Le rapport de Micro-projet Smart Institute
Modélisation pour un bâtiment intelligent

Xin ZHANG

Ce modèle est conçu avec la méthode de développement itérative et incrémentale indiquée et ce rapport a suivi ce processus de modélisation. Certaines erreurs et incompréhensions causées par mon manque d'expérience en développement avec UML et Design Patterns peuvent exister, et je voudrais connaître vos instructions et corrections généreuses.

I – 1re itération

A.1 Use Case général

Nous savons qu'il existe au total cinq types d'acteurs (Ils sont tous des occupants), enseignants, étudiants, personnel administratif, personnel de maintenance et visiteurs. Et aussi cinq opérations qui leur sont liées, donc la réglage simple, l'interaction via les bornes, l'enregistrement du profil, la configuration des règles et la manœuvre d'ouverture et de fermeture du système. Toutes les opérations marchent après l'identification, et leur accès aux occupants varient selon leurs catégories, donc occupants permanents, administrateur ou visiteur. Alors je le dessinais comme ci-dessous :

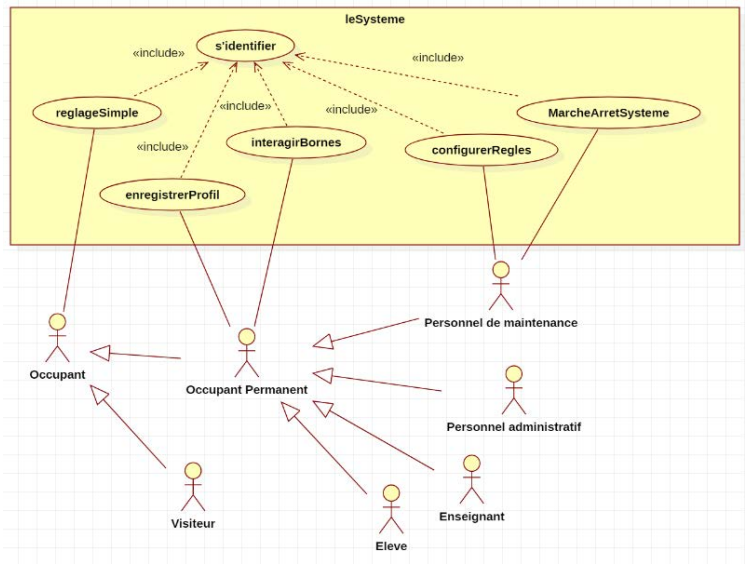

Figure 1 Use Case général

A.2 Le diagramme de classes initiale

Avant de discuter du diagramme de classes, je dois avouer que je ne suis pas certain d'ajouter des acteurs à un diagramme de classes. Certains utilisateurs de Stack Overflow ont indiqué que nous ne pouvions pas les ajouter, mais il y a également une personne sur YouTube en expliquant comment ajouter des acteurs aux diagrammes de classes dans un logiciel UML autre que StarUML, à la fin j'ai choisi de croire nos TPs et Stack Overflow.

En bref, les principaux objets d'un bâtiment sont les salles, et dans chaque salle, il y a une borne et plusieurs actionneurs et capteurs. De plus, les communications entre eux et le contrôleur central sont par la borne, alors je dessinais la diagramme de classes comme ci-dessous :

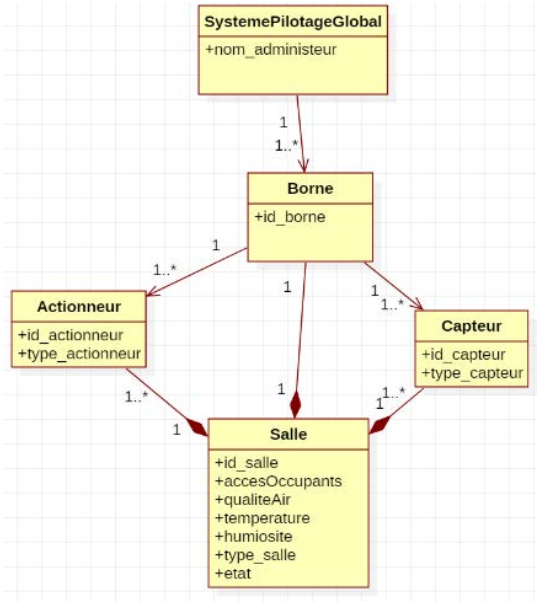

Figure 2 le diagramme de classes initiale

A.3 Les DSS initiales

Afin de trouver les principaux messages, les diagrammes de séquences système (DSS) correspondant les cinq opérations sont montrés ci-dessous, et ils seront ensuite détaillés.

