Rapport UV5.6

Modélisation de Sécurité Pattern

Tao ZHENG, Xinrui GUO, Sisi REN

Option – SLS ENSTA Bretagne

27/02/2016

Table des matières

1. Introduction

Dans le cas général, le problème de cyber sécurité aujourd'hui pose de plus en plus de problèmes. La protection de l'application et la prédiction d'une faille de l'application sont désormais indispensable. Dans ce contexte-là, nous avons utilisé deux outils différents pour réaliser une modélisation du patron sécurité.

2. Modélisation en UML

2.1 Système SCADA

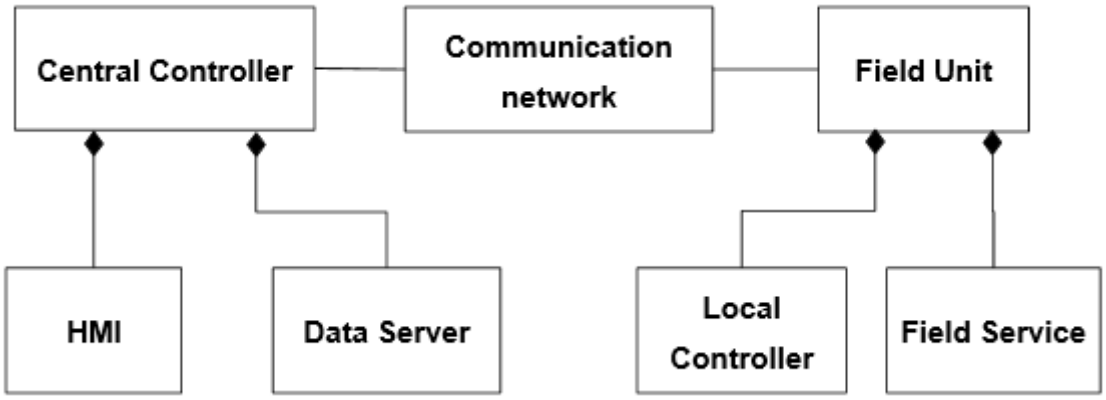


Figure 2.1-1 le diagramme de SCADA

Le system SCADA comporte une unité centrale, des réseaux et communications, et une unité locale. Dans l'unité centrale, l'utilisateur est permet de superviser et de commander les processus par HMI (une interface homme-machine), et de regarder les données en Data Server. Une infrastructure de communication relie le système de supervision et contrôle locale, et puis, l'utilisateur peut commander le service local par le contrôleur local.

2.2 Risque de SCADA

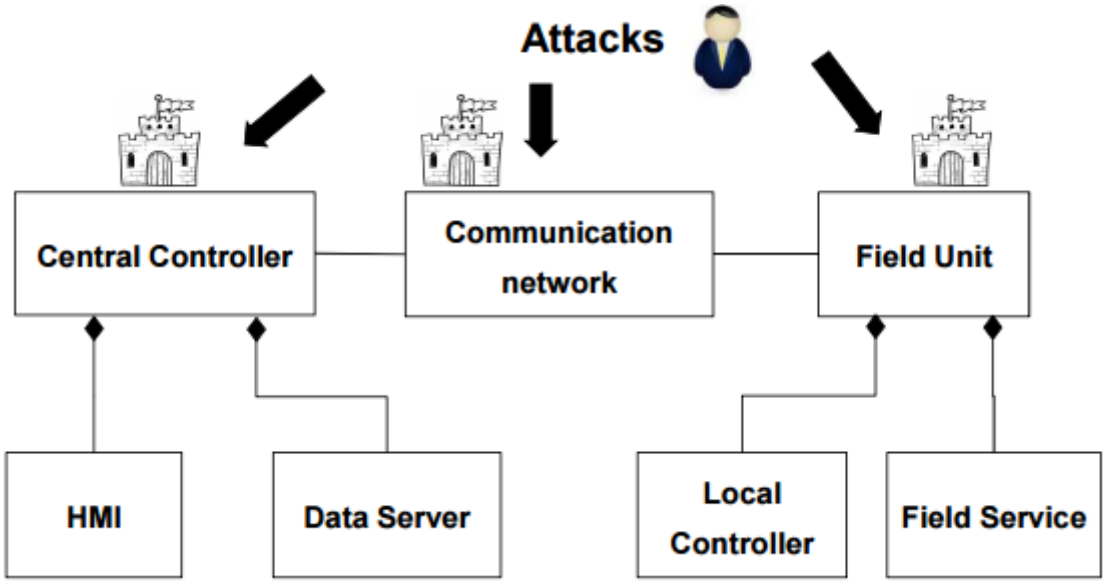


Figure 2.2-1 le problème de sécurité

Dans un SCADA basique, il n'y a pas de protection ; les attaquants peuvent attaquer le Central Controller ou le réseau communiquant ou le Field Unit. Donc nous avons besoin de protéger tous les parties. Par contre pour simplifier la modélisation, nous avons ajouté le patron sécurité seulement pour le Central Controller.

La sécurité pattern comporte l'identification et l'authentification, le contrôle d'accès et l'autorisation.

