Modélisation, simulation et vérification formelle de patterns de sécurité

UV 5.6

Philippe Dhaussy, Joel Champeau

Lab-STICC
UMR CNRS 6285
ENSTA-Bretagne, Brest.

prenom.nom@ensta-bretagne.fr

Modélisation, simulation et vérification formelle de patterns de sécurité Application à la cyber-défense

- Politiques et patterns de sécurité
- Patterns de sécurité et SCADA
- Modélisation et simulation UML
- Mise en œuvre de la vérification formelle de propriétés
- Travail à réaliser

Critères Communs et Propriétés de sécurité

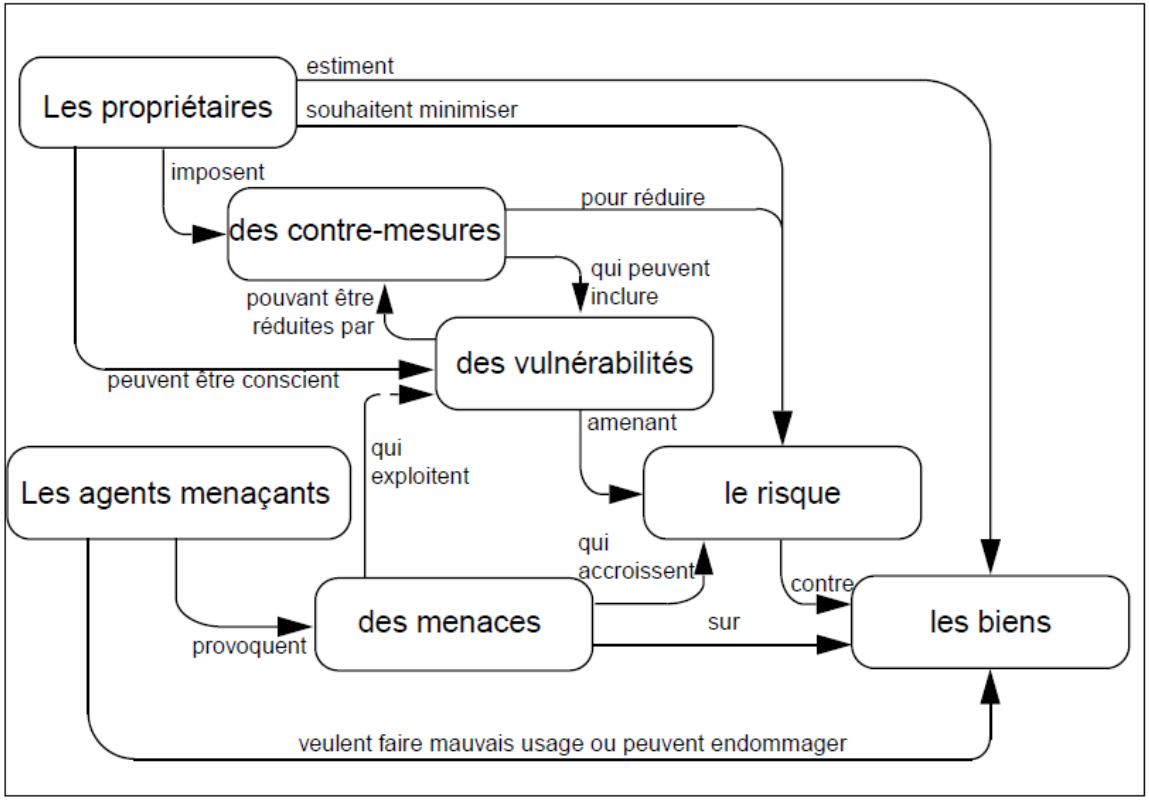
Critères Communs [Anssi-CC] :

- Critères pour l'évaluation des propriétés de sécurité
- Fournir une méthode d'évaluation partageable

Concepts de sécurité des Critères Communs

Objectifs essentiels :

- **Maitriser les risques des systèmes**
- **Augmenter la confiance dans les systèmes des propriétaires ou utilisateurs**



Critères Communs et Propriétés de sécurité

Altérations classiques des biens possibles :

- Perte de confidentialité
  - Divulgation nuisible du bien à des destinataires non autorisés
- Perte d'intégrité
  - Dommage provoqué par une modification non-autorisée
- Perte de disponibilité
  - Déni d'accès au bien

Identification de propriétés pour caractériser la sécurité des systèmes :

- Intégrité
- Confidentialité
- Disponibilité

Critères Communs et Propriétés de sécurité

Les propriétés de sécurité doivent être préservées tout au long du processus de développement.

- **Spécifications, conception et implantation**

Quelque soit l'étape du processus, on doit garantir :

- **Confidentialité :** non divulgation d'information aux entités non autorisées (protection de la vie privée, identification, etc…)
- **Intégrité :** Pas d'altération ou de destruction, volontaire ou accidentelle, des données. Lors de leur traitement, de la conservation ou de la transmission.
- **Disponibilité :** Fonctionnement sans faille et en garantissant l'accès aux services avec leur QoS. Proche à la sûreté de fonctionnement. Liée au contexte et prend en compte les temps de réponses, les modèles de fautes (pannes franches, fautes d'omissions, temporelles, byzantines).