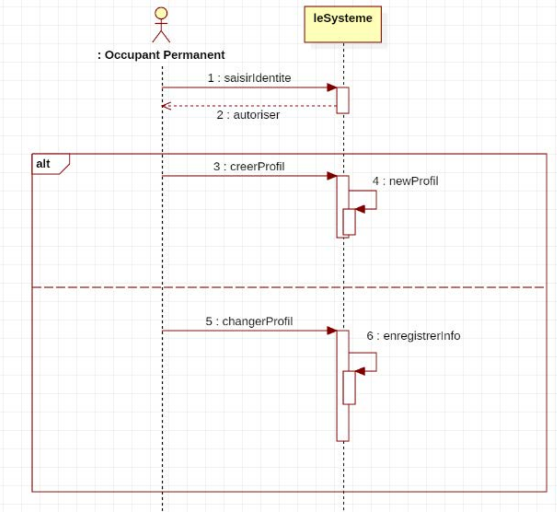

Figure 3 l'enregistrement du profil initial

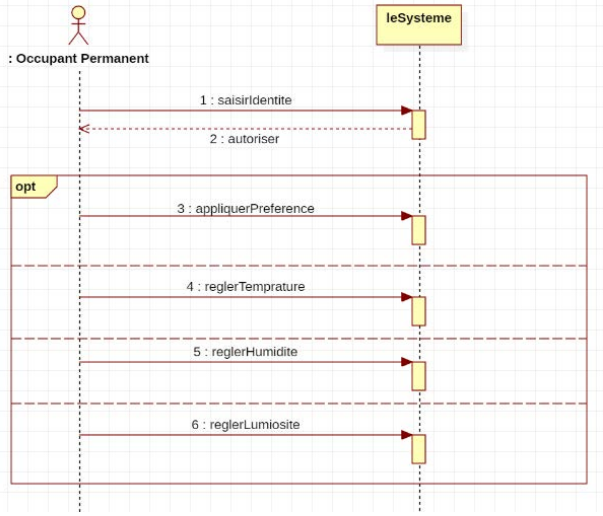

Figure 4 l'interaction via la borne

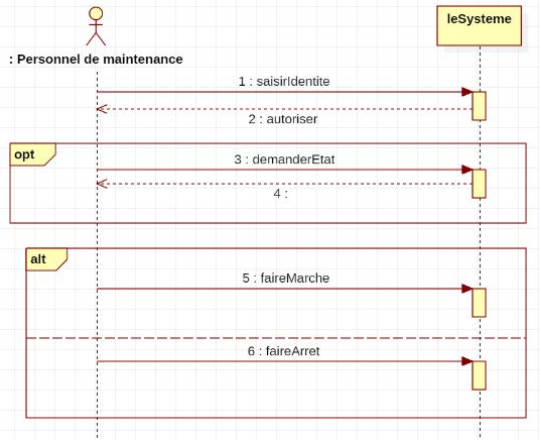

Figure 4 la manœuvre du système

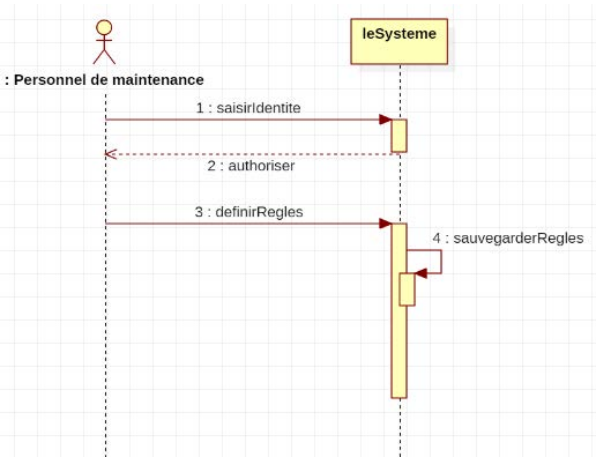

Figure 5 la configuration des règles

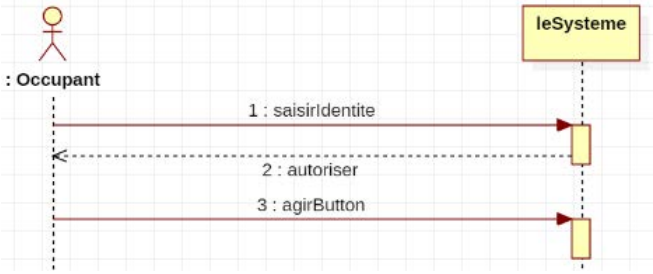

Figure 6 le réglage simple

A.4 Le diagramme de classes d'analyse initial

Comme nous en avons discuté, la communication entre les acteurs et le système se fait par la borne, alors je l'ai mis comme *boundary*, et évidemment *control* pour le système de pilotage, pour les autres classes, provisoirement, je les ai mis comme *entity*, comme indiqué ci-dessous.

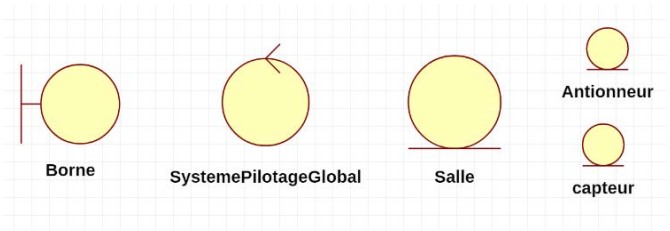

Figure 7 le diagramme d'analyse initial

II - 2e itération

B.1 DP pour la catégorie *construction* – Pattern *Singleton*

Tout d'abord, il y a seulement un système de pilotage global, sinon, ça va être impossible de configurer le système par un seul administrateur, la classe de ce système ne peut pas être créé par les autre classes, sa méthode de création dois rester *private, static* et *final*, et toutes les méthodes définis sous cette classe doivent s'assurer que seule une classe de **SystemePilotageGlobal** est créée.