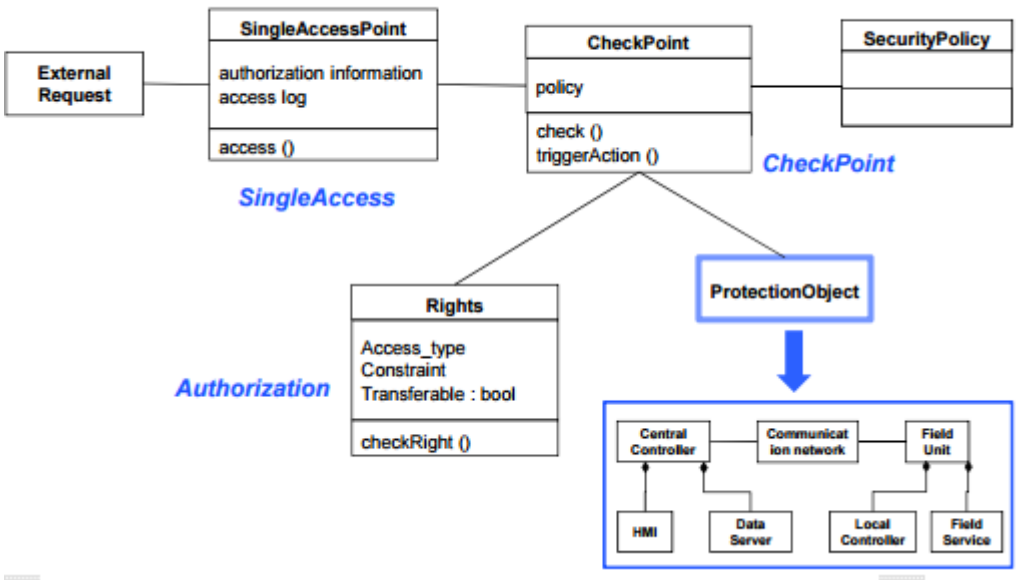


Figure 2.2-2 patron de sécurité

2.3 Réalisation sous Rhapsody

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

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

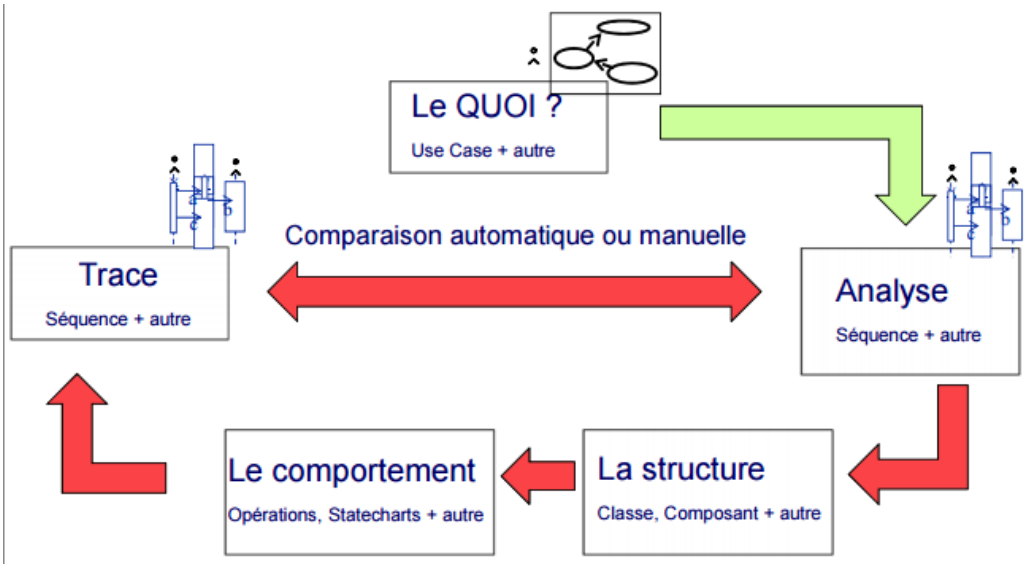


Figure 2.3-1 Méthodologie de Modélisation avec UML

2.3.1 Diagramme de use case

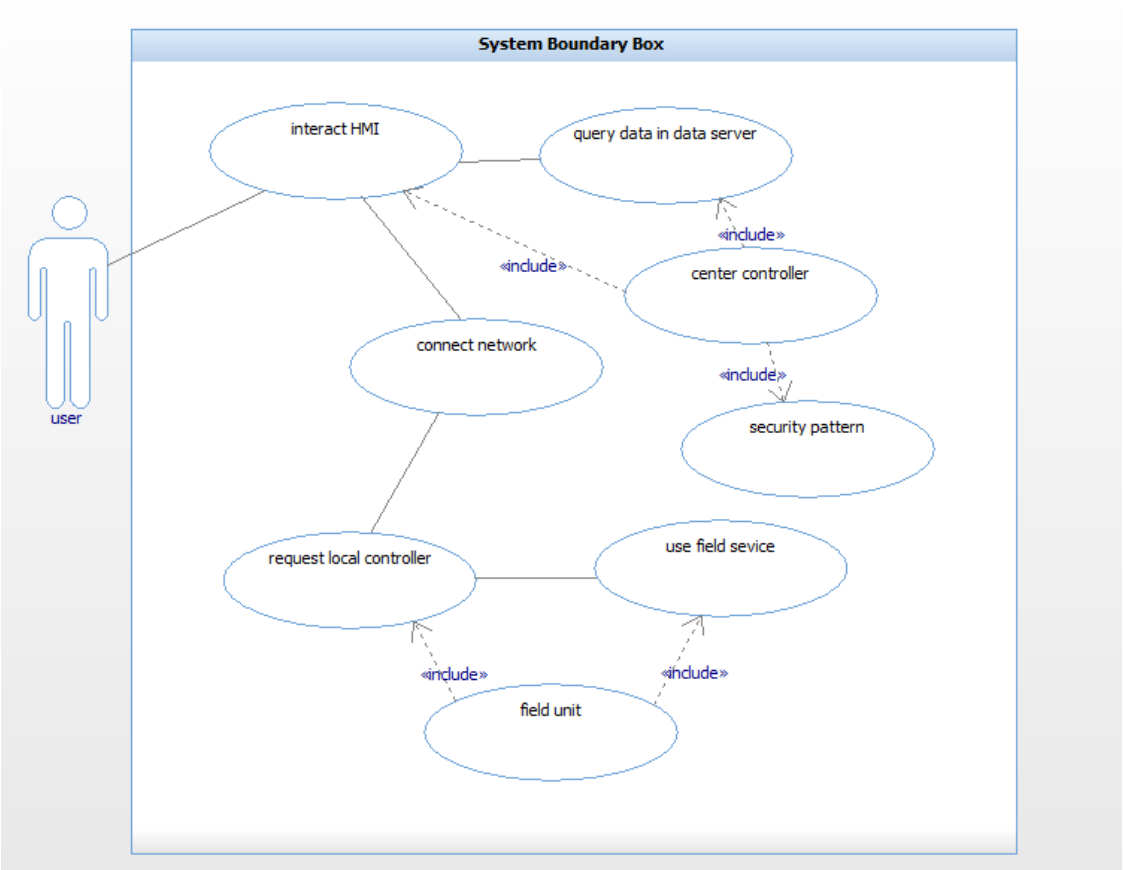


Figure 2.3.1-1 le diagramme de use case

Le diagramme de use case définit une manière d'utiliser le système et permet d'en décrire les exigences fonctionnelles. On définit un opérateur qui interagit avec le system SCADA, avant il fait la commande par HMI, il faut passer la sécurité pattern. Si l'utilisateur qui réussit de authentifier, il peut rechercher les données dans la system de supervision, ou il peut commander le service local via les réseaux.

2.3.2 Diagramme de scénario

En fait, le patron sécurité contient deux parties principales :

- **Authentification**

  Pour chaque utilisateur qui veux connecter sur le système, il faut envoyer un « usrid », pour simplifie, nous avons défini que :

Tableau Usrid associé avec le rôle

Usrid	Role
aaa	Admin
bbb	Center Controller (CC)
ccc	Local Controller(LC)
Other	Unknown

- **Autorisation**

En fait, les droits des utilisateurs sont définis ci-dessous :

Tableau les droits des utilisateurs différents

	Central Controller		Local Controller	
	read	write	read	write
Aministrator	Yes	Yes	Yes	Yes
CC Owner	Yes	Yes	No	No
LC Owner	No	No	Yes	No
Unk. Entity	No	No	No	No

A partir de ces droits, pour simplifier, nous avons défini que un utilisateur peuvent effectuer une mission dans un lien seulement, c'est-à-dire, soit une mission à Central Controller, soit une mission à Field Unit. Alors nous représentons les droits par un nombre.

Tableau les droits des utilisateurs en représentant avec les nombres

	Read in Central Controller (1 pour oui 0 pour non)	Write in Central Controller (1 pour oui 0 pour non)	Read in Local Controller (1 pour oui 0 pour non)	Write in Local Controller (1 pour oui 0 pour non)	Final score
CenterTask_rw	1	1	0	0	1100
LocalTask_rw	0	0	1	1	11
LocalTask_r	0	0	1	0	10
Do noting	0	0	0	0	0

Pour effectuer une opération, l'utilisateur doit envoyer une mission d'abord en attachant un usrid. Avec cet usrid, nous avons obtenu le droit de cet utilisateur en vérifiant avec le Rights. A partir du droit et la mission, nous allons vérifier l'autorisation.

1) Rejet de l'authentification

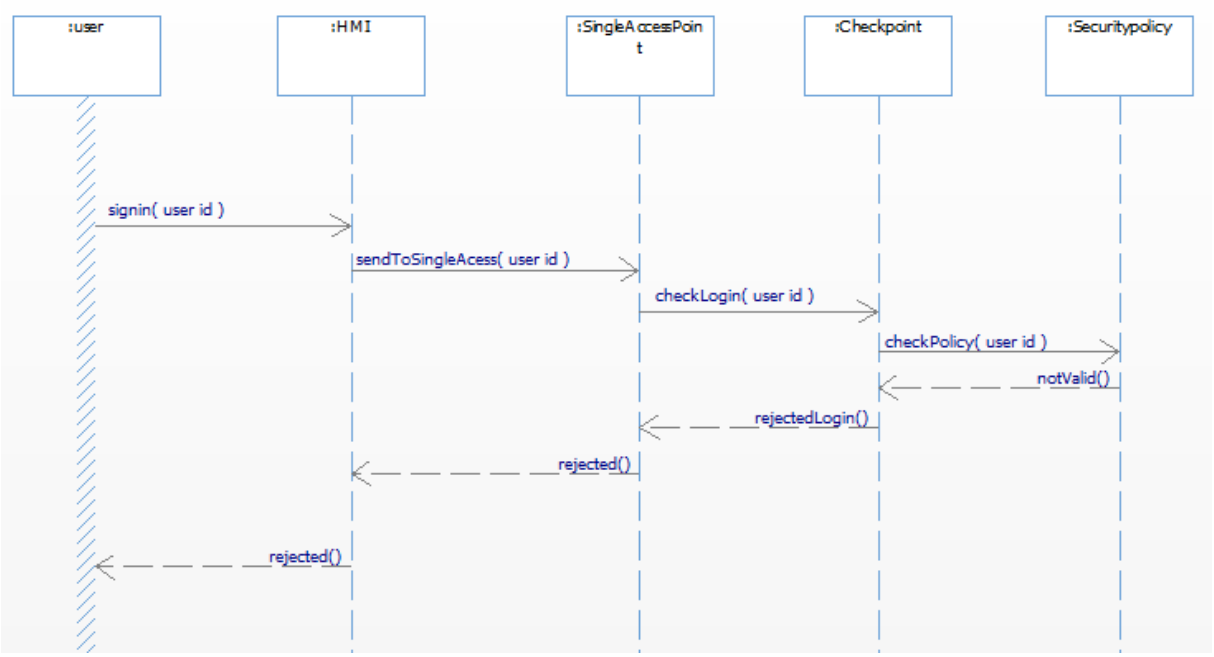


Figure2.3.2-1 Scenario refuser les inconnus

On définit que l'utilisateur interagit avec HMI. Avant de faire les opérations, il faut passer la sécurité pattern. Il entrée son id, et puis HMI envoie le id au Checkpoint pour l'identification via le SingleAccessPoint, le checkpoint envoye le id au SecurityPolicy pour authentifier. Quand l'utilisateur est inconnu, il va renvoyer un feedback non valide au Checkpoint, et le Checkpoint va le renvoyer au SingleAccessPoint, le SingleAccessPoint va rejeter l'utilisateur inconnu.

