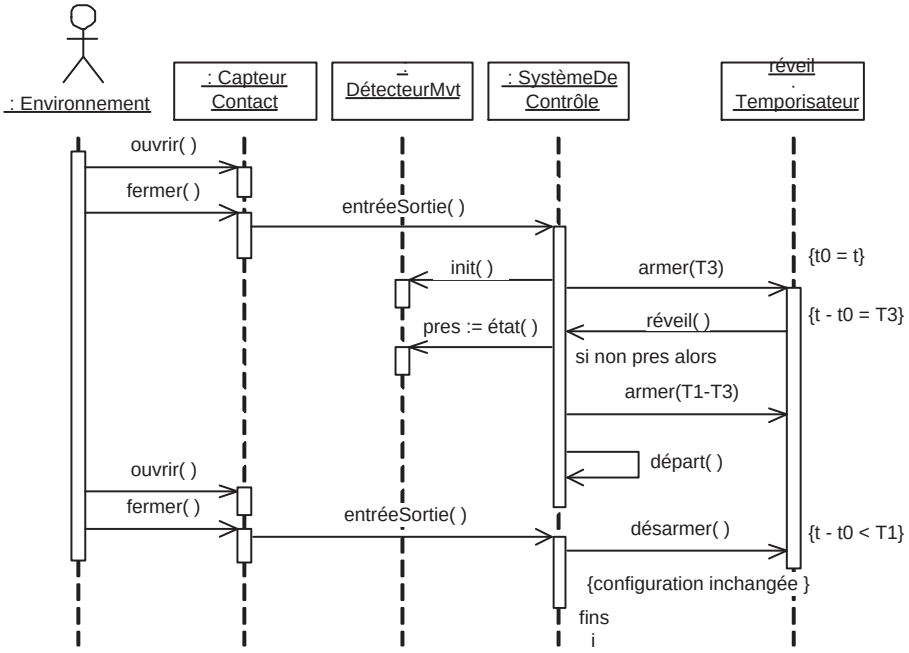

Figure 20 : *Diagramme de séquence, déplacement/retour avant T1 - Lumière*

3 Analyse

A partir des objets identifiés dans les diagrammes de séquence, un diagramme de classes est construit et les diagrammes états-transitions sont élaborés. La présentation est organisée en trois partie : hiérarchie des capteurs, hiérarchie des lumières, et système de contrôle.

Nous avons affiné les directions des liens navigables, en fonction de la relation entre le système de contrôle et ses composants et des besoins définis dans l'étude des comportements dynamiques :

– Les capteurs passifs (lumière extérieure, mouvement) n'ont pas accès à leur contrôleur, ils sont interrogés.
– Les consoles et les capteurs de contact ont une communication unidirectionnelle avec le contrôleur.
– Les interrupteurs ont une communication dans un seul sens avec leur lampe : en entrée.

D'autre ont une communication bi-directionnelle.

– Les lampes communiquent dans les deux sens avec le contrôleur (état/commande).
– Le temporisateur collabore avec le contrôleur (armer/réveiller).

3.1 Capteurs

Le comportement des capteurs est organisé dans une hiérarchie de spécialisation.

Le comportement commun (abstrait) d'un capteur est défini comme suit.

Le détecteur de mouvement est un capteur dont l'état `EnFonctionnement` est redéfini en deux sous-états : pas de mouvement ou mouvement détecté. A l'initialisation, il ne détecte rien. Lorsqu'un mouvement est signalé, il passe dans l'état mouvement détecté, une opération interne `enregistre` est alors exécutée.

Le capteur de lumière est un objet passif dont le comportement est simplement celui d'un capteur enrichi d'une transition pour l'état `EnFonctionnement`.

Le capteur de contact est un capteur dont l'état `EnFonctionnement` est affiné en deux états : ouvert/fermé. Lors de l'ouverture, il informe de système d'une entrée/sortie d'usager.

3.2 Lumières

Le comportement des lumières est organisé dans une hiérarchie de spécialisation.

La lumière extérieure, c'est-à-dire l'objet chargé de piloter le capteur de lumière, est un objet actif qui interroge périodiquement le capteur de lumière. On ne considère pas d'automate pour fixer son comportement interne. On peut le coupler à un temporisateur.