Critères Communs et Propriétés de sécurité

Autres propriétés plus fines : Intimité (privacy), Authenticité / non-répudiation, Responsabilité, Pérennité, Exclusivité, Protection de la propriété intellectuelle, … [TCSEC, 1985, ITSEC 1991, Bishop, 2003, Clemente 2010, Rouzaud Cornabas 2010,….].

Critères Communs et Propriétés de sécurité

Un des objectifs des CC est de fournir un processus : Secure by Design

Etapes d'analyse, de spécification et de conception :

- Définir les objectifs de sécurité par analyse système
- Mettre en œuvre les objectifs en propageant les exigences de sécurité
  - Exigences fonctionnelles et fonctions de sécurité
  - Chemin de confiance dans les composants fonctionnels de sécurité
- Conception des composants fonctionnels de sécurité
  - **Conception orientée utilisation Security patterns**

Security Patterns

Describe general reusable solution to a well-known problem of security

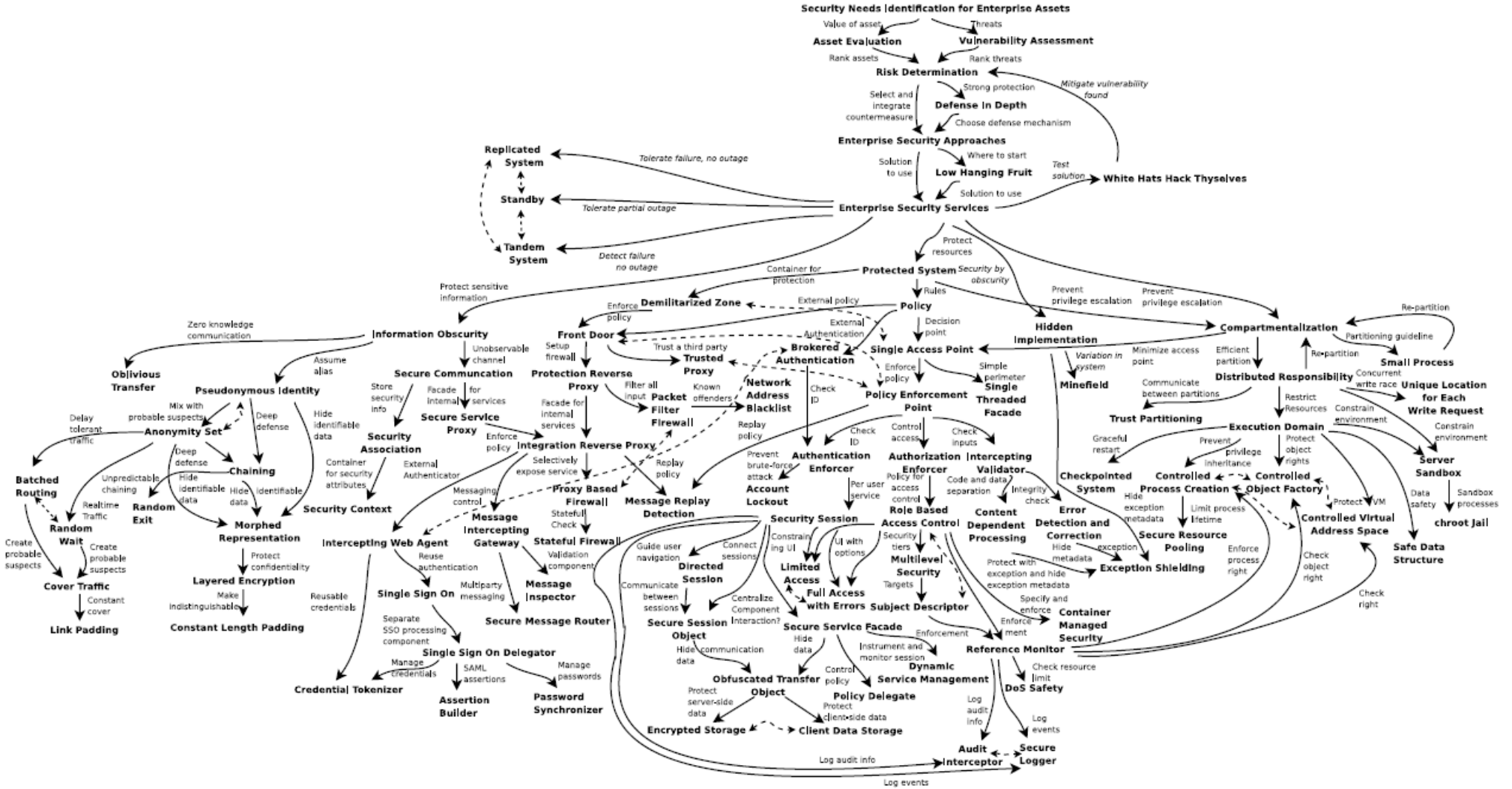
- [Yoder & Barcalow, Proc of 4th Pattern Language of programs, 1997]
- [Schumacher, Roedig, 2001]
- [Schumacher, Fernandez, Hybertson, Buschmann. Wiley & Sons, 2005]
- [Fernandez, 2006]
- [Heyman, Yskout, Scandariato, Joosen. Proc. of 3rd International Workshop on Software Engineering for Secure Systems, 2007]
- [Yoshioka, Washizaki, Maruyma. Progress in Informatics, 2008]
- https://en.wikipedia.org/wiki/Security_Patterns