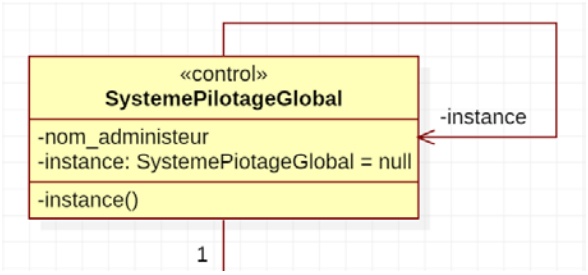

Figure 8 DP Singleton Cas Système de pilotage global

B.2 DP pour la catégorie *structuration*

Pattern *Proxy*

Dans le but de respecter des normes de sécurité incendie : la détection de fumée dans une pièce doit déclencher une alarme dans toutes les salles, une communication entre tous les capteurs est indispensable, mais ce n'est pas flexible de construire les connexions entre tous les capteurs de fumée car un capteur dans une salle peut être remplacé, des nouvelles salles peuvent être ajoutés et un institut a normalement un très grand nombre de salles. Donc on a besoin de réaliser un proxy entre les capteurs et le système de contrôle.

Et pour ce modèle ci, j'utilise le Borne comme le proxy. Lorsque la borne détecte que l'état du capteur de fumée est "oui", elle transmettra ce message au contrôleur et celui-ci notifiera à tous les bornes qui lui sont connectés de faire sonner tous les capteurs de fumée.