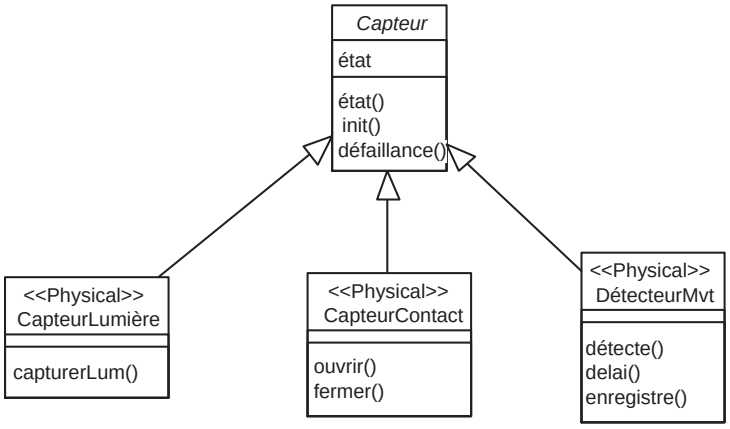

Figure 21 : *Diagramme de classes des capteurs - Lumière*

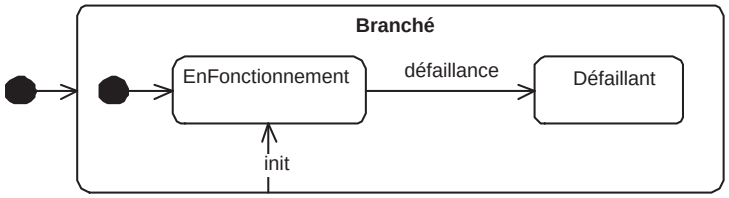

Figure 22 : *Diagramme états-transitions d'un capteur - Lumière*

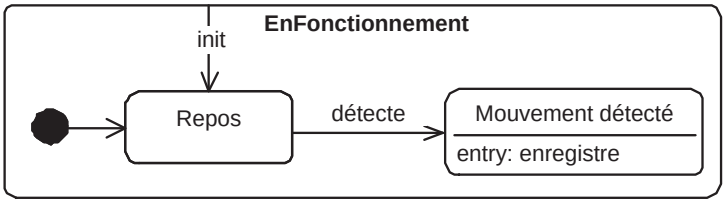

Figure 23 : *Diagramme états-transitions du détecteur de mouvement - Lumière*

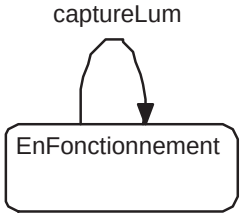

Figure 24 : *Diagramme états-transitions du capteur de lumière - Lumière*

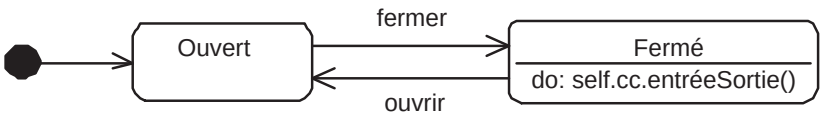

Figure 25 : *Diagramme états-transitions du capteur de contact - Lumière*

Chaque interrupteur est relié à une seule lampe, chaque lampe dispose d'un seul interrupteur. La contrainte {xor} est assurée par la symétrie des deux associations navigables.

```
context LampeDeBureau
inv symetrie:
```