Security Patterns : classification

- [Washizaki, Fernandez, Maruyama, Kubo,Yoshioka. Int Conf on Database and Expert Systems Applications, 2009.]
- [Hafiz, Adamczyk, Johnson. IEEE Software, 2007]
- Hafiz, Johnson. Tech report, 2006]
- http://www.munawarhafiz.com/securitypatterncatalog/index.php

Security patterns and pattern language

[Hafiz & al, 2012]



Security Patterns

Sécurité est souvent contraire au secret :

- Favoriser la réutilisation de code connu et partagé
- Favoriser la réutilisation de solutions identifiées et éprouvées
- Chercher le « Secure by design »

Intérêt des Patterns dans un contexte de sécurité

- Capitaliser et partager les expériences
- Réutilisation de solutions éprouvées
- Pattern fourni une abstraction partageable
- Pattern et code associé
- Pattern solution générique et simple
- + de 400 patterns de sécurité
- Un exploit répertorié = pattern d'attaque

Security Patterns : problèmes

Modification fréquente du logiciel :

- Evolution des besoins,
- Introduction de bugs => failles de sécurité.
- Comment et ou introduire les patterns

Maintenance difficile des patterns dans les grands systèmes

- Le système conserve-t-il ses propriétés de sécurité ?
- Comment introduire un pattern suite à un test de pénétration ou attaque?
- Reconnaissance, identification des patterns dans le code ?
- L'implantation est-elle conforme au pattern de sécurité voulu ?

Besoin en abstraction et modélisation

➔ But : Spécifier, Concevoir et Garantir que les exigences ou propriétés sont préservées

Security Patterns : problèmes

Besoin en abstraction et modélisation

Reverse engineering

- Reverse engineering to detect security patterns in code. [VanHilst, Fernandez. In Proc. of 1st International Workshop on Software Patterns and Quality. Information Processing Society of Japan, December 2007]
- Reconnaissance souvent difficile à partir du code
- Vérification des propriétés au niveau pattern plutôt que code

Voir exemple de Single Point Acces pattern …

Exemple de Security Pattern : Single Access Point

Synopsis :

