Rapport UV5.1

Ascenseur

Tao ZHENG, Xinrui GUO, Sisi REN

Option – SLS ENSTA Bretagne

29/02/2016

Table des matières

1. Introduction

L'objectif est de modéliser d'un schématique UML, et puis le transformer vers Fiacre. Ensuite, le programme cible Fiacre sera utilisé pour vérifier formellement (model-checking) des propriétés avec l'outil OBP Explorer. Pour cela, les propriétés du système, encodées par des prédicats et des automates d'observateurs sont décrits avec le langage CDL.

2. Modélisation avec UML

Le processus de réalisation est comme la figure au-dessus :

Use Case -> Diag Sequence Static -> Diag Class -> Diag Object -> StateChart -> Diag Sequence dynamic

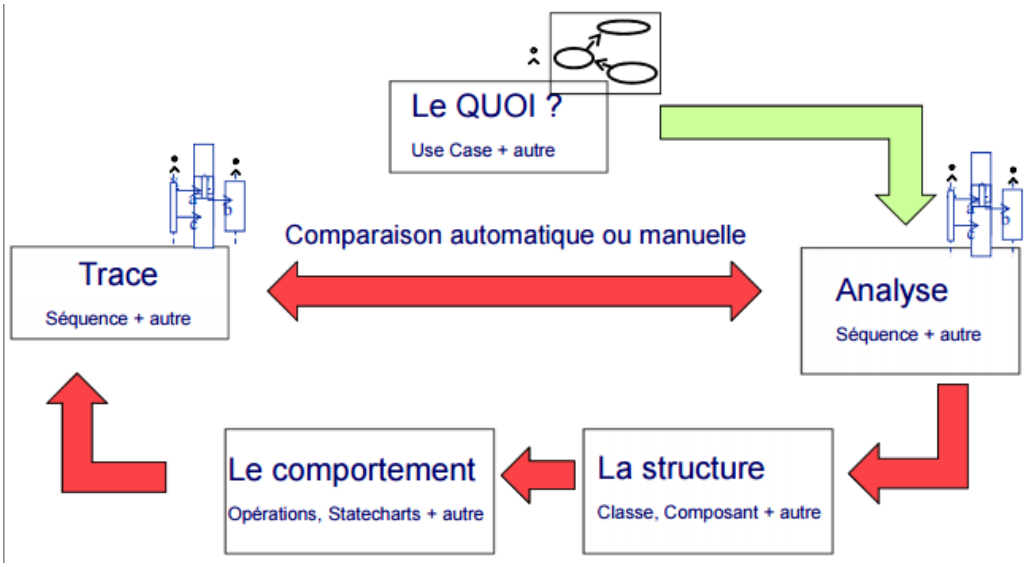


Figure 2-3 Modélisation avec UML

2.1 Diagramme de use case

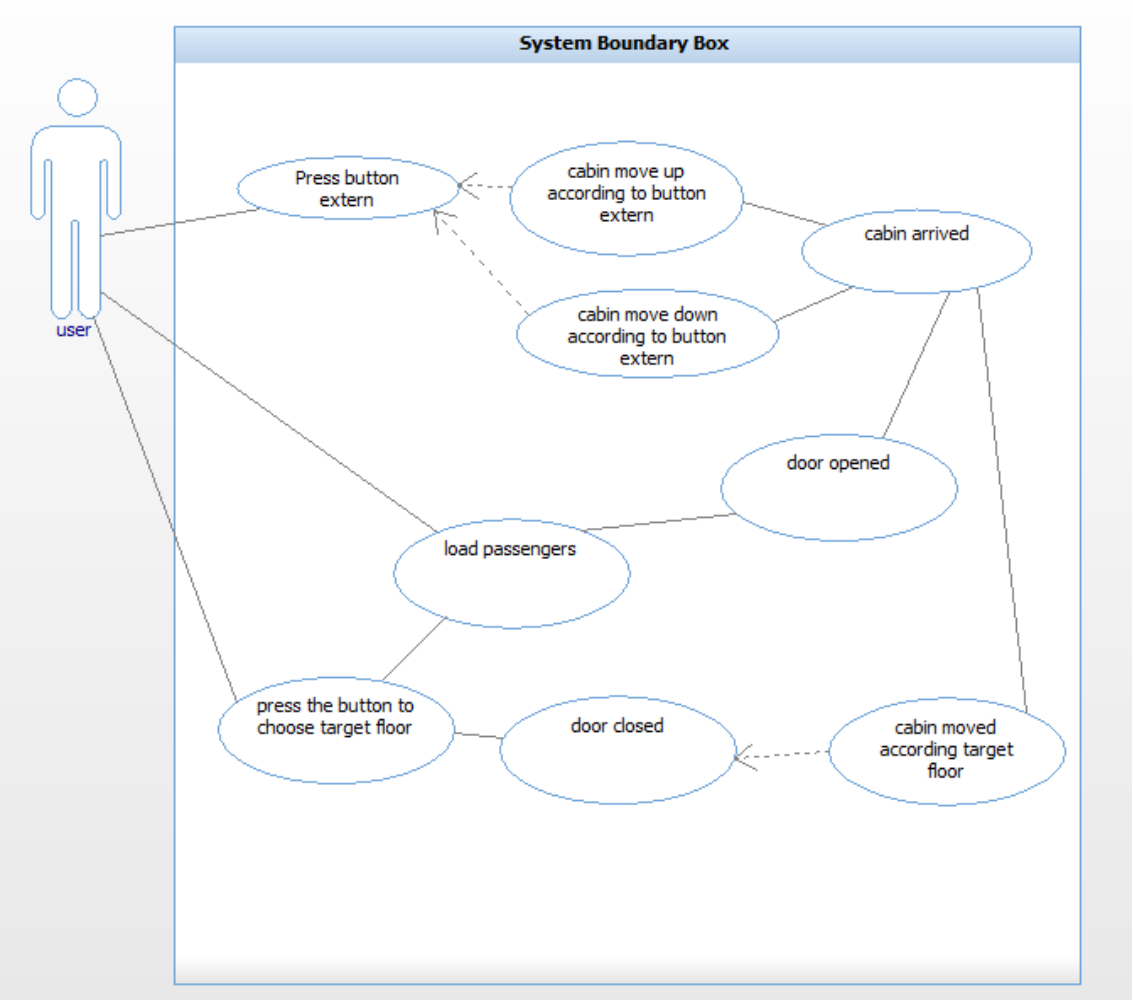


Figure 2.1-1 use case