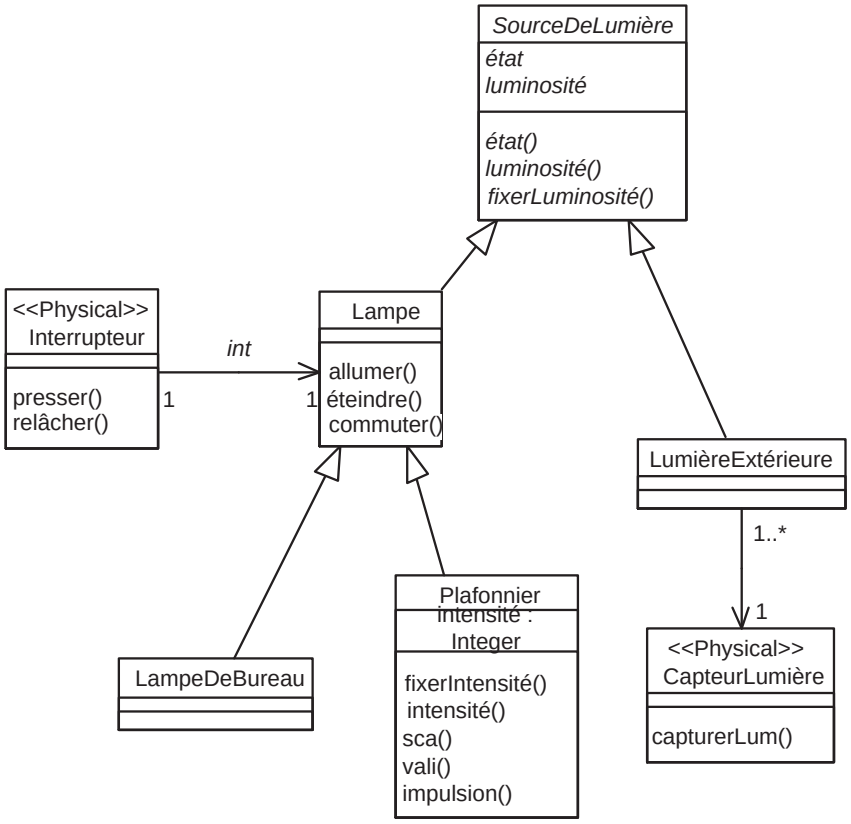

Figure 26 : *Diagramme de classes des capteurs - Lumière*

```
    self.int3.lampe = self
context Plafonnier
inv symetrie:
    self.int.lampe = self
```

L'interrupteur envoie le signal 1 tant qu'on appuie dessus (événement `commuter`). Le destinataire du signal est la lampe associée à l'interrupteur.

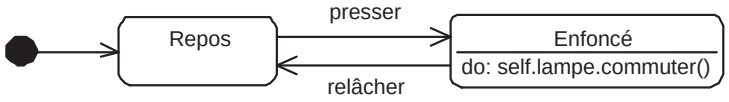

Figure 27 : *Diagramme états-transitions de l'interrupteur - Lumière*

Le comportement commun (abstrait) d'une lampe est défini comme suit. Une lampe se trouve dans deux états, `allumé` ou `éteint`. Deux événements mènent dans ces états. L'opération `état` renvoie l'état de la lampe (ligne d'état). La commutation change d'état.

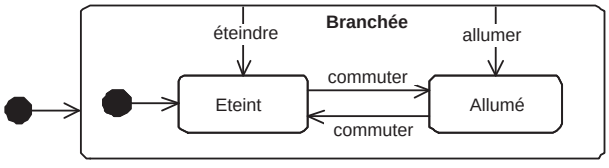

Figure 28 : *Diagramme états-transitions de la lampe - Lumière*

La lampe de bureau est une lampe, son automate d'état est hérité. Un plafonnier est une lampe au comportement plus complexe. Sa luminosité est le produit de son intensité par sa luminosité maximale (fixée par l'administrateur).

La conception détaillée du composant séparera le composant physique (commande, état, intensité) et la partie logique (luminosité). L'état `Branché` est redéfini dans la classe `Plafonnier` par un état composite concurrent défini par un mode et une lumière.

Le mode est autonome si le contrôleur n'a pas envoyé le signal `sca` depuis 60 secondes. Le délai de 60 sec peut être paramétré par une temporisation T4, en fonction des paramètres acceptés par le composant physique.