- Protéger l'accès à un système

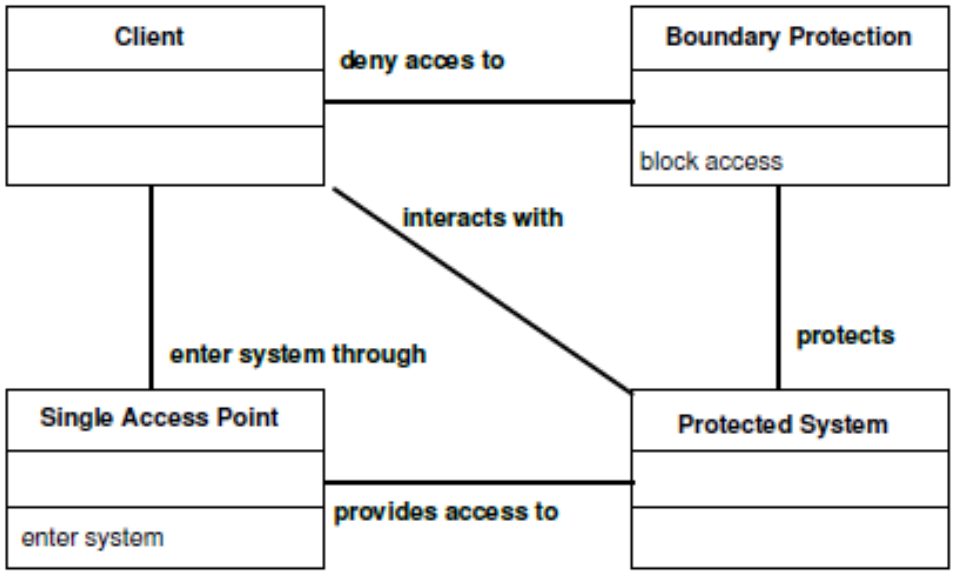
Contexte

- Plusieurs entités accèdent au système
- Protection du périmètre du système

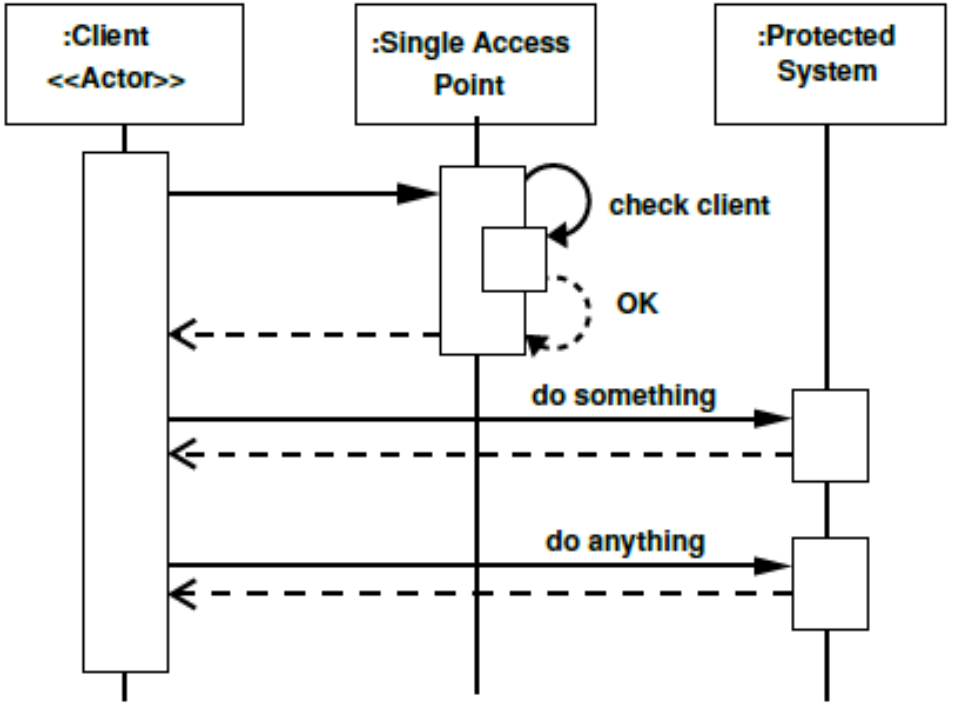
Forces

- Diminuer la surface d'attaque
- Focaliser les aspects authentification
- Focaliser la gestion des autorisations, droits d'accès et du chiffrement

Exemple de Security Pattern : Single Access Point



(a) *Single Access Point* UML diagram



(b) *Single Access Point* Sequence Diagram

Modélisation, simulation et vérification formelle de patterns de sécurité Application à la cyber-défense

- Politiques et patterns de sécurité
- Patterns de sécurité et SCADA
- Mise en œuvre de la vérification formelle de propriétés
- Modélisation et simulation UML
- Travail à réaliser

Designing Secure SCADA Systems Using Security Patterns

[Fernandez, Larrondo-Petrie, 2010]

Security countermeasures : five groups

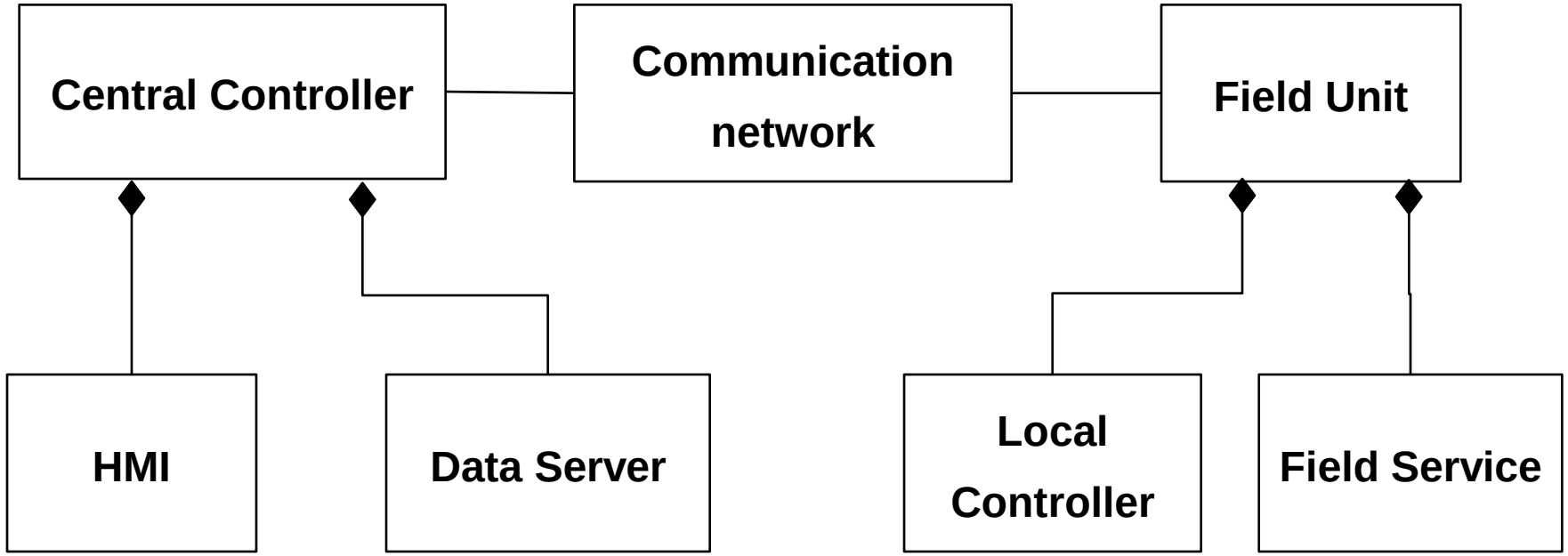
- Identification and Authentication,
- Access Control and Authorization,
- Logging,
- Cryptography,
- Intrusion detection.