2) Rejet de l'autorisation

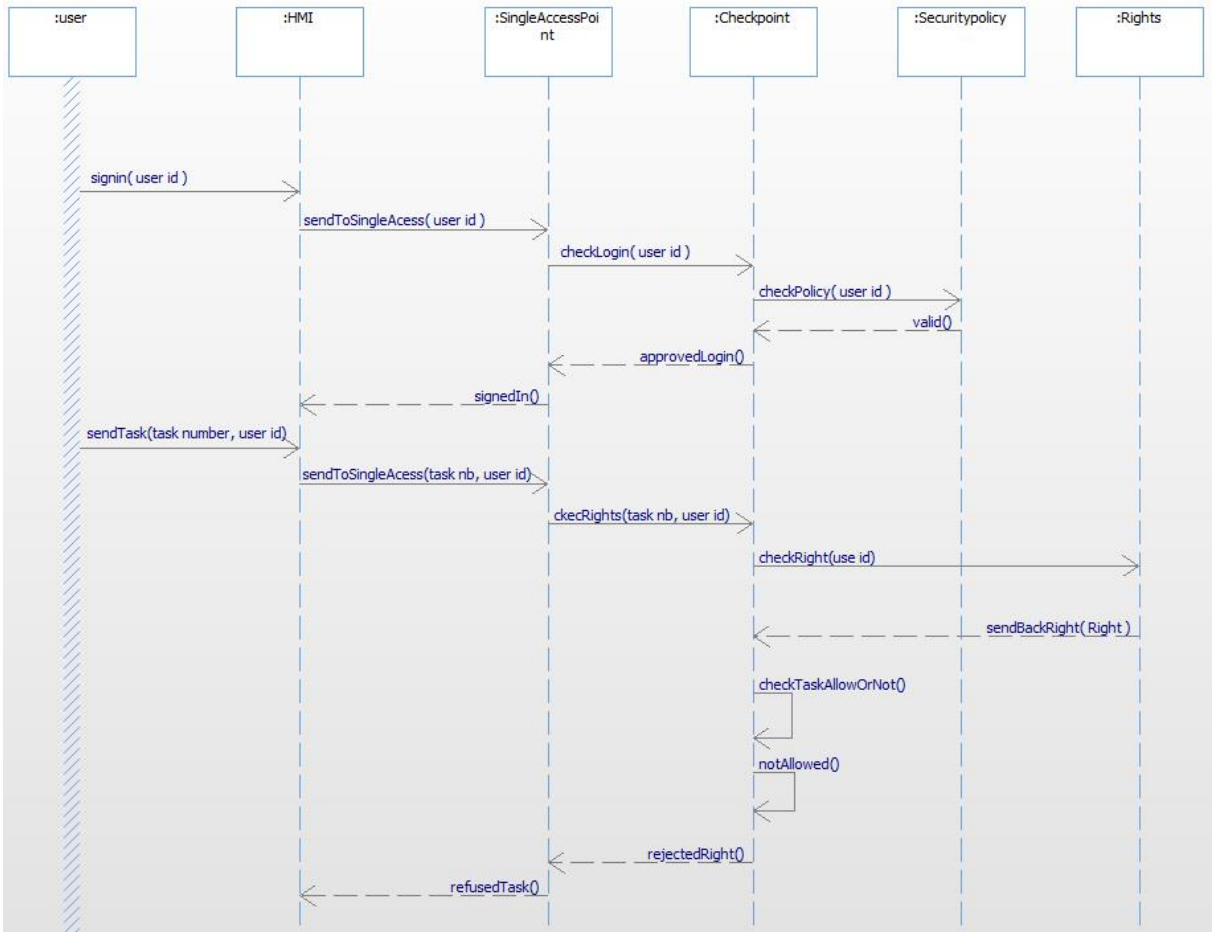


Figure2.3.2-2 Scenario l'authentification valide & l'autorisation non valide

Dans ce cas-là, l'utilisateur interagit avec HMI. Et il est identifié par le SecurityPolicy. Ainsi, il réussit de login dans le system de SCADA. Après, il commande une mission dans le HMI. Le HMI envoie son id et numéro de mission au Checkpoint via le SingleAccessPoint, le checkpoint envoie l'id au Rights pour juger l'identité de l'utilisateur, il y a administrateur qui peut gérer tout le system, le contrôleur centrale qui gère l'unité de centrale, mais pas droit de gère l'unité locale, le contrôleur locale qui n'a qu'un droit de lire dans l'unité locale. Apres il le renvoie au Checkpoint, on peut vérifier son droit et le numéro de mission ce qu'il choisit et puis juger son autorisation. S'il son identité ne match pas le droit, le Checkpoint va renvoyer l'autorisation non valide au SingleAccessPoint.

3) Validation de l'authentification et l'autorisation

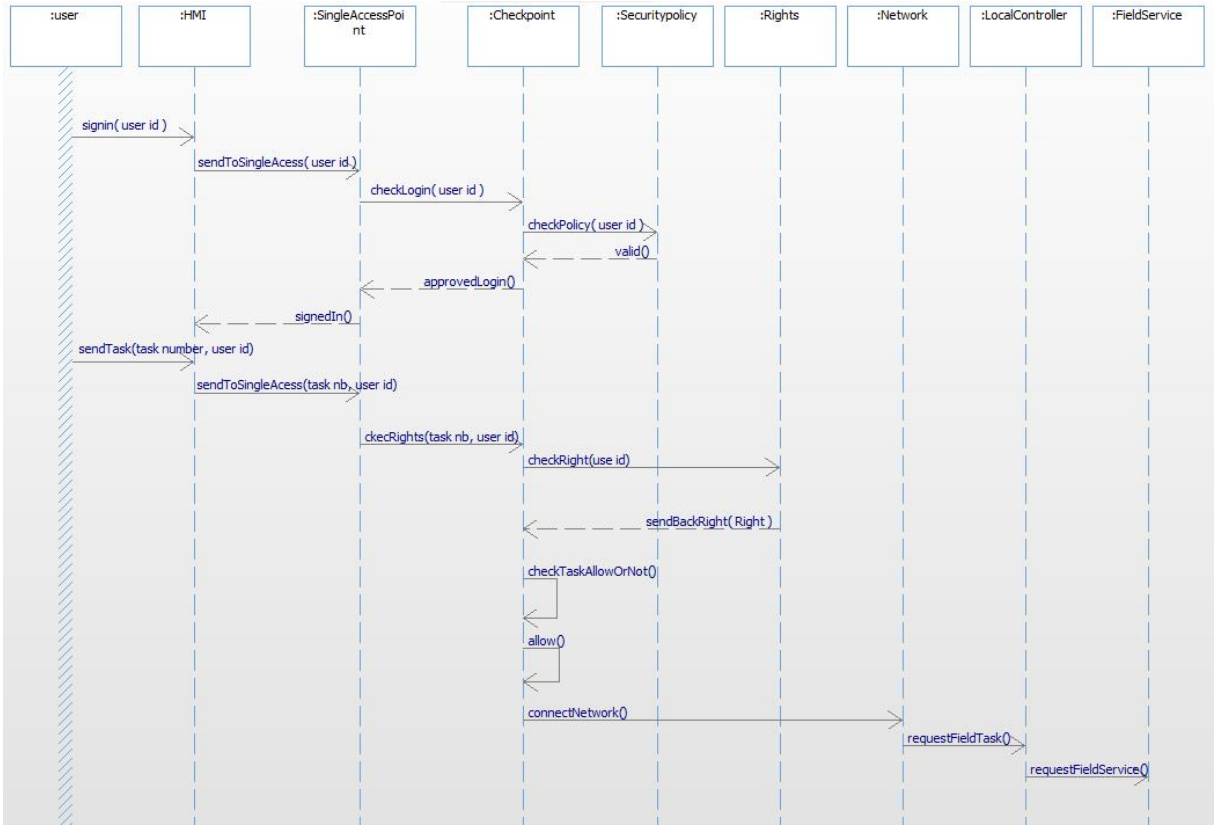


Figure2.3.2-3 Scenario l'authentification valide & l'autorisation valide

Dans ce cas-là, l'utilisateur interagit avec HMI. Il réussit de login dans le system de SCADA. Et son identité est match avec le droit. Ainsi le Checkpoint envoie la mission au Contrôle Locale via communication de réseau, et command le service.