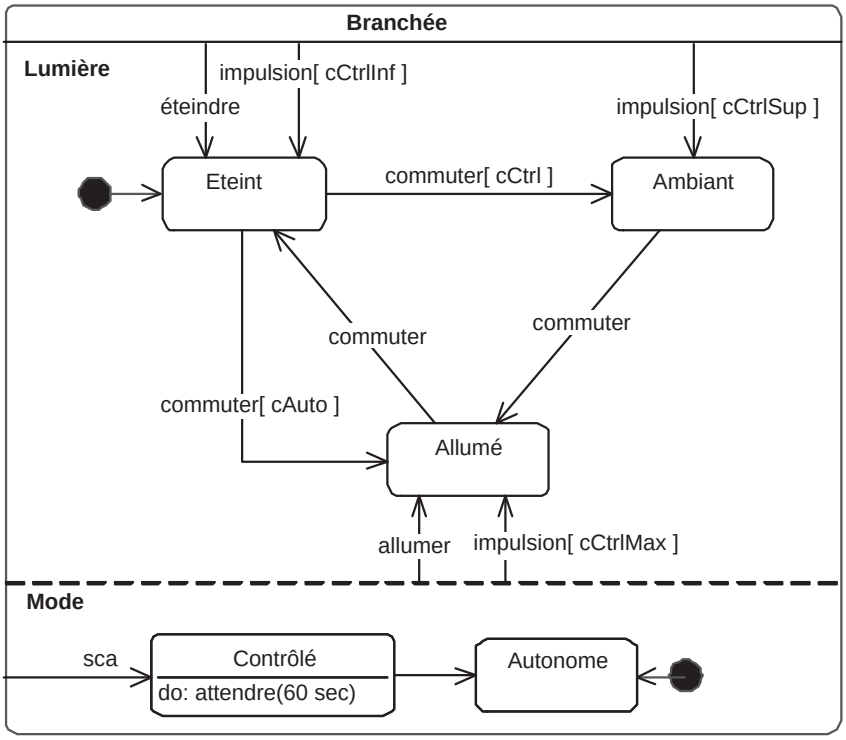

Figure 29 : *Diagramme états-transitions d'un plafonnier - Lumière*

Dans le mode autonome, la lumière est soit allumée soit éteinte. Dans le cas contraire, l'intensité définit l'état (éteint si ¡ 0 %, allumé si 100 %, ambiant autrement). Les gardes sont définies par des conditions sur l'état du plafonnier :

```
context Plafonnier::cCtrl() = (self.state = Branchée::Mode::Contrôlé)
context Plafonnier::cAuto() = (self.state = Branchée::Mode::Autonome)
context Plafonnier::cCtrlInf() = self.cCtrl() and vali < 10
context Plafonnier::cCtrlSup() = self.cCtrl() and vali >= 10 and vali < 100
context Plafonnier::cCtrlMax() = self.cCtrl() and vali = 100
```

Plusieurs opérations de la classe n'apparaissent pas car elles ne sont pas spécifiques aux états. La luminosité est fonction de l'état, de l'intensité et de la luminosité maximale. Cette dernière est définie par l'opération `fixerLuminosité`. L'événement `vali` (valeur d'intensité) met à jour l'intensité (à représenter par un attribut privé). L'intensité n'est réellement mise à jour qu'à l'occurrence d'une impulsion (intensité = vali). Les impulsions sont ignorées si le plafonnier est autonome. L'opération `état` prend ses valeurs uniquement dans le sous-état `Lumière`.

3.3 Système de contrôle

Le système de contrôle comprend l'ensemble des composants du système. Les plafonniers sont distingués : le plafonnier `plaf1` est celui près de la fenêtre. Les capteurs de porte n'ont pas été distingués comme cela est fait pour les plafonniers car ils ne sont pas pilotés par le système. Les associations sont nommées pour simplifier les expressions de navigation.

Nous avons affiné certaines associations en agrégation ou composition. Le capteur de lumière extérieure et les liens temporaires (stéréotype `temporaire`) ne sont pas des agrégations. Les consoles (panneau de contrôle, console administrateur) transmettent les requêtes qu'ils reçoivent, il n'y a pas d'automate.