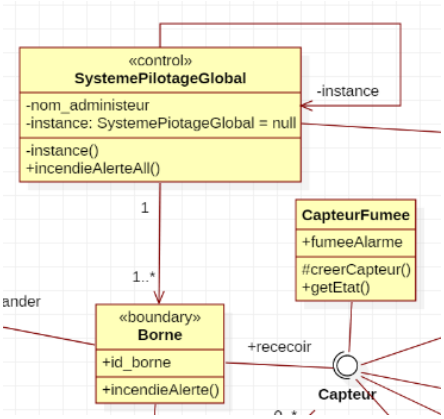

Figure 9 DP Proxy Cas sécurité Incendie

Pattern *Decorator*

Nous avons déjà vu qu'il faut y avoir plusieurs types de terminaux pour occupants différentes, et leurs capacités de contrôle sur cette borne doivent être modifiable, alors j'ai choisi le pattern *Decorator* pour modéliser ce cas, alors une alternative flexible au sous-classement pour étendre les fonctionnalités est fournie.

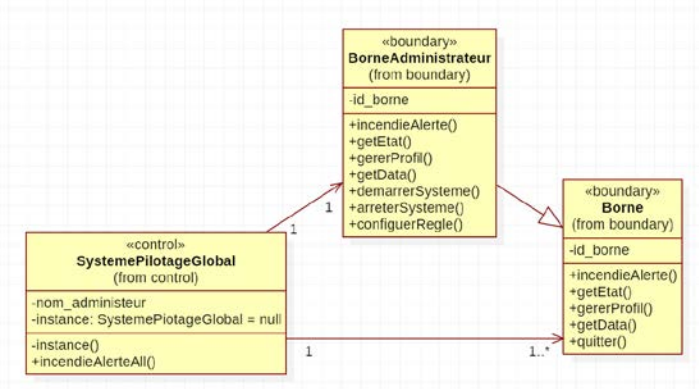

Figure 10 DP Decorator Cas Borne

B.3 DP pour la catégorie *comportement* – Pattern *Strategy*

On a un algorithme spécifique pour chaque salle et chaque type de travail, alors on dois appliquer un pattern *Strategy* afin que le contrôle peut effectuer une validation sur les données entrantes peut utiliser le modèle de stratégie pour sélectionner un algorithme de validation en fonction du type de données, de la source des données, du choix de l'utilisateur ou d'autres facteurs discriminants. Et stratégie spéciale pour la situation comme une panne de courant ou sécurité incendie peuvent être ajouté facilement en utilisant ce type de design pattern.

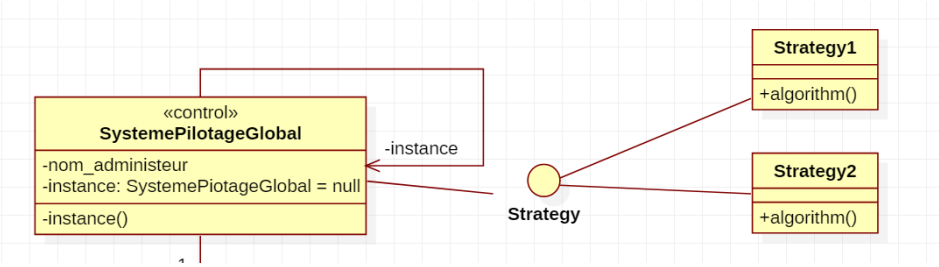

Figure 10 DP Strategy Cas Algorithme

Et après l'application des Design Patterns, les scénarios d'utilisation sont écrits comme :