Le diagramme d'Use Case définit une manière d'utiliser le système et permet d'en décrire les exigences fonctionnelles. On définit un utilisateur qui interagit avec le system d'ascenseur. Tout d'abord, l'utilisateur appuie le bouton externe de l'ascenseur. Et puis, la cage va monter ou descendre par treuil. Quand la cage arrive, la porte s'ouvert, l'utilisateur entre dans la cage et choisit le numéro d'étage. Après la porte se ferme, la cage va bouger jusqu'à l'étage cible, la porte s'ouvert, l'utilisateur sortie.

2.2 Diagramme de classes

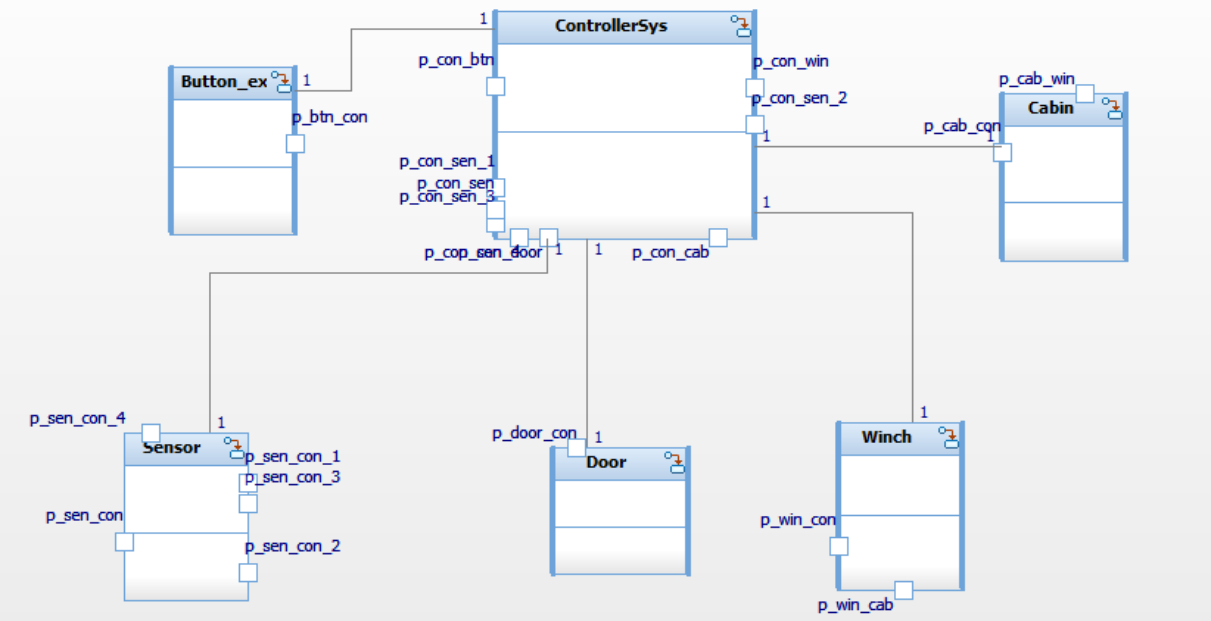


Figure 2.2-1 diagramme de class

Le diagramme de classe présent les relations entre les classes. Dans ce cas-là, ce diagramme contient six classes, Button_ex, Contoller, Winch, Cabin, Door et Sensor. La classe Controller est un centre contrôleur qu'il relie avec tous les autres, il gère tous les commandes et donne l'ordre aux autres classes.