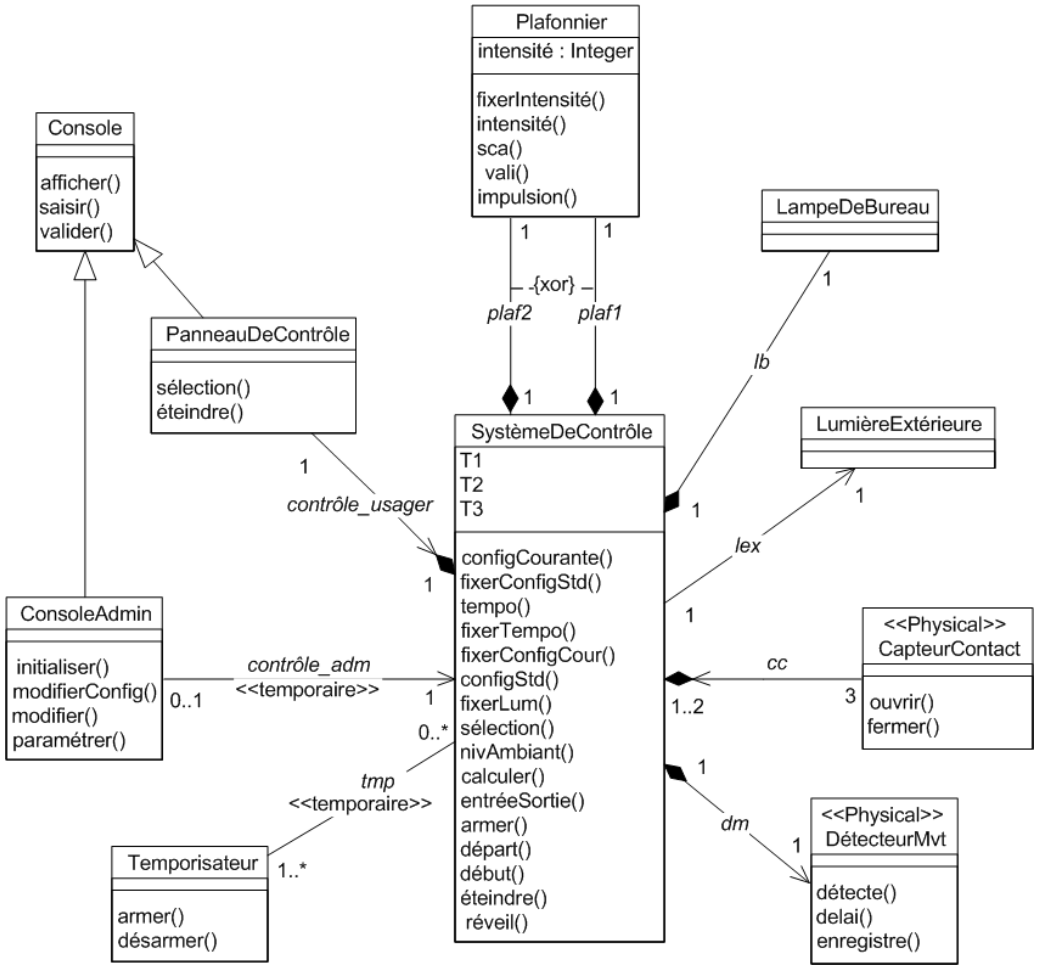

Figure 30 : *Diagramme de classes du contrôle - Lumière*

Il reste à définir le comportement dynamique du contrôleur. On reprend toutes les spécifications précédentes pour établir l'automate du contrôleur (cas d'utilisation, séquences, automates, classes). Certaines séquences se traduisent par une opération unique, tandis que d'autres correspondent à différentes transitions du diagramme état-transitions.

On distingue trois états : `configuration`, `panne` et `fonctionnement`. Les transitions correspondantes n'ont pas été vues dans les diagrammes de séquence. On pourra les y ajouter.

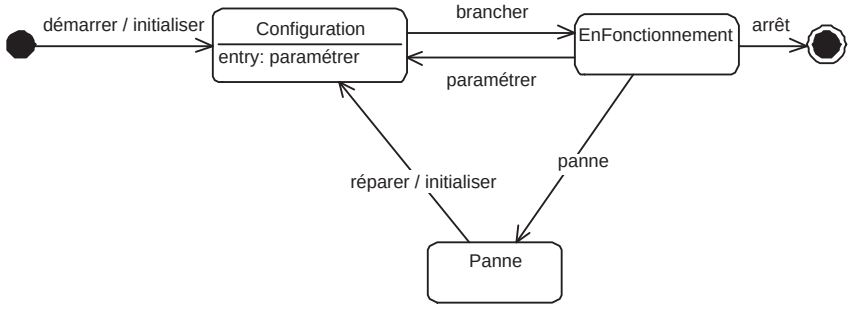

Figure 31 : *Diagramme états-transitions du contrôle - Lumière*

La configuration se réduit, pour nous, au cas d'utilisation `Besoins administrateur`. On suppose que les lumières passent en mode manuel lorsque l'administrateur configure le système. Les opérations `initialiser` et `paramétrer` doivent être conformes aux diagrammes de séquence des figures 6 et 7. Ces opérations n'induisent

pas d'autres états et transitions mais on pourra les décrire par des diagrammes d'activités.

N'ayant pas plus d'information sur la gestion des pannes (cas d'utilisation `Mode dégradé`), nous ne détaillons pas cette partie.