Security patterns

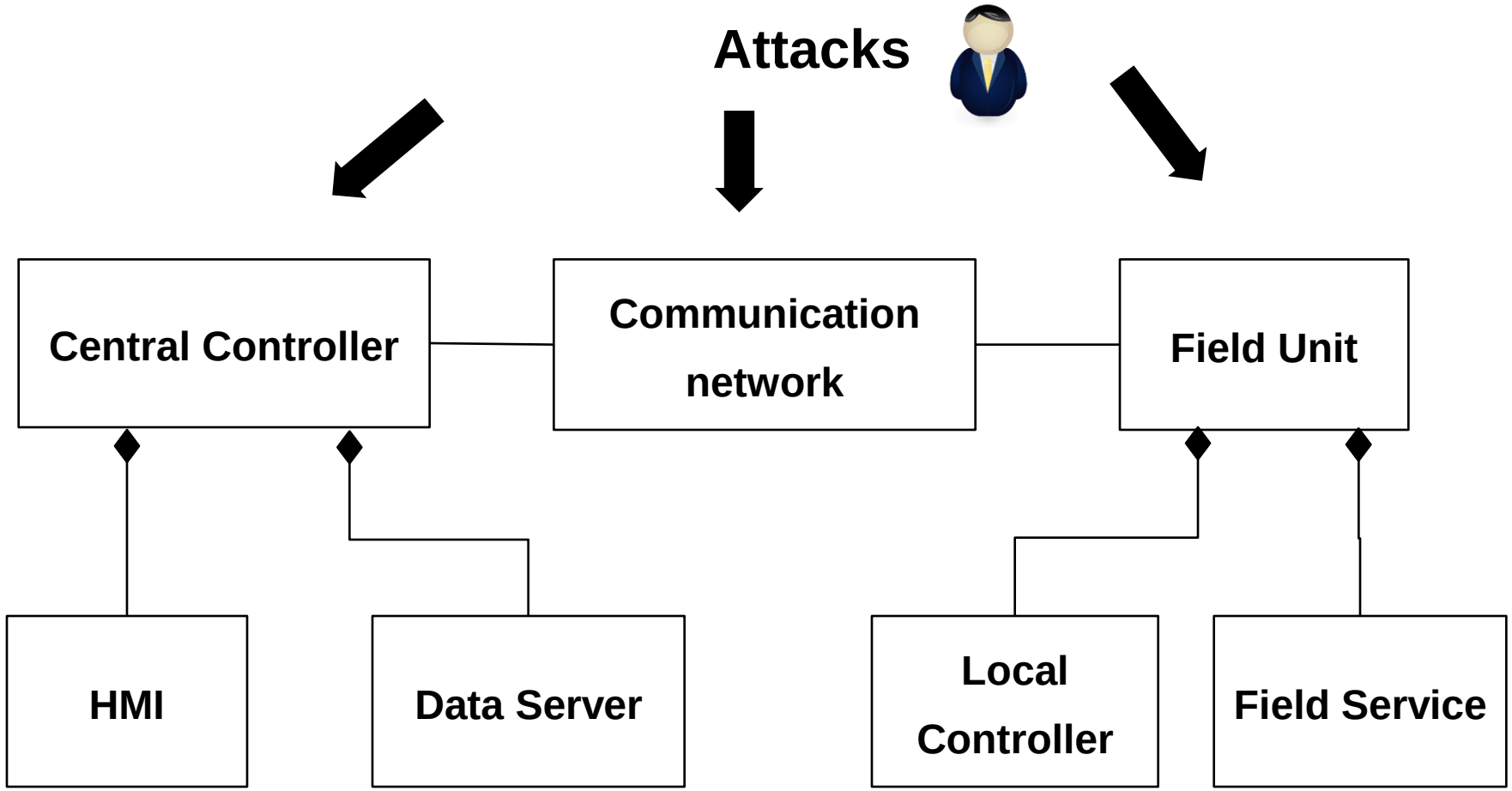
- describe **mechanisms** that fall into these categories (or combinations thereof)
- describe the **abstract models** that guide the design and evaluation of these mechanisms