Cas Réglage Simple	Cas Configurer Règles
Acteur(s) : Occupant (Tous les acteurs avec badge) Description : L'occupant agit sur un bouton après l'indentification Pré-conditions : L'utilisateur doit être authentifié en tant qu'occupant Démarrage : L'utilisateur entre la chambre et active la borne. *Les scénarios nominaux :* 1. L'utilisateur entre la salle et démarre la borne. 2. l'IHM de la borne s'affiche. 3. L'utilisateur présente son badge. 4. L'utilisateur agit sur le bouton de réglage simple. *Les scénarios alternatifs :* 2.a L'utilisateur décide de quitter la borne. 3.a L'utilisateur décide de quitter la borne. *Les scénarios erronés :* 2.e L'écran de la borne est en panne. 3.e Le badge de l'utilisateur est périmé. 4.e Le bouton ne fonctionne plus.	Acteur(s) : Occupant (Tous les acteurs avec badge) Description : L'utilisateur configure les règles du système de pilotage global sur son IMH. Pré-conditions : L'utilisateur doit être authentifié en tant que personnel de maintenance. Démarrage : L'utilisateur entre la chambre et active la borne. *Les scénarios nominaux :* 1. L'utilisateur entre la salle et démarre la borne. 2. l'IHM de la borne s'affiche. 3. L'utilisateur présente son badge. 4. L'utilisateur configure les règles du système de pilotage global. *Les scénarios alternatifs :* 2.a L'utilisateur décide de quitter la borne. 3.a L'utilisateur décide de quitter la borne. *Les scénarios erronés :* 2.e L'écran de la borne est en panne. 3.e Le système a été piraté.

Table 1 Example des scénarios d'utilisation

Par ailleurs, les diagrammes associés sont dessinés dessous :

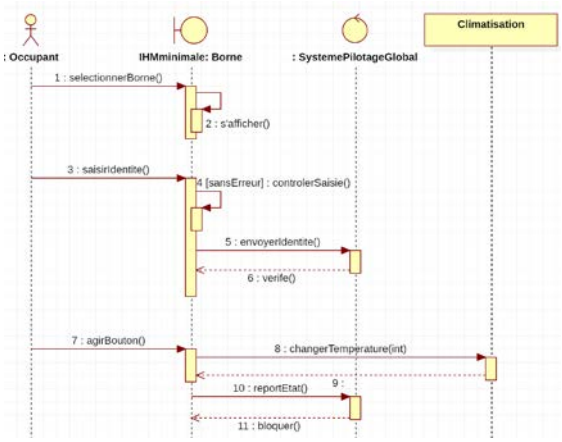

Figure 10 Cas réglage simple raffiné

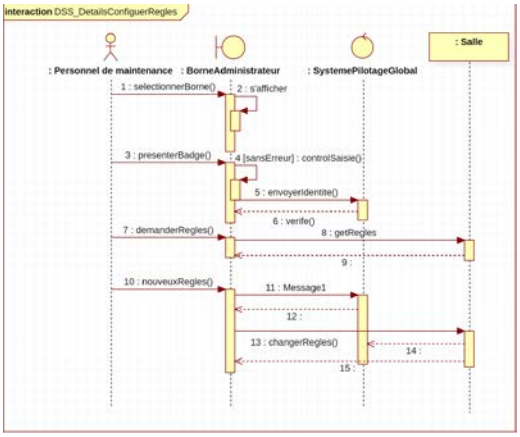

Figure 11 Cas

III 3e itération

C.1 entre Vue et Modèle

Le Design Pattern choisi pour cette caractéristique est *Observer*, dans ce cas, le contrôleur observe l'état de la salle par la borne dans cette salle en utilisant la méthode getEtatSalle(), en cas de changement d'état inattendu, alias non contrôlé par le contrôleur, ce contrôleur avertira l'administrateur en envoyant un message à la borne de la personnel de maintenance.