L'état `EnFonctionnement` du système de contrôle est relativement complexe au premier abord, car plusieurs activités sont menées en parallèle : suivi des événements (capteurs), réception des commandes des consoles, gestion des temporisations, pilotage des lampes, etc. Elle comprend les éléments de spécification relatifs aux cas d'utilisation `Besoins usager` et `Perturbations`.

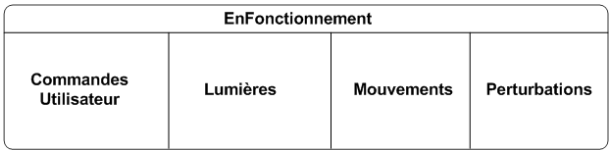

Figure 32 : *Diagramme états-transitions du contrôle état `EnFonctionnement` - Lumière*

Parmi les commandes de l'usager, seules celles effectuées à partir du panneau de contrôle sont à prendre en compte car les actions sur les interrupteurs ne passent pas par le système de contrôle. Choisir une configuration, allumer et éteindre sont des opérations invocables à tout moment. Il s'agit de commandes vers les lampes. Elles n'induisent pas de comportement particulier, on les représente par des diagrammes d'activités ou simplement des actions. Les actions `augmenter` ou `diminuer` l'intensité dépendent du contexte (voir ci-après). Dans ce contexte, le sous-état `CommandesUtilisateur` contient un seul état.

Les perturbations représentent la partie réactive et programmée du contrôleur (sous-états `Perturbations` et `Lumières`). La mise à jour des intensités suite à une commande utilisateur ou une modification de la lumière du jour (événement `majAmbiant`) n'a d'effet que si l'état courant des lampes est `ambiant`. Cela peut se traduire par :

- Une précondition dans l'action associée : la modification n'est pas prise en compte si sa condition n'est pas respectée.
- Un état spécifique : dans les autre états l'événement sera-t-il ignoré, retardé ou bloqué ? Le choix est lié à la politique de gestion des événements. Avoir un état complique aussi la description dynamique car elle induit une représentation des états des plafonniers dans le système de contrôle (sous-état `Lumières`).
- Une alternative (si-alors-sinon) : l'événement est pris en compte mais le traitement n'est pas systématique.

On préfèrera la dernière solution car ces événements peuvent intervenir à tout instant et ils sont optionnels. Une autre raison est que cela simplifie le comportement dynamique.

```
context SystèmeDeContrôle::majAmbiant()
 pre: -- true
 post:
  if self.plaf1.state = Branchée::Mode::Contrôlé and
     self.plaf2.state = Branchée::Mode::Contrôlé
  then
     if self.lb.état = Allumé then llb := self.lb.luminosité
     else llb := 0 endif ; l1 := self.plaf1.luminosité ;
        l2 := self.plaf2.luminosité ; lext := le.luminosité ;
        (v1, v2) := self.calculer(l1, l2, lext, llb) ;
        self.plaf1.vali(v1) ; self.plaf2.vali(v2) ;
        self.plaf1.impulsion ; self.plaf2.impulsion
     endif
  endif
```

Dans ce contexte, les sous-états `Perturbations` et `Lumières` contiennent un seul état. Il reste à traiter le cas de l'occupation de la salle (sous-état `Mouvements`), qui consiste à piloter les capteurs directement connectés au système de contrôle.

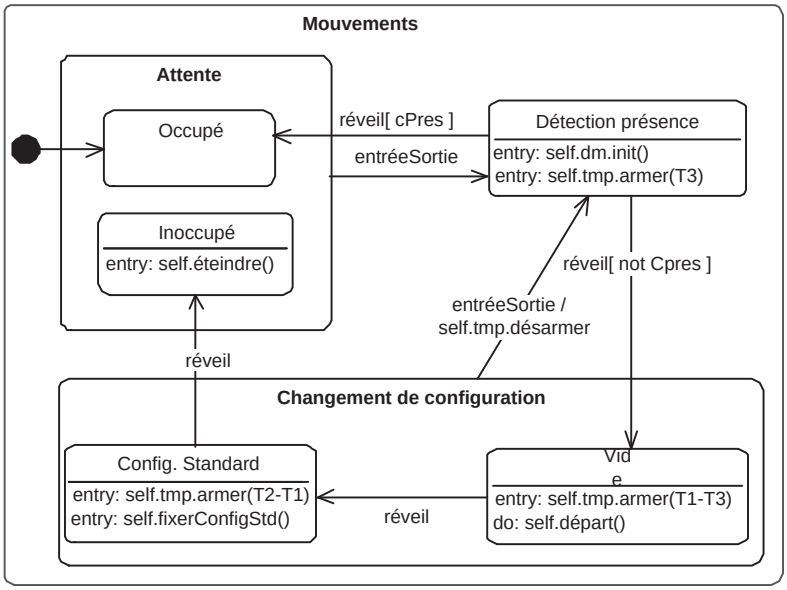

Figure 33 : *Diagramme états-transitions du contrôle état* `Mouvements` *- Lumière*