2.3.3 Diagramme de classes

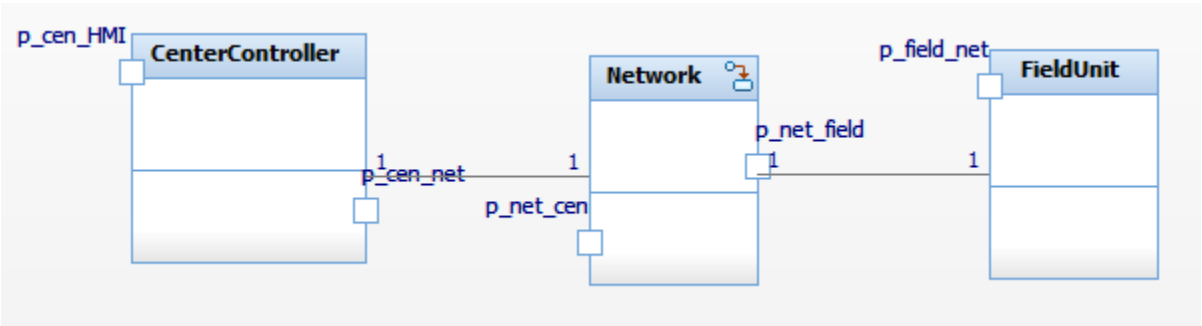


Figure2.3.3-1 le diagramme de classe de sécurité pattern

2.3.4 Diagramme d'objet

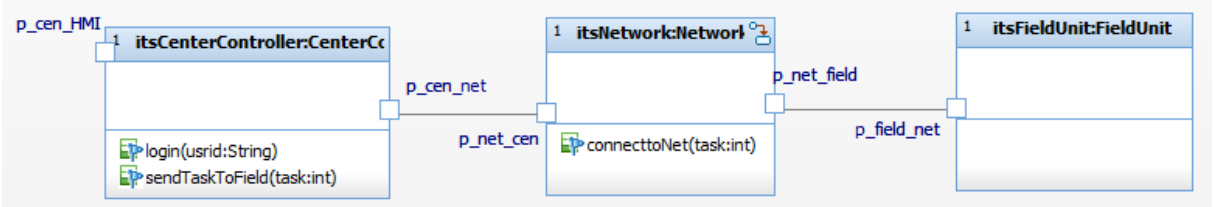


Figure2.3.4-1 le diagramme d'objet

Le diagramme d'objet présent les relations entre les objets. Il est l'instanciation de diagramme de class. Les objets sont relies pas les ports et la relation Link.

2.3.5 Diagramme de structure

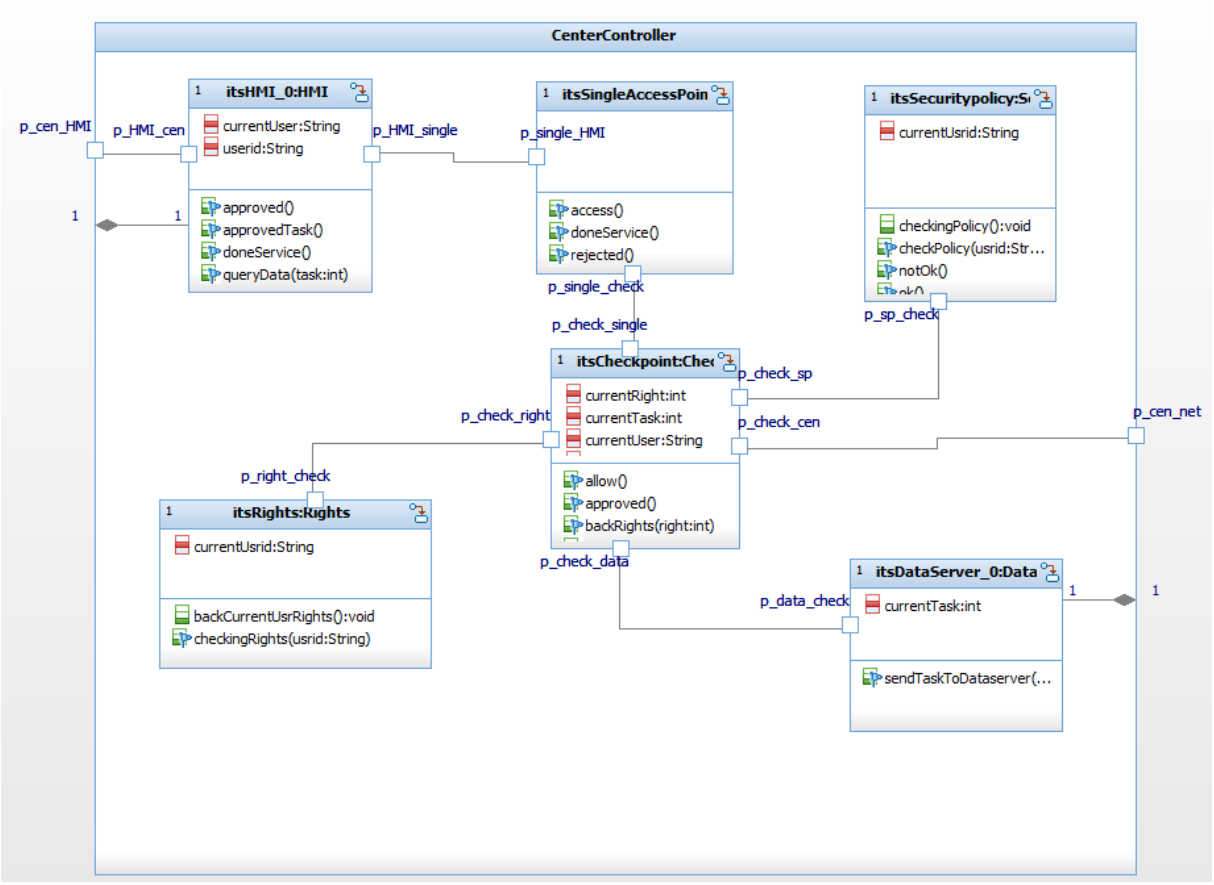


Figure2.3.5-1 le diagramme Structure d'unité centrale

L'utilisateur interagit avec HMI via le port de p_cen_HMI. Et puis, HMI connecte avec la sécurité pattern pour vérifier l'identité de l'utilisateur. Apres il réussit de login, il peut connecter les réseaux par le port p_cen_net, et commander le service locale.

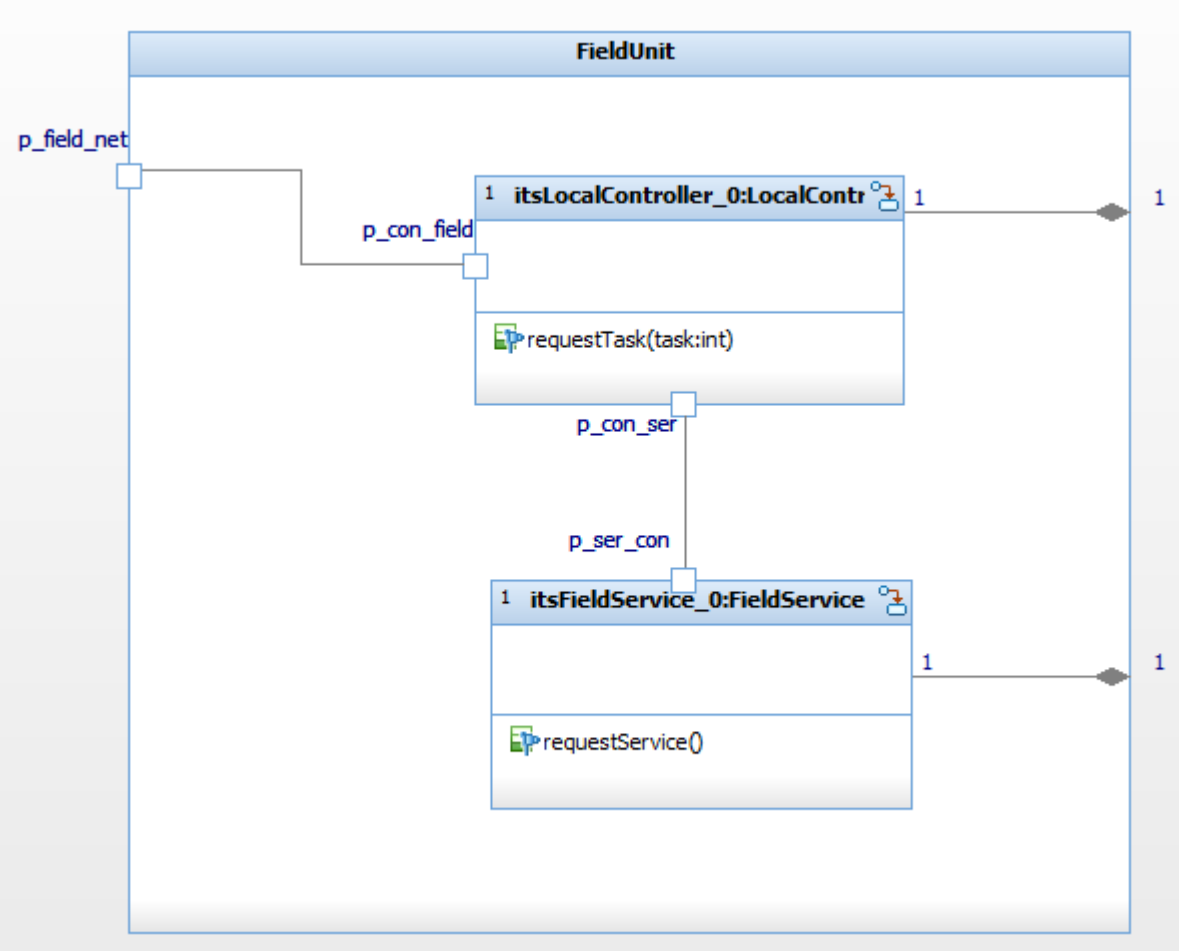


Figure2.3.5-2 le diagramme Structure d'unité locale

L'unité locale est lie avec le réseau via le port p_field_net. Quand il reçoit le commande par le system de supervision, le contrôle locale va commander le service locale.