2.3 Diagramme de scénario

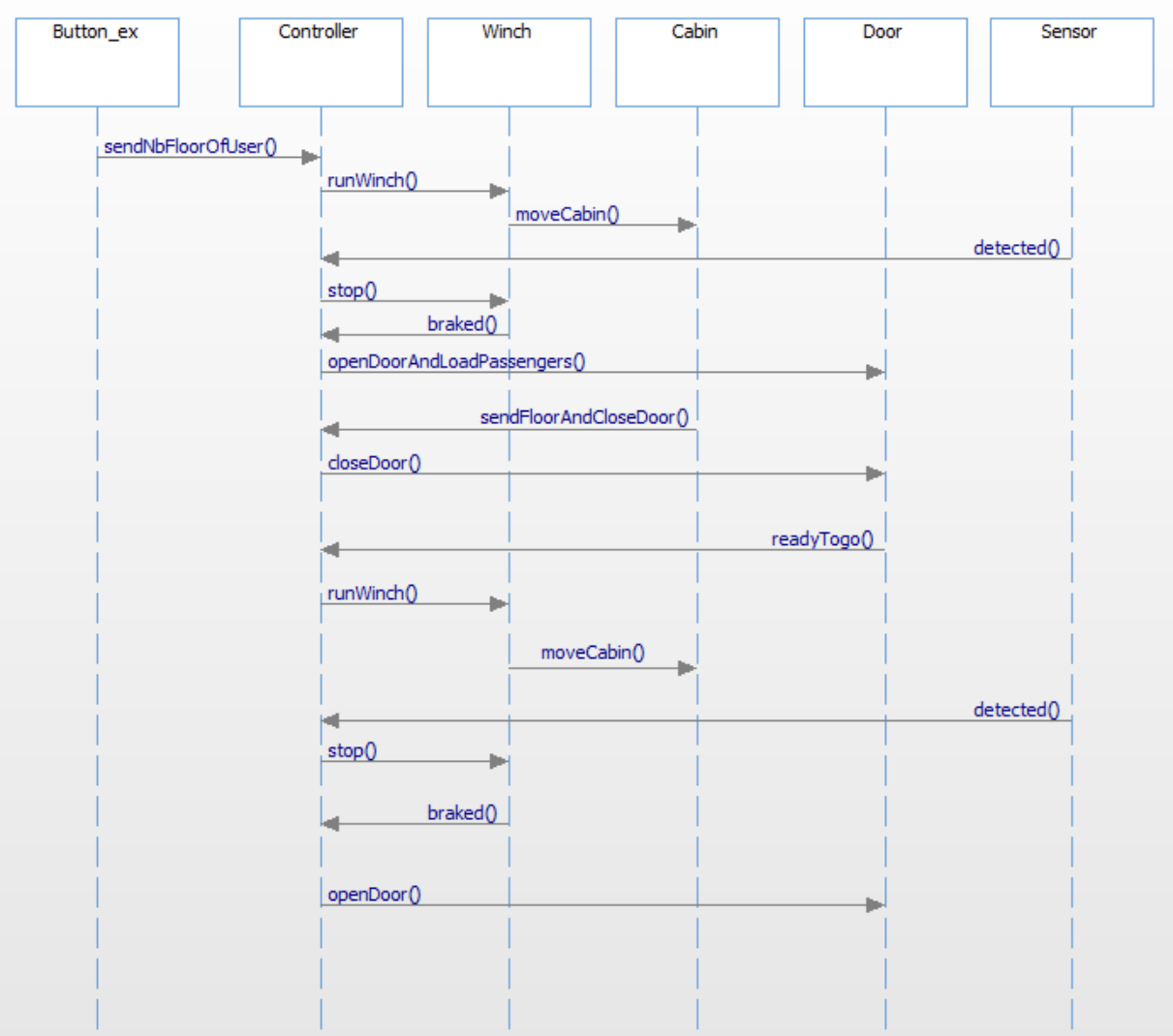


Figure 2.2-1 diagramme de scenario

Notre system d'ascenseur comporte Button_ex, Contoller, Winch, Cabin, Door et Sensor. Quand l'utilisateur appuie le bouton externe, le Button_ex va envoyer le numéro d'étage de l'utilisateur au Controller. Et le Controller juge la direction de mouvement de cage, et puis il actionne la cage par le treuil, quand le capteur de l'étage cible détecte la cage, le treuil va freiner la cage, et la porte va ouvrir. L'utilisateur entre et appuie l'étage cible, la porte ferme, et le treuil actionne la cage, quand la cage arrive, la porte ouvert, et l'utilisateur sort.

2.4 Diagramme d'objet

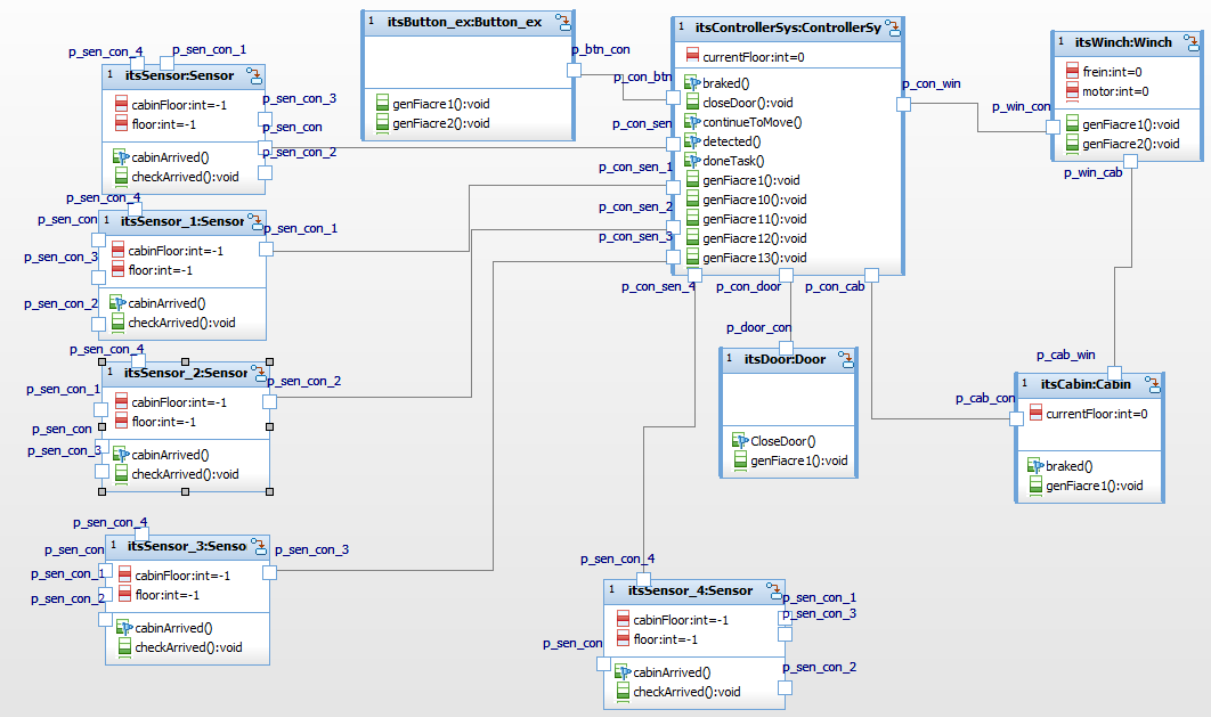


Figure2.4-1 diagramme objet

Le diagramme d'objet présent les relations entre les objets. Il est l'instanciation de diagramme de class. Les objets sont reliés pas les ports et la relation Link. Et, on instance cinq capteurs, car il y a cinq étages, et chaque étage possède un capteur pour détecter la cage.