On se base essentiellement sur les diagrammes de séquence de la partie perturbation (figures 17, 20, 19) et les diagrammes états-transitions des capteurs. Les conditions sont définies comme suit :

```
context SystèmeDeContrôle::cPres() =
  (self.dm.state = EnFonctionnement::Mouvement détecté)
```

Rappelons que la gestion des temporisations peut être gérée par le système de contrôle lui-même, par exemple sous forme d'activité d'attente interruptible (`do: attendre(T3)`). La fin d'activité correspond alors à un réveil. Le diagramme précédent peut être "assaini" en séparant l'intérieur et l'extérieur de chaque état : respect du principe d'encapsulation.

3.4 Protocole de communication

Les dispositifs physique échangent avec le contrôleur directement. On supposera un système simple d'échanges de messages en circuit fermé.

La télécommande communique à distance avec le contrôleur pour commander l'allumage ou l'extinction des lumières, l'augmentation ou la diminution de l'intensité. Les capteurs échangent avec le contrôleur pour fournir les informations de l'environnement selon des protocoles spécifiques. Enfin, les actionneurs effectuent les commandes électro-mécaniques toujours selon des spécifications de communication dédiées.

A terme on peut souhaiter mettre en place un réseau de capteur et actionneur (SAN), des protocoles messages MQTT et un système de monitoring SCADA).

3.5 Synthèse, critique

La modélisation proposée ici reste une esquisse pour plusieurs raisons :

- Nous avons défini un protocole général de communication entre les composants physiques et le système de contrôle : c'est une abstraction des comportements. Ce protocole doit être validé avec les spécifications techniques des matériels utilisés. Ils sont transformés en fonction des interfaces matérielles et logicielles des composants choisis, des protocoles de communication acceptés, des normes en vigueur (phases de conception).
- La modélisation n'est pas complète : tous les scénarios et toutes les conséquences n'ont pas été visités. Les types des attributs, paramètres et opérations sont à préciser. Les contraintes, invariants et descriptions des actions sont à représenter (avec OCL par exemple). Les diagrammes d'activités des opérations complexes sont laissés à la charge du lecteur.
- La modélisation n'est pas forcément cohérente. Par exemple, les événements reçus doivent être cohérents avec les interfaces des classes (opérations). Une convention simple est d'affecter un nom d'opération à

chaque message reçu. Par exemple, les lampes reçoivent l'événement `signalInt(val)`, il est donc important d'ajouter une telle opération dans la classe `Lampe` pour assurer la cohérence pour la vérification (voir les chapitres suivants).
- Certains aspects temps-réel nécessitent une étude approfondie : temporisations, priorité d'événements, parallélisme, etc.
- La sémantique du comportement dynamique est à affiner. Nous avons mis en évidence des événements, des envois de messages, des actions et des opérations. Cependant la sémantique de ces concepts est à définir : mémorisation et durée de vie d'un événement (est-il perdu au bout d'un certain temps ?), gestion des erreurs et des conflits (un événement inattendu provoque-t-il une erreur ou est-il retardé ? Une garde fausse génère-t-elle une exception ?), lien entre événements, messages et actions (voir en particulier les exercices du TD relatifs à la cohérence entre diagramms UML). La sémantique définitive est celle du noyau temps-réel cible, EV3 en l'occurrence pour le prototypage. Il est bon d'en faire un modèle qu'on met en adéquation avec les modèles d'analyse (cf dossier d'analyse technique demandé dans ce projet).
- Une validation doit être effectuée par le demandeur et/ou des experts, qui attestent de la bonne interprétation que nous avons faite de l'énoncé.

Malgré tout, le modèle proposé permet de mieux comprendre le système et d'en faire une simulation, non seulement logicielle (le *twin digital*) mais aussi matérielle avec le prototype Lego.