2.3.6 Diagramme d'états principaux

1) HMI

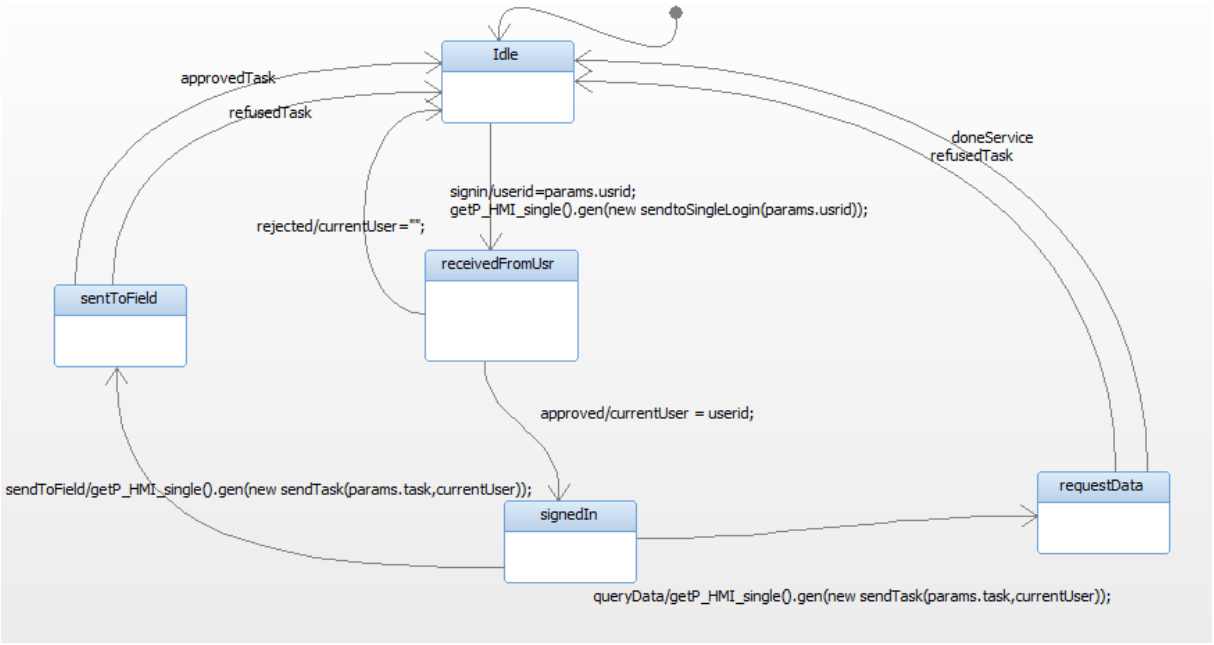


Figure2.3.6-1 automate de HMI

L'automate de HMI contient quatre états. Idle attend l'utilisateur de commander login. S'il reçoit le feedback non valide, il retourne à l'Idle, sinon il est en état signedIn. Et puis, l'utilisateur peut choisir son mission, commande à l'unité locale ou recherche les données.

2) Checkpoint

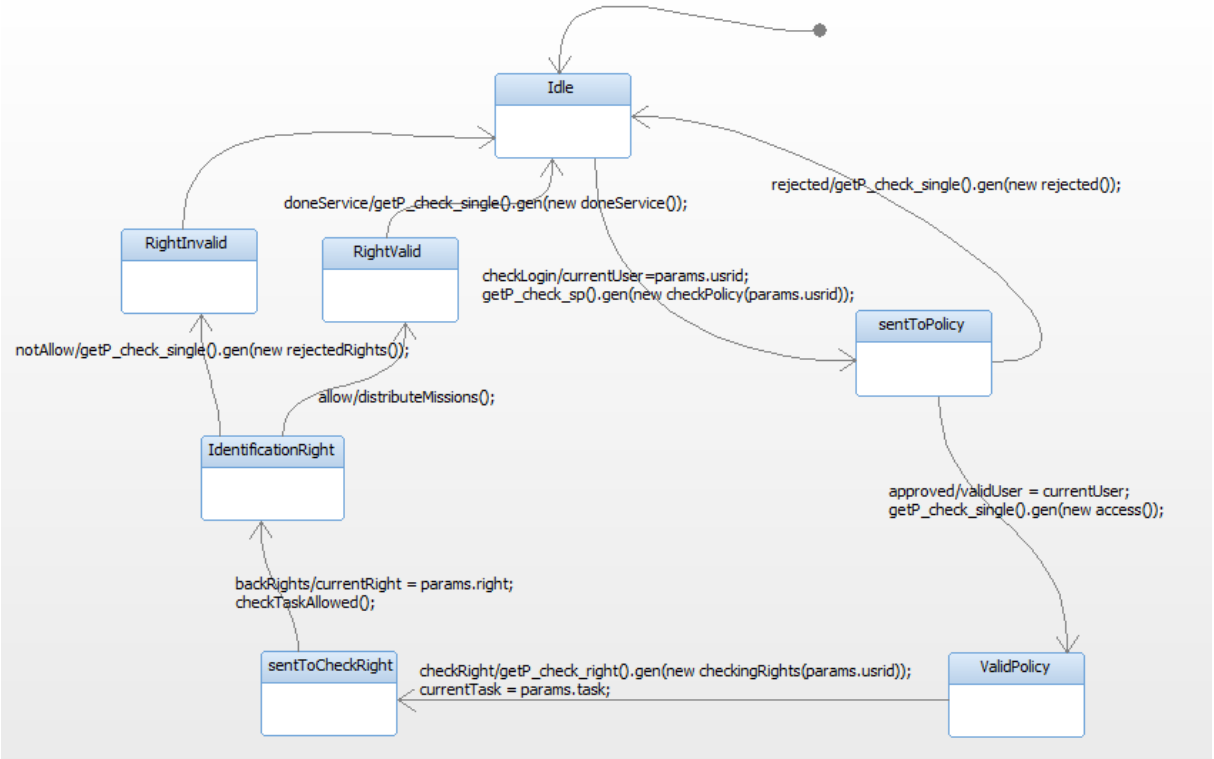


Figure2.3.6-2 automate de Checkpoint

Dans l'automate de Checkpoint, quand il reçut un id de l'utilisateur, il va l'envoyer au SerityPolicy, si l'authentification est non valide, il va retourner à Idle. Sinon, il va à l'état ValidPolicy et attend la commande de l'utilisateur. Quand il reçut la mission choisie par l'utilisateur, il envoie le numéro de mission à l'automate Rights. Si l'identité de retourne est match avec la mission, il peut faire les opérations ce qu'il veut.