2.5 Diagramme d'états principaux

1) Automate de ControllerSys

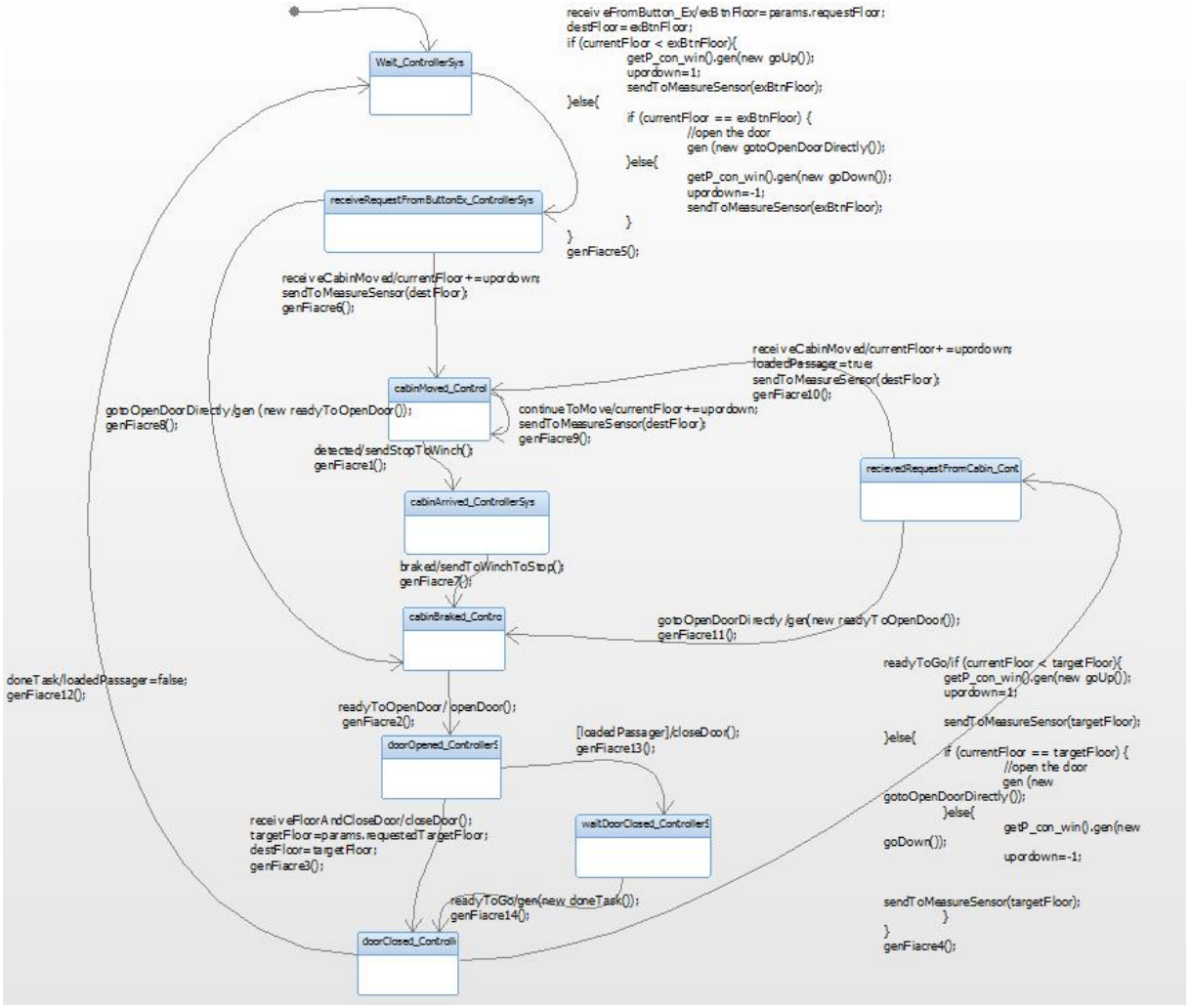


Figure 2.5-1 automate de Controller

La classe Controller est un centre contrôleur qu'il gère tous les commandes et donne l'ordre aux autres classes. Au début, il reçoit la commande de l'automate Button_ex, il a l'étage de la cage initial et l'étage de l'utilisateur, ainsi il peut juger la direction de mouvement de cage. Et puis, il donne ordre au treuil pour actionner la cage, et aussi active les capteurs de chaque étage pour détecter la position de cage. Ensuite, quand la cage arrive, il commande la porte ouvrir ou fermer. Quand l'utilisateur choisit l'étagé, il va refaire le cycle de mouvement de cage.