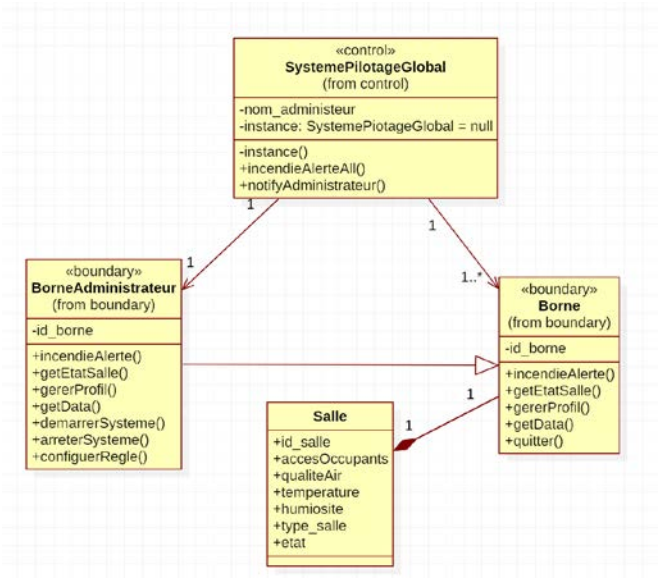

Figure 13 entre Vue et Modèle

C.2 entre Contrôleur et Modèle

Le Design Pattern choisi pour cette caractéristique est *State*, le pattern *State* est interprété comme un pattern Strategy capable de basculer la stratégie actuelle via des appels de méthodes définies dans l'interface du modèle. Cela signifie que, après avoir appuyé sur le bouton, la salle se changera en état dans lequel ce bouton ne pourra plus modifier la température.

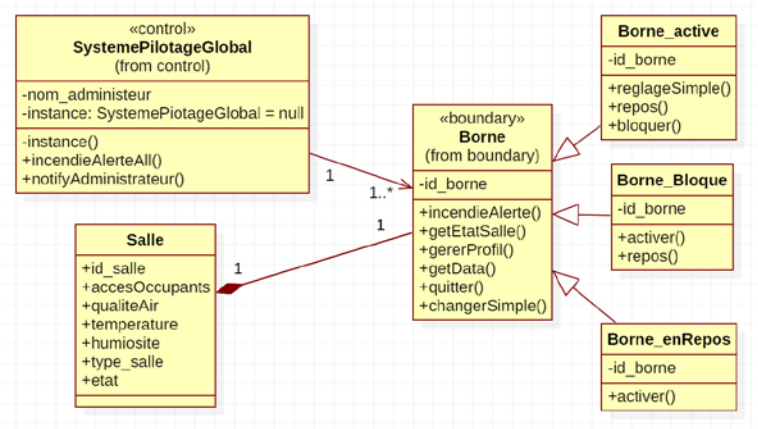

Figure 14 entre Contrôleur et Modèle

C.3 entre Vue et Contrôleur

Le Design Pattern choisi pour cette caractéristique est *Facade*, une facade améliore la lisibilité et la convivialité de ce système en masquant l'interaction avec toutes ces salles derrière une API unique et simplifiée fournie par le contrôleur.

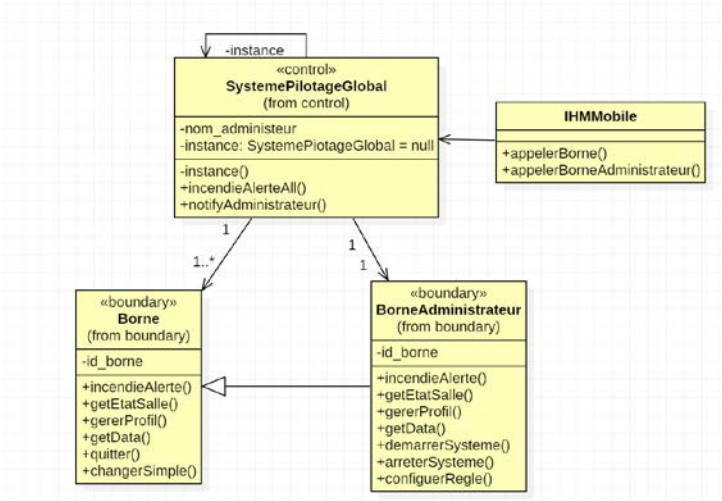

Figure 15 entre Vue et Contrôleur

IV Conclusion

Avant de venir en France cet été, j'avais une expérience de développer un petit système de contrôle de domicile via Bluetooth dans mon université en Chine. Cependant, à l'époque, j'avais réalisé ce système fonctionnalité à fonctionnalité. De nouvelles fonctionnalités n'ont été prises en compte que lorsque toutes les anciennes fonctionnalités fonctionnaient bien car je n'ai pas appris la modélisation avec UML Franchement, en faisant ce projet, je me suis senti un peu dépaysé, mais j'ai la confiance qu'avec les cours suivants de la filière SLR, je vais beaucoup progresser et Je vais m'habituer à penser dans la grande image.