3) secuPolicy

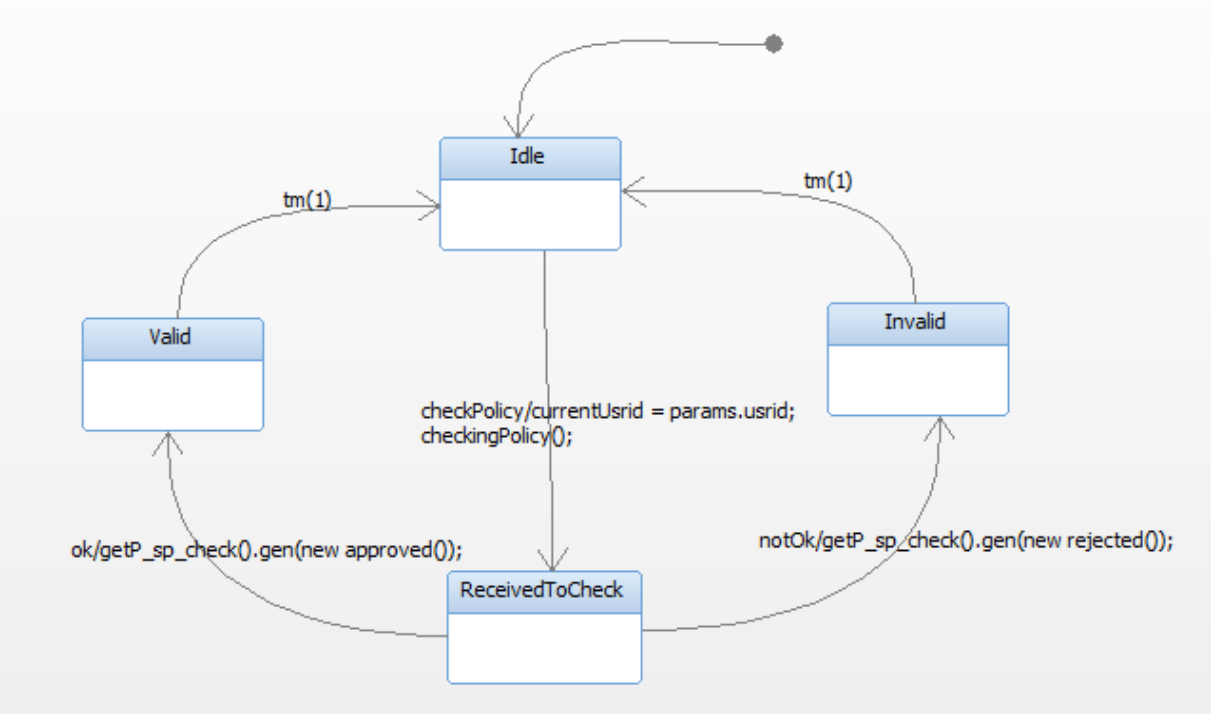


Figure2.3.6-3 automate de Policy

Dans l'automate de SecurityPolicy, il vérifie l'authentification de l'utilisateur. S'il est inconnu, il renvoie non valide au Checkpoint, sinon, il renvoie valide.

4) Rights

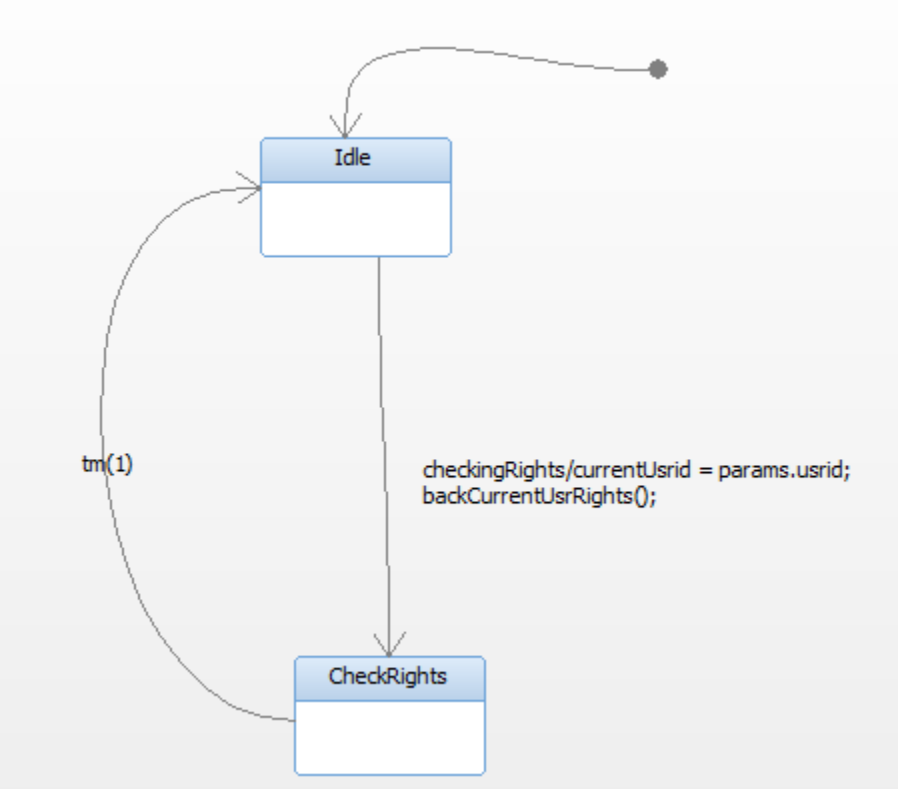


Figure2.3.6-4 automate de Droit

Dans cette automate, il reçut un id de l'utilisateur, et juge son identité, l'administrateur, le contrôleur centrale, le contrôleur locale. Et puis, renvoie au checkpoint.

5) Network

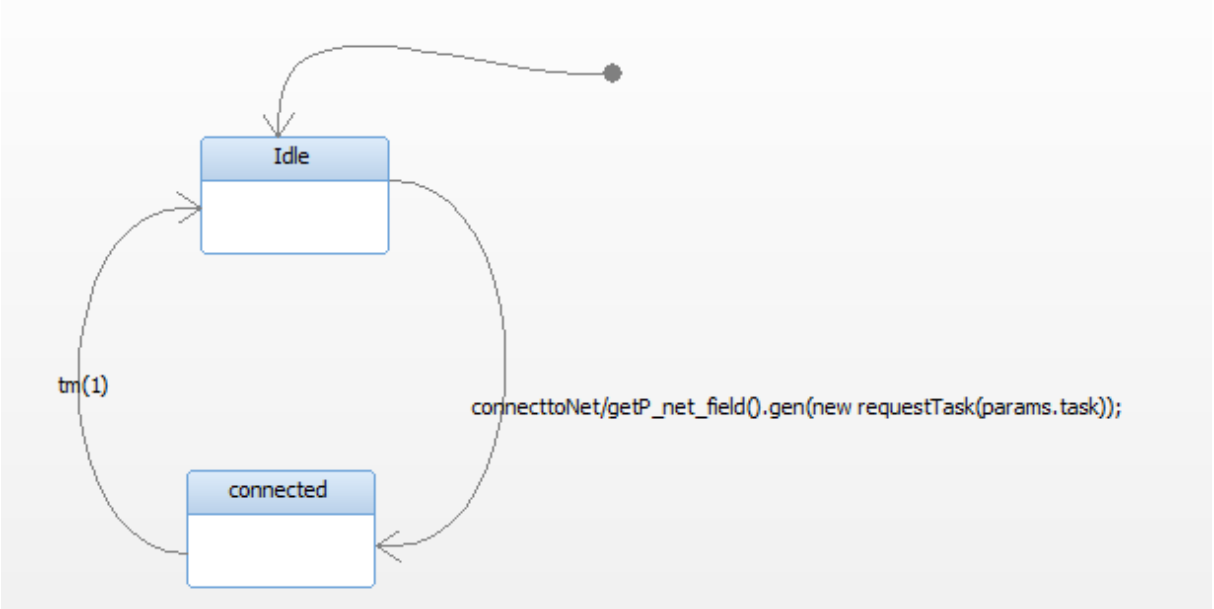


Figure2.3.6-5 automate de réseau

La communication de réseau qui relie le centre et local, et envoie le commande par centrale a l'unité locale.

6) LocalControler

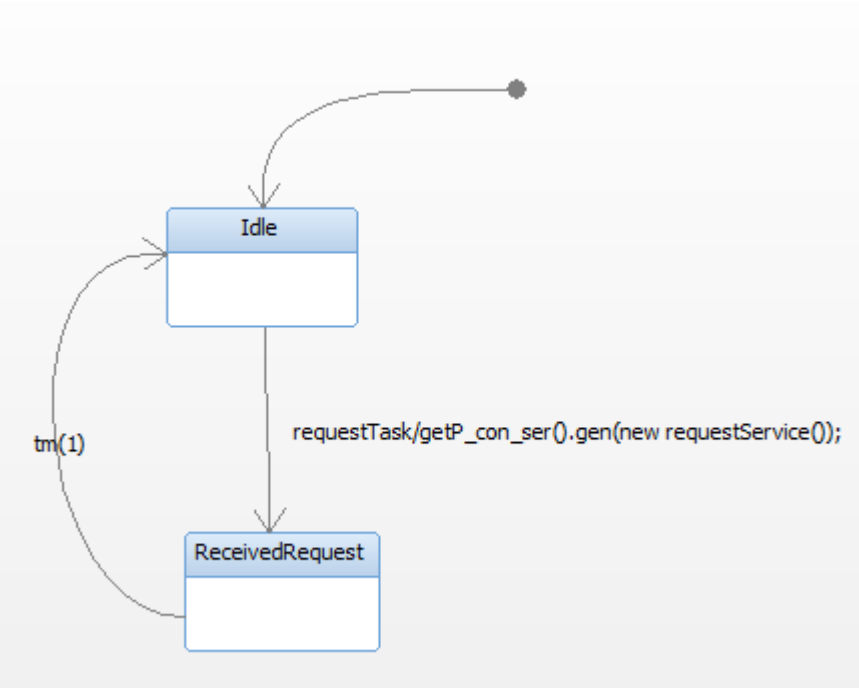


Figure2.3.6-6 automate de localControler

LocalControler attend le command de centrale, quand il reçoit le numéro de mission, il va commander le service correspond.