2) Automate de Cabin

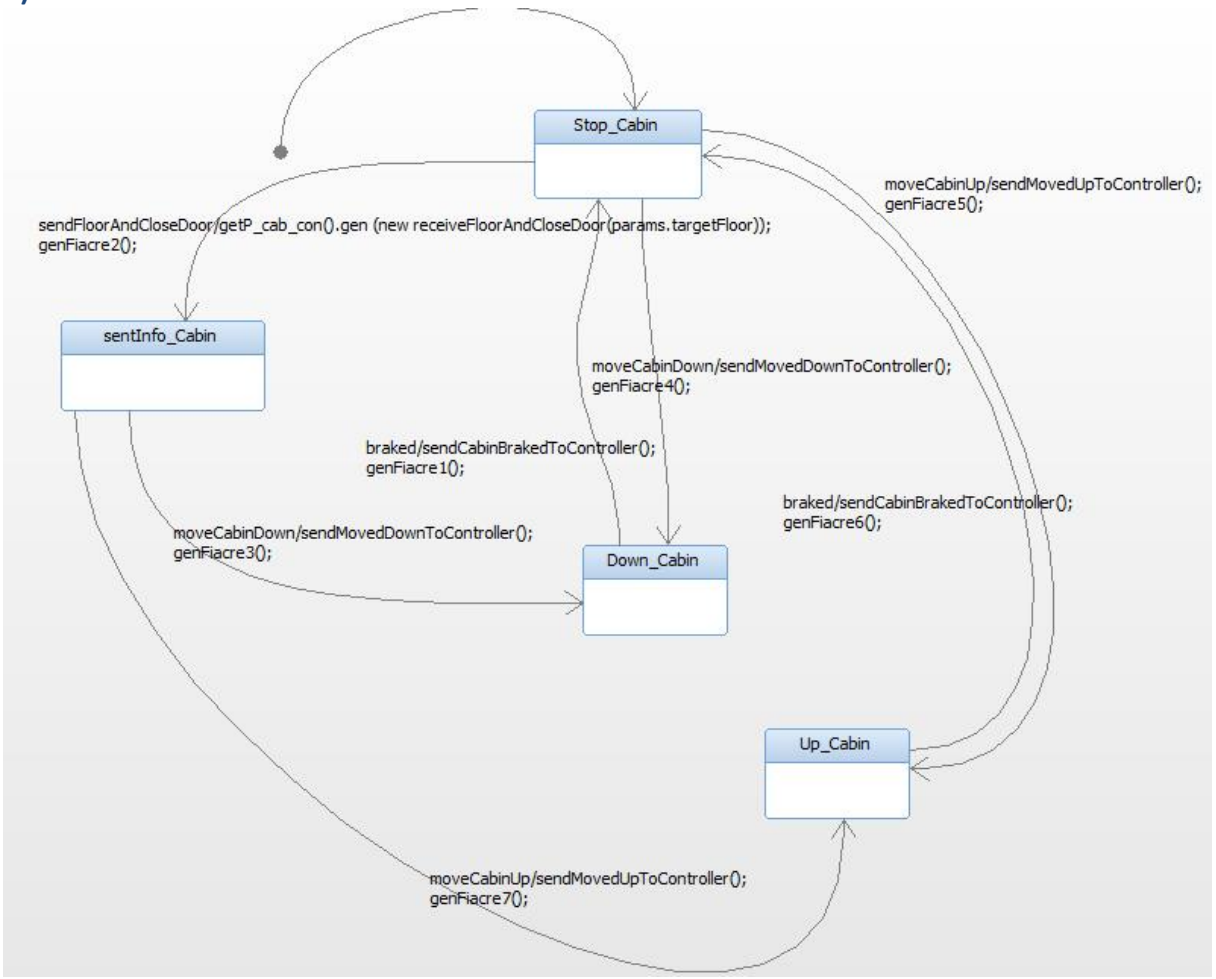


Figure 2.5-2 automate de cage

Dans cet automate, il contient les quatre états. Trois états de mouvement, frein, descente, s'élevé, et un état pour donner le numéro de l'étage cible au centre contrôleur.

3) Automate de Sensor

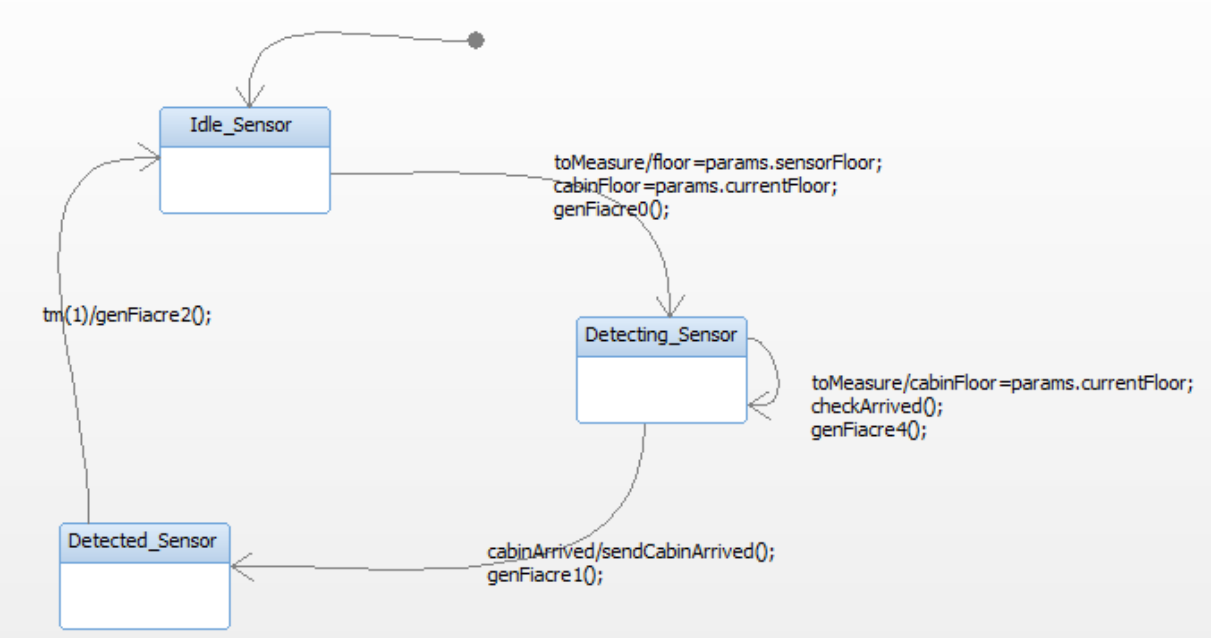


Figure 2.5-3 automate de capteur

Dans l'automate de capteur, il détecte la position de cage. Il reçoit le numéro de l'étage cible par centre contrôleur, quand ce numéro égale au numéro de capteur, il va renvoie cabinArrived à l'unité centre.