Class diagram for a general SCADA System

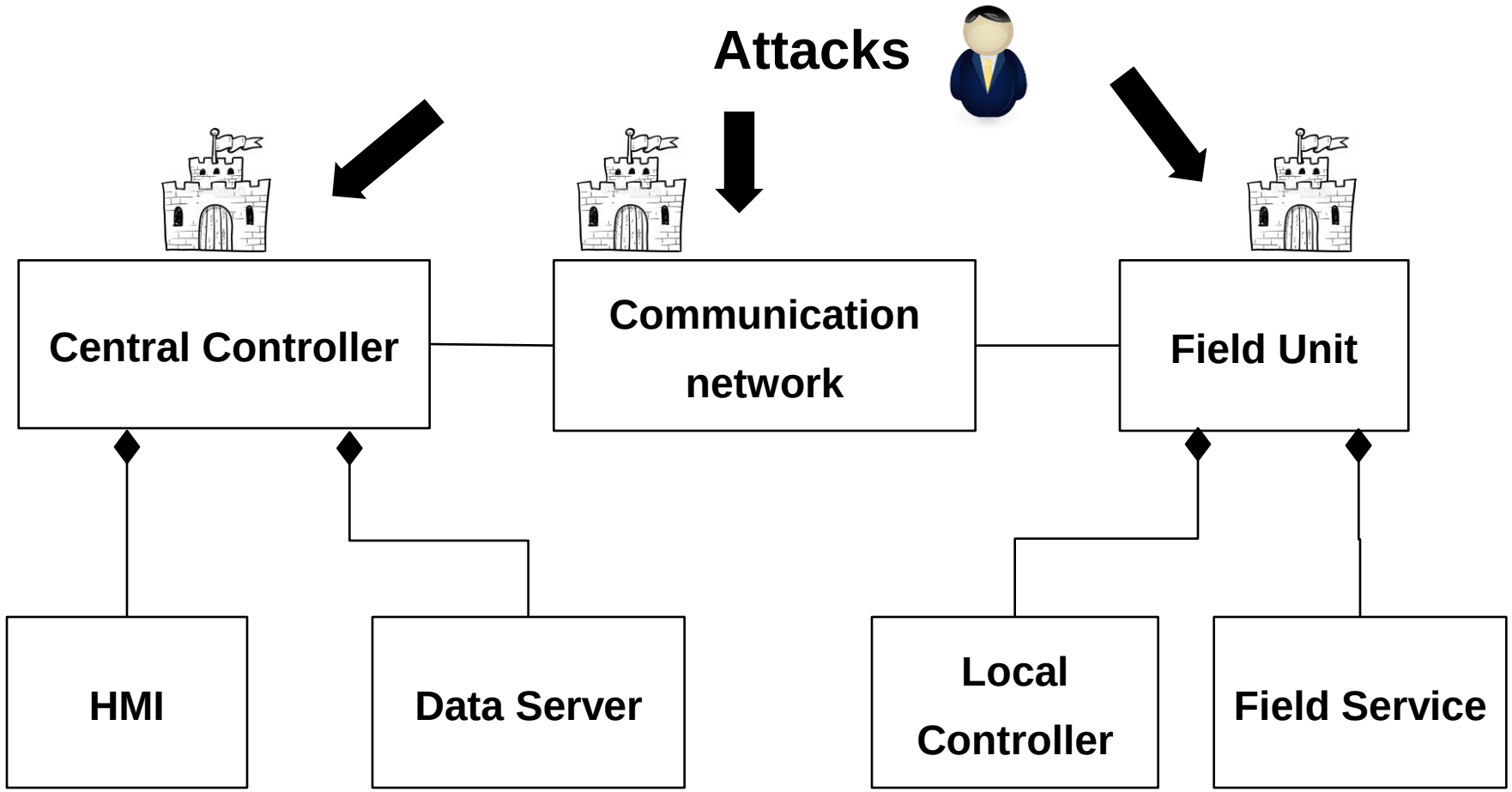
D'après [Fernandez, Larrondo-Petrie, 2010]



Class diagram for a general SCADA System



Class diagram for a general SCADA System



Security countermeasures : Identification and Authentication, Access Control and Authorization, Logging, Cryptography, Intrusion Detection, …

Security Patterns

Solution générale réutilisable pour un problème connu de sécurité

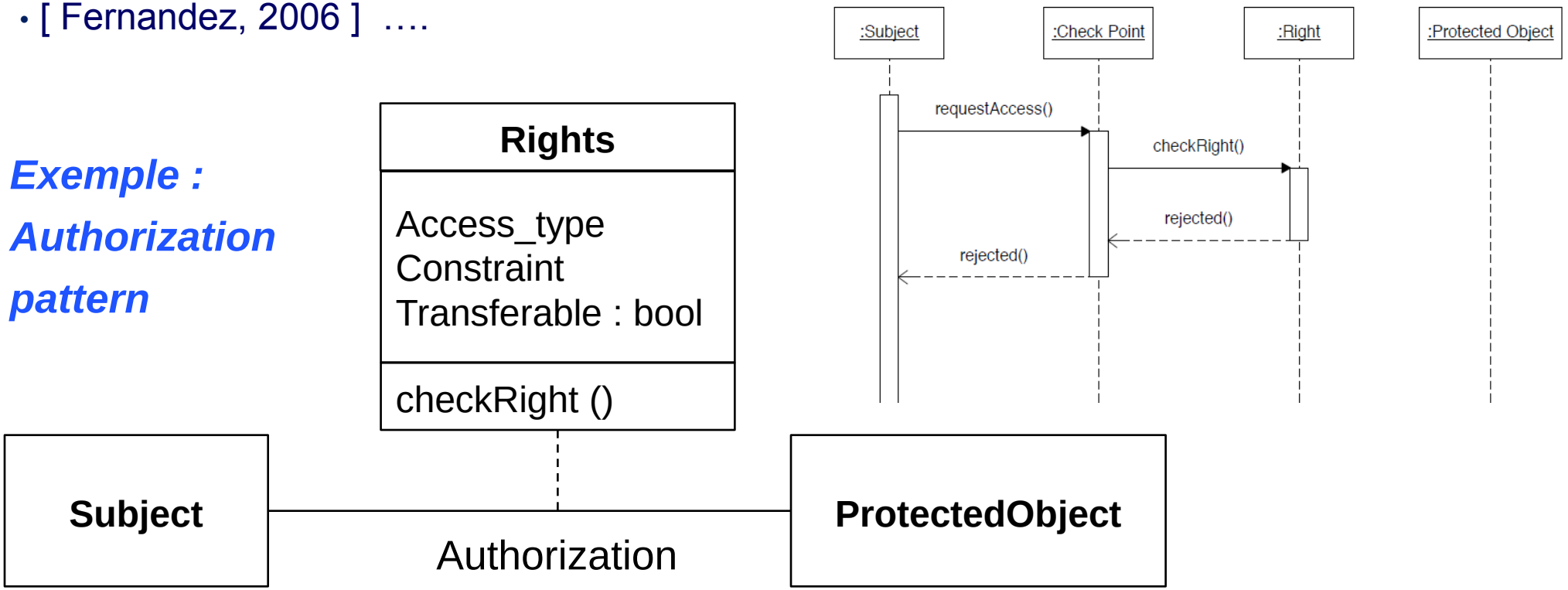
- [Yoder & Barcalow, Proc of 4th Pattern Language of programs, 1997]
- [Schumacher, Roedig, 2001]
- [Schumacher, Fernandez, Hybertson, Buschmann. Wiley & Sons, 2005]
- [Fernandez, 2006] ….