4 Exigences, contraintes et hypothèses

Nous précisons ici quelques exigences du système.

4.1 Hypothèses et restrictions

On ne traite que de la partie logicielle. On ne tient pas compte du fonctionnement en mode manuel, lorsque le contrôleur est déconnecté.
- Pour le prototypage, on représentera le système d'une manière propre aux dispositifs disponibles en Lego Mindstorm EV3 : l'intensité de la lumière sera représentée par un degré de rotation sur un bras armé par le grand moteur.
- Les signaux sonores sont déclenchés par le boitier EV3 lui-même.
- Le capteur de luminosité sera mis en œuvre par le détecteur de couleur : clair pour le jour et sombre pour la nuit.

4.2 Exigences fonctionnelles

Le système doit permettre de manipuler la lumière via son contrôleur. C'est un système distribué sur plusieurs dispositifs physique, la phase de mise en route et d'initialisation n'est donc pas triviale, de même que la phase d'arrêt.

Les exigences fonctionnelles sont classées par priorité.

1. Gestion de configuration : lancement, arrêt des différents dispositifs.
   (a) Initialiser le mode automatique, paramétrer
      i. Vérifier l'état des dispositifs physique (capteurs et actionneurs), tableau de commande, supervision.
      ii. Initialiser chaque dispositif.
      iii. Initialiser le mode automatique du contrôleur.
      iv. Vérifier la cohérence globale (assertions).
   (b) Basculer en mode Manuel
2. Gestion des panneaux de commande (écran PC ou mobile)
   (a) Enregistrer un panneau
   (b) Supprimer un panneau
3. Gestion des interrupteurs (via les panneaux)
   (a) Allumer / Éteindre

   (b) Augmenter / diminuer l'intensité
4. Monitoring
   (a) Afficher l'état du système :
      i. Etat du contrôleur,
      ii. Etat des dispositifs matériels
   (b) Gérer les événements
      i. luminosité extérieure,
      ii. détection de présence (allumer si mouvement, éteindre si pas de mouvements depuis 1 minute)
   (c) Gestion d'un historique (Log)
      i. Liste des commandes et des états du contrôleur
      ii. Liste des anomalies et changements de modes (manuel/automatique)
      iii. Journalisation des événements et états pour tracer la source d'erreurs
5. Gestion des incidents
   (a) Passer en mode Manuel
   (b) Revenir en mode Automatique
   (c) Prise en compte des interruptions de courant et des opérations de reprise.
   (d) Prise en compte des communications avec les dispositifs matériels et des opérations de reprise.
   (e) Suivi des pannes (journalisation)
6. Divers.

4.2.1 Communication et prototypage

En pratique, la comunication avec LEGO MINDSTORMS EV3 se fait en bluetooth ou Wifi pour la commande et en liaison filaire pour le reste.

En extension, on définira un protocole sécurisé vérifiant l'authenticité de la télécommande pour ces échanges.

4.3 Extra-Fonctionnelle

Le système doit permettre de manipuler la lumière en toute sécurité.

- L'intensité doit rester dans les normes des ampoules utilisées. En cas de présence et de luminosité extérieure faible, un niveau minimal d"éclairage est attendu.
- L'état des dispositifs est cohérent avec celui du contrôleur (cohérence des capteurs).
  - Si la lumière est allumée, le contrôleur la perçoit comme tel.
  - Si une présence est détectée, le contrôleur est actif.
  - Si la commande d'intensité augmente, l'intensité constatée est plus forte.
  - Si la commande d'intensité diminue, l'intensité constatée est plus faible. ...
- En cas de dysfonctionnement d'un dispositif, le contrôleur se met automatiquement en mode manuel, par exemple si un capteur n'est plus actif (ne répond plus).

4.4 Exigences techniques

La mise en œuvre du prototype se fera avec les outils Java pour Lego Mindstorm, la plate forme **Lejos**.

4.5 Evolutions à prévoir

On peut imaginer des facilités supplémentaires paramétrables.

- Plusieurs éclairages à synchroniser avec plusieurs capteurs de luminosité, selon la taille de la pièce.
- L'éclairage de la salle "inoccupée" diminue au bout de 1 minute et s'éteint au bout 3 minutes.
- Plusieurs télécommandes (murale, portable, internet...).