3. Transformation de UML à Fiacre

3.1 Meta modèle de Fiacre

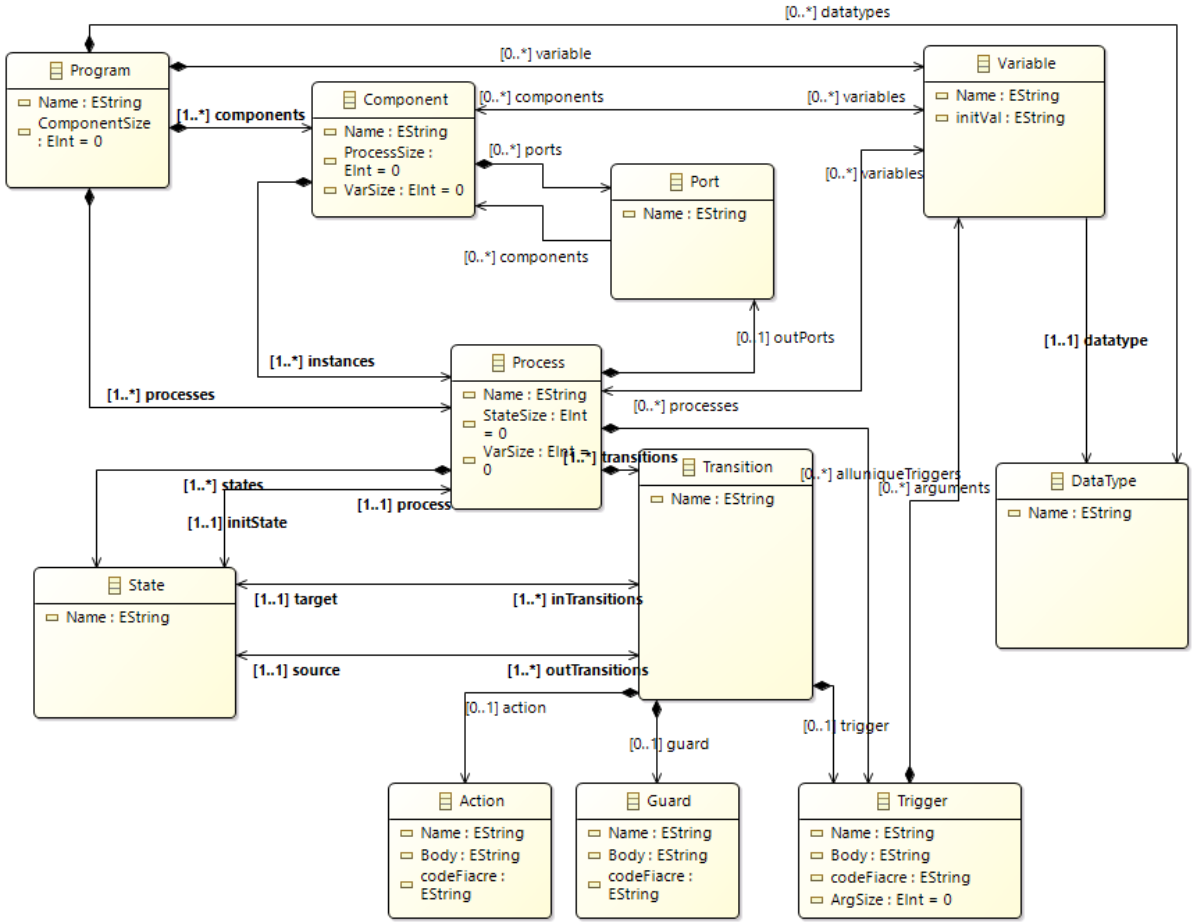


Figure 3.1-1 méta modèle Fiacre

Dans le Meta modèle de Fiacre, on définit son structure. Dans un programme de Fiacre, il contient plusieurs de Component, Processus, Variables. Chaque Component contient les ports, variables, les instances objet. Et pour chaque processus qu'il comporte les états, transitions, variables. Chaque transition contient une opération, un guard, un trigger. Pour l'état de source, il y a une transition de sortie, et pour l'état cible, il y a une transition d'entrée. Et, chaque variable peut choisir les différents types.

Des relations importantes entre UML et Fiacre

UML	Fiacre
ObjectDiagram	component
Statechart	Process
All unique trigger	Port
State	State

3.2 Traduction de UML

En fait, avec le Meta modèle Fiacre et les templates, nous avons obtenu un code Fiacre bien structuré, par contre il y a toujours des petits problèmes, donc le code Fiacre qui est géré automatiquement ne fonctionne pas correctement. Car nous n'avons beaucoup de temps à faire, nous avons décidé de le modifié manuellement à partir du même structure.

Nous avons noté quelque différence entre le langage UML et le langage Fiacre :

- Les transitions en UML sont effectués par les triggers, en Fiacre, nous les représentons avec la synchronisation de port, donc un problème est dans UML, nous pouvons ajouter plusieurs synchronisations dans l'action en même état, par contre, en Fiacre, il y a que une synchronisation en un état, donc cette différence pose des problèmes sur les états. Pour résoudre ce problème, nous avons besoin d'ajouter des états à la main.

- Comme nous avons considérons que les triggers en UML sont les ports en Fiacre, alors les triggers en sorites qui sont défini dans les actions et qui ne sont pas entré dans le meta modèle Fiacre, nous avons besoin d'ajouter à la main.

- Les codes Java dans les actions sont reçu comme les String dans le meta modèle, donc c'est aussi difficile à traiter, nous le traitons à la main aussi.

4. Validation des règles de transformation

En fait, les évènements d'utilisateurs sont créé comme un évènement venant de l'environnement dans UML, par contre, en Fiacre, nous avons créé un processus utilisateur qui permet de cliquer les buttons extérieurs et cliquer les buttons sur cage automatiquement. Une fois les utilisateurs sont entré dans la cage, le processus doit cliquer les buttons dans la cage ; le processus utilisateur cliquer les buttons extérieur au tout début.