3. Modélisation en Fiacre avec la vérfication

3.1 Modélisation

A partir du même modèle, nous avons défini le même structure en ajoutant un processus utilisateur pour gérer des évènements automatiquement.

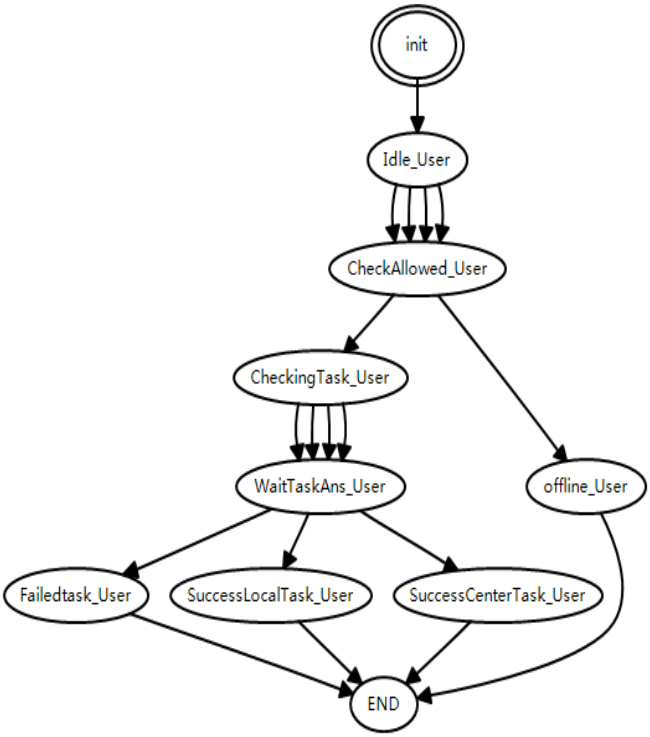
En fiacre, il n'y a pas de type String, donc nous avons le usrid par un nombre au lieu d'un String :

Tableau usrid associé avec les rôles différents en Fiacre

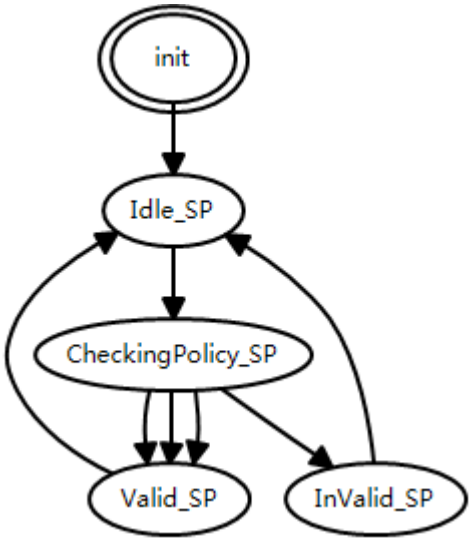
Usrid	Role
111	Admin
222	Center Controller (CC)
333	Local Controller(LC)
Other	Unknown

Nous avons défini deux FIFOs, un pour envoyer les identifiants, l'autre pour envoyer les missions.

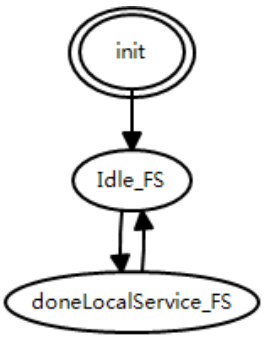
Alors, avec l'outil Obp, nous avons obtenu les diagrammes d'état :