Security Patterns

Solution général réutilisable pour un problème connu de sécurité

- [Yoder & Barcalow, Proc of 4th Pattern Language of programs, 1997]
- [Schumacher, Roedig, 2001]
- [Schumacher, Fernandez, Hybertson, Buschmann. Wiley & Sons, 2005]
- [Fernandez, 2006] ….

Exemple : Authorization pattern



Authorization pattern

Description détaillée du pattern

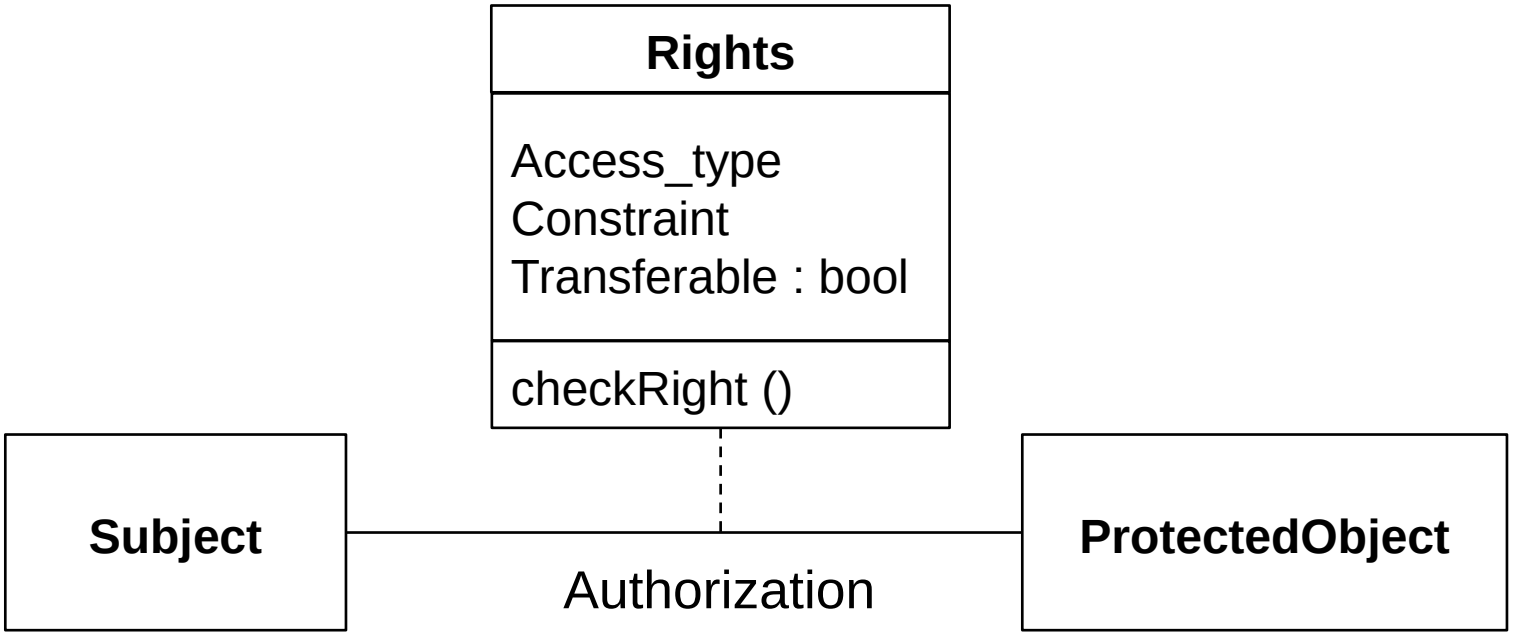
Intention :
Vérification d'autorisation au niveau requêtes de services indépendamment de l'authentification

Vérification au niveau fonctions ou lien entre entités.