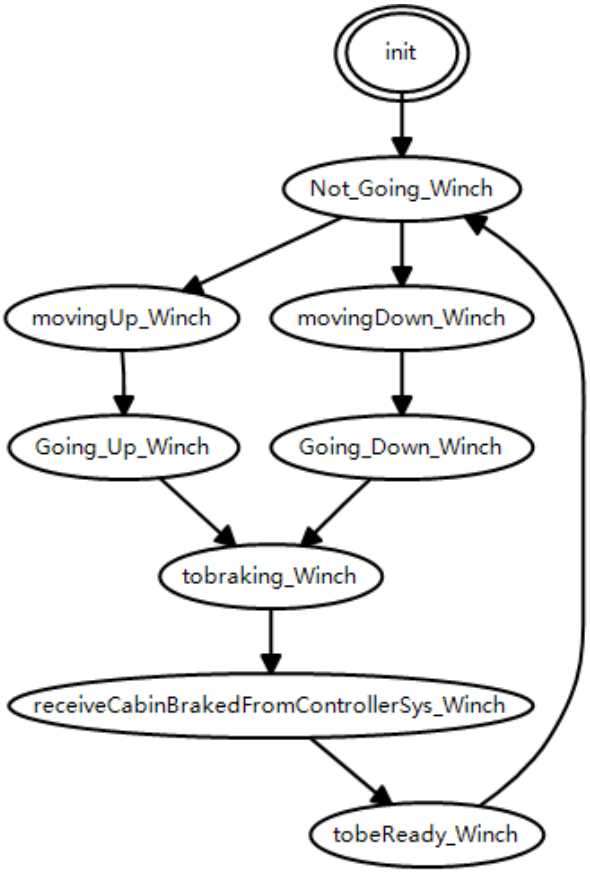
Après une modification manuellement, nous avons obtenu un code Fiacre qui fonctionne correctement.

Des propriétés à vérifier sous CDL:

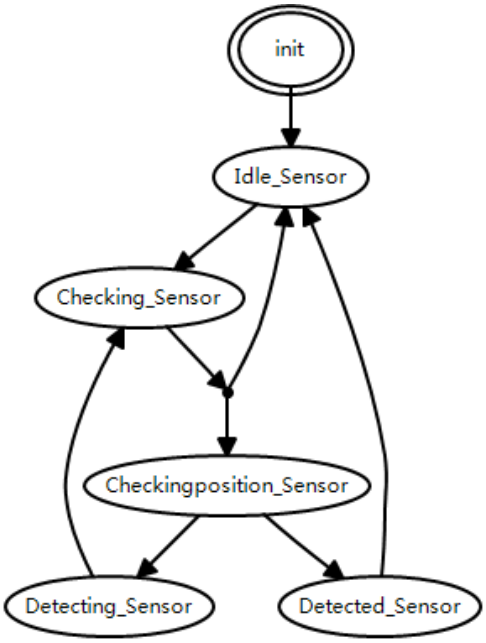
- Les transitions d'états soient accessibles et atteignable

- Une fois un utilisateur clicker sur un Button extérieur, le capteur assoicé avec cet étage doit measurer la position de cage. Une foit ce capteur a détecté la cage, il doit informer le central controlleur, le centrale controlleur doit gérer des evenements tels que informer le treuil et ouvrir les ports etc.

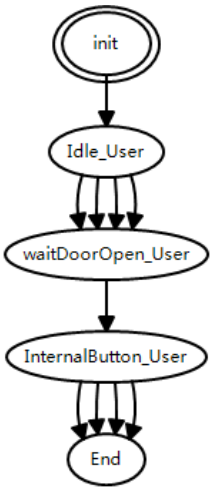
Après une exécution sous obp/cdl, nous avons obtenu tous les diagrammes d'états par processus comme ci-dessous :