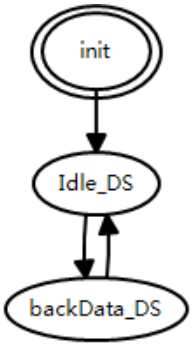
User



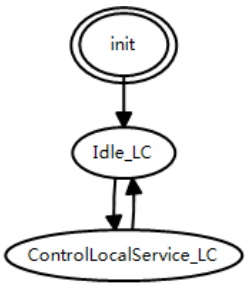
Security Policy



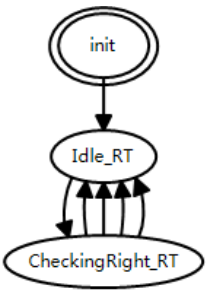
Local Service



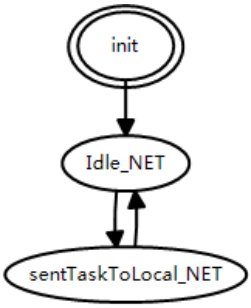
Data Server



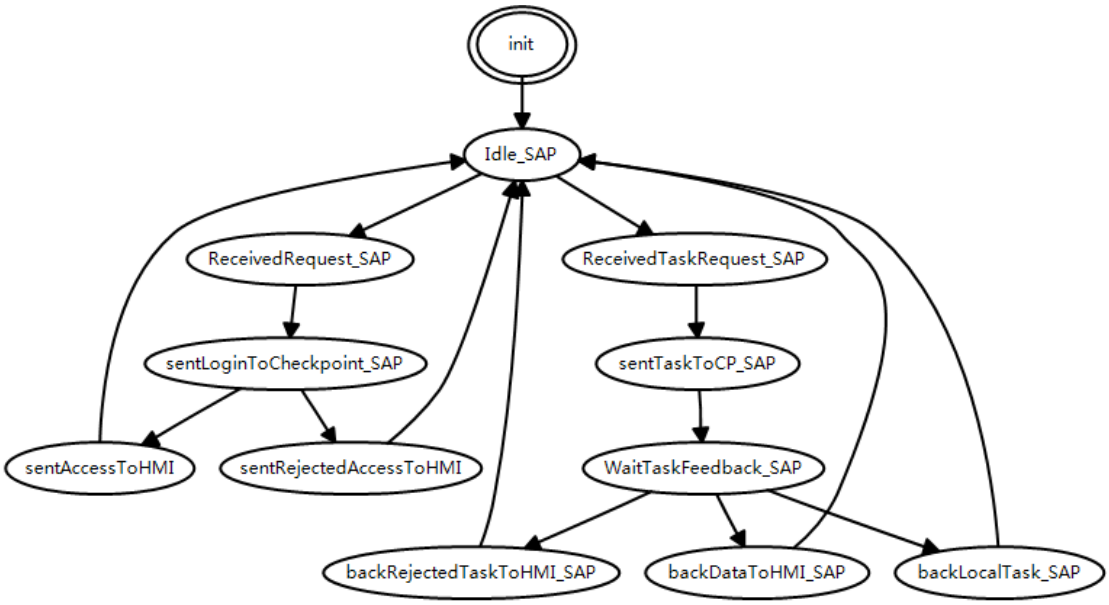
Local Controller



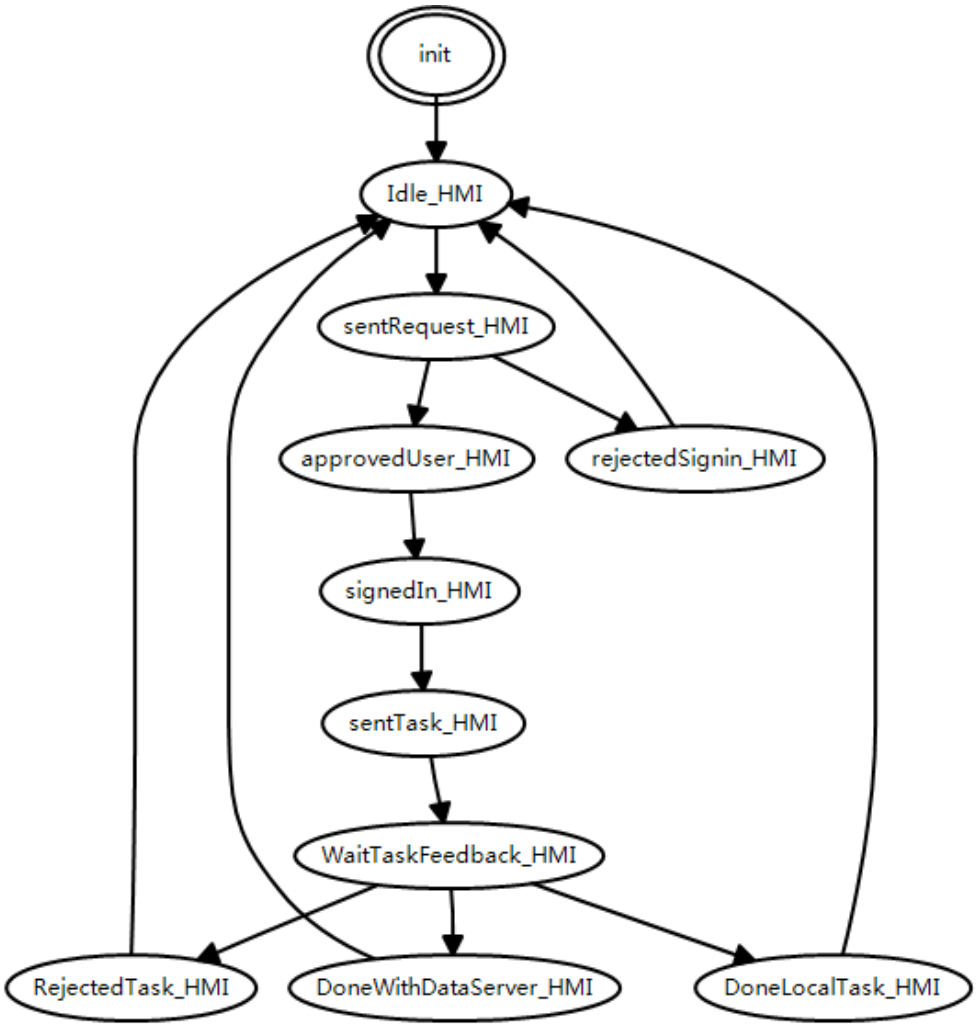
Right



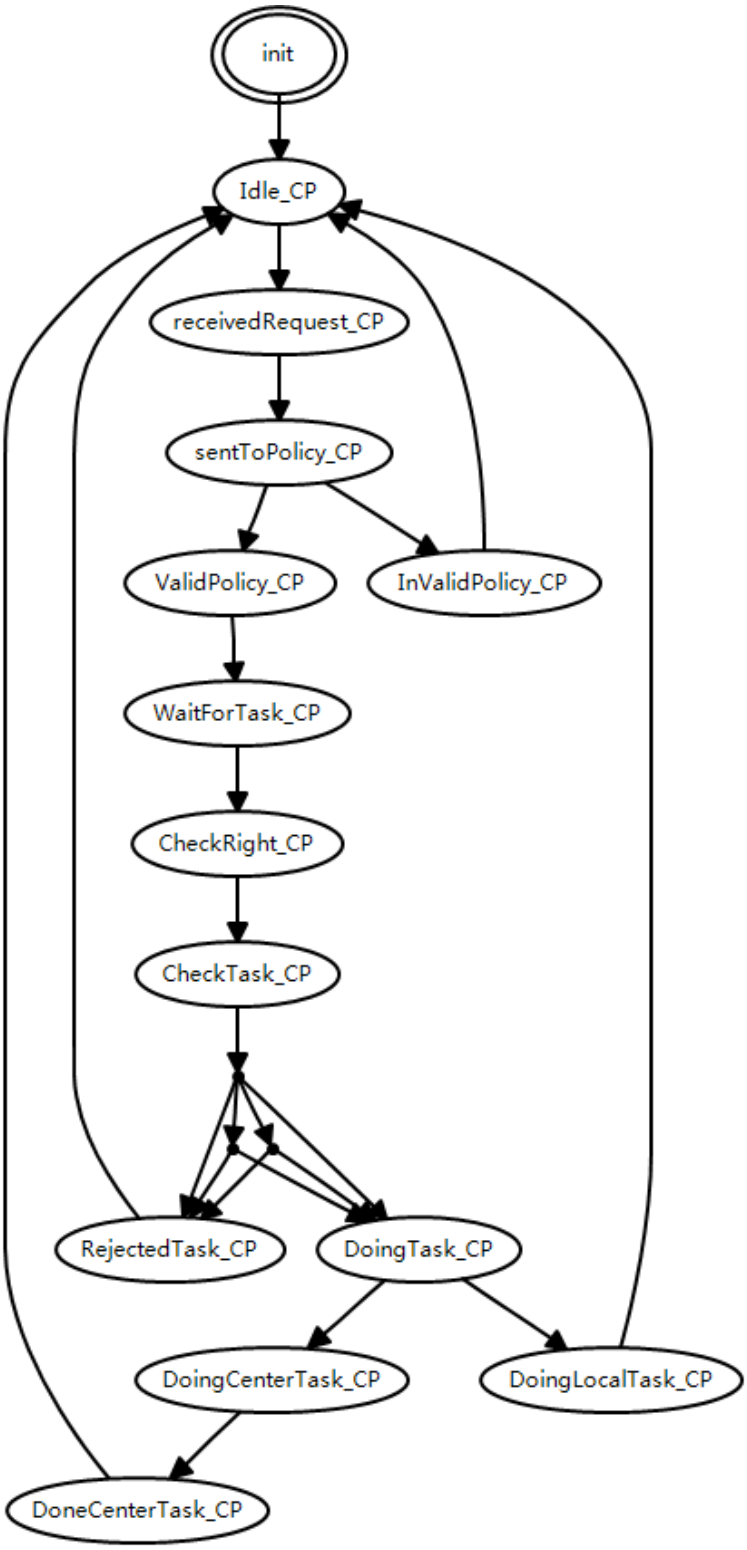
Network



Single Access Point



HMI



Check Point

3.2 Vérification

Les vérifications nécessaires sont les vérifications pour les droits d'utilisateurs.

Tableau propriété de vérification sous cdl

	Center Read and Write	Local Read	Local Write
Admin	pte_Center_ADM	pte_Local_ADM	
Center Controller	pte_Center_CC	pte_Localr_CC	pte_Localrw_CC
Local Controller	pte_Center_LC	pte_Localr_CC	pte_Localrw_CC
Unknown	pte_Access_UNK		

Avec l'exécution sous obp/cdl, il y a 1089 états et 2376 transitions, alors il est trop grand. Donc nous avons choisi la trace 777 pour vérifier le diagramme de séquence.

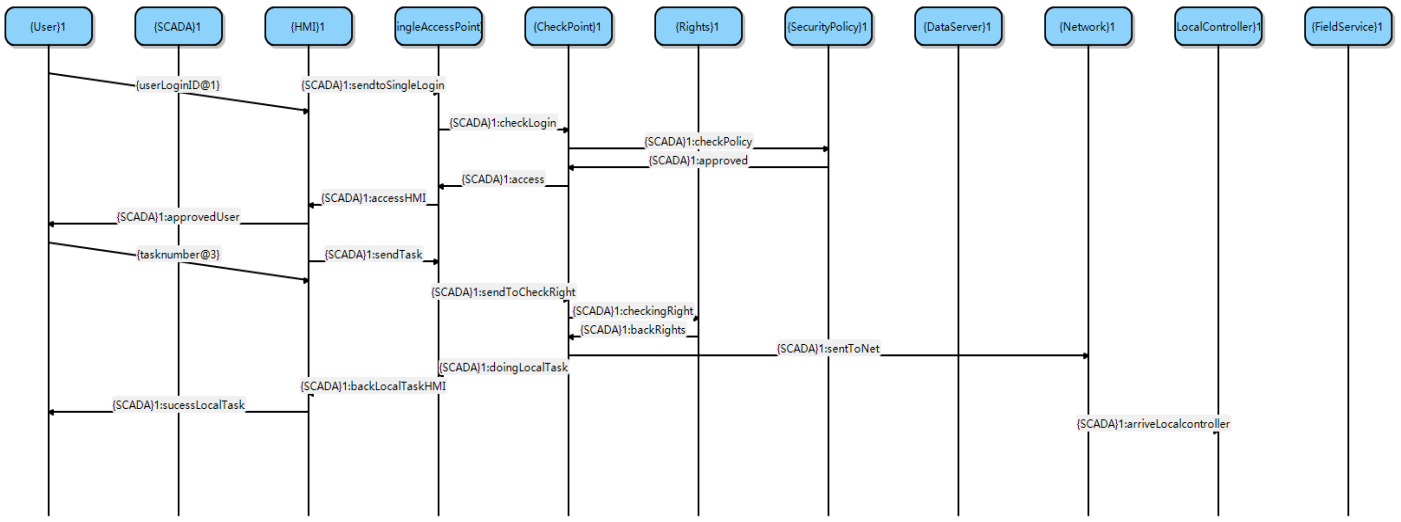


Figure 3.2-1 trace de diagramme de séquence sous obp

Conclusion

En comptant la trace sous obp/cdl et le trace sous Rhapsody, avec deux outils différents, nous avons obtenu le même résultat. Alors la réalisation du patron sécurité a été bien fait. En fait, avec Rhapsody, le développement est plus simple et plus visible, par contre, il est trop difficile de simuler tous les cas, par exemple, nous avons simulé l'utilisateur comme un évènement venant de l'environnement, donc pour chaque essaie, il faut gérer l'évènement manuellement. Avec l'outil obp/cdl, il est plus simple de simuler tous les cas, par contre le nombre d'états et transition est assez limité.