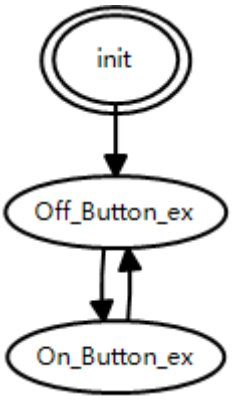
Winch



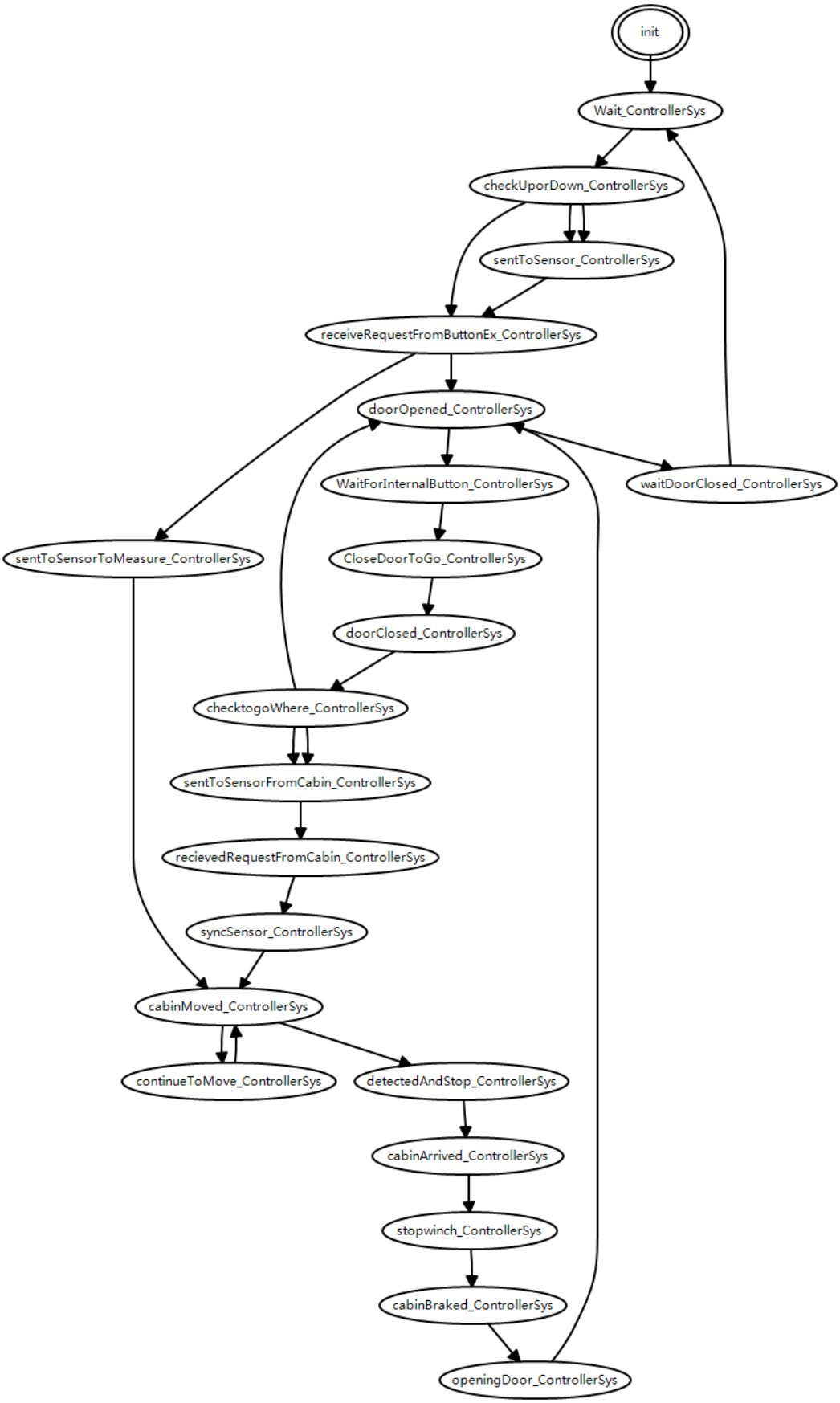
Sensor



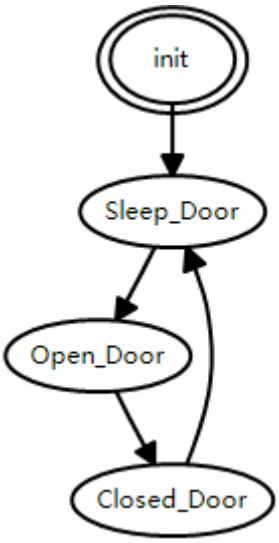
User



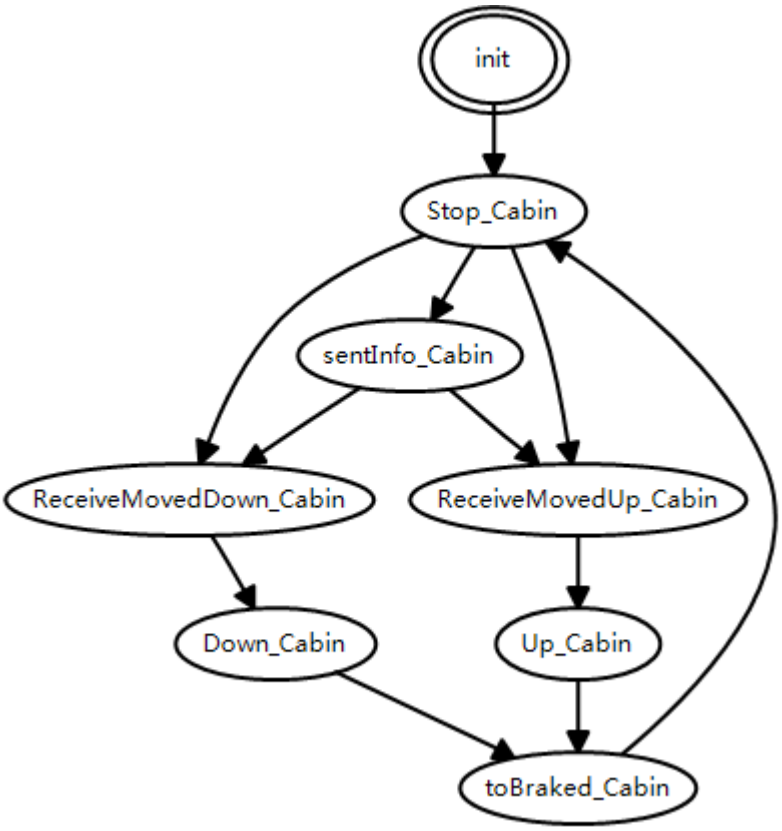
Button



ControllerSys

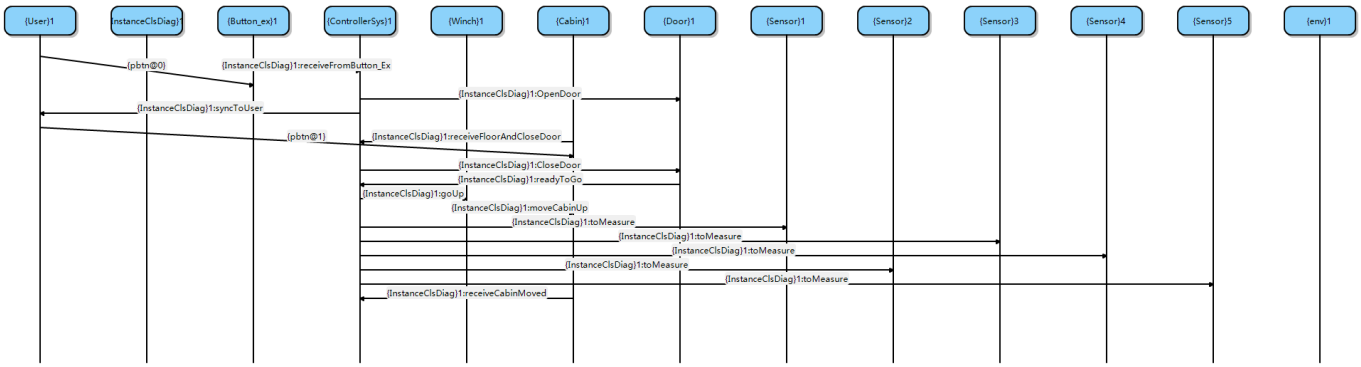


Door



Cabin

Avec l'exécution sous obp/cdl, il y a 1229 états et 4356 transitions, il est très grand. Donc avec une trace 1099, nous avons obtenu le diagramme de séquence.



Conclusion

Même si nous avons des problèmes sur la transformation d'UML à Fiacre, mais à partir de code généré avec une petite correction à la main, nous avons obtenu un code Fiacre qui fonctionne. Avec l'outil obp/cdl, nous avons bien vérifier le résultat et nous avons comparé le trace obtenu par obp avec le trace obtenu sous Rhapsody , le résultat a été correctement et cohérant.