Authorization pattern

Description détaillée du pattern

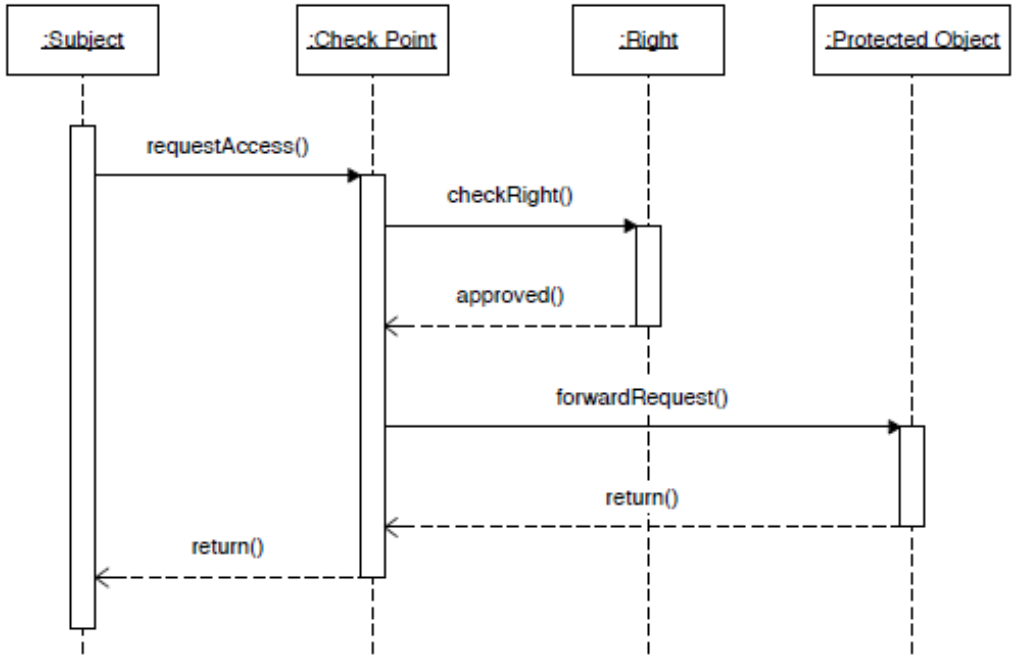
Structure abstraite du pattern :



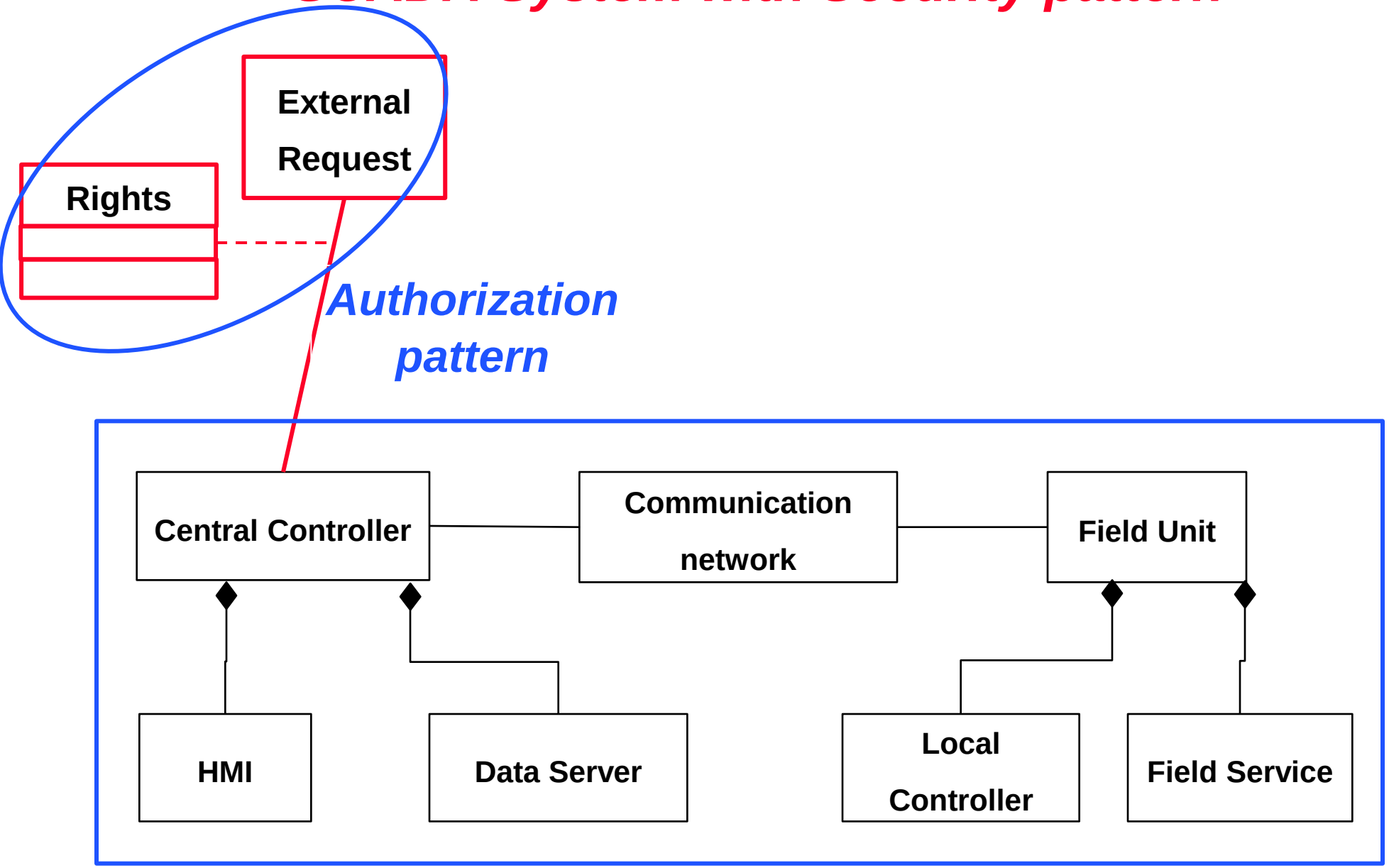
Authorization pattern

Description détaillée du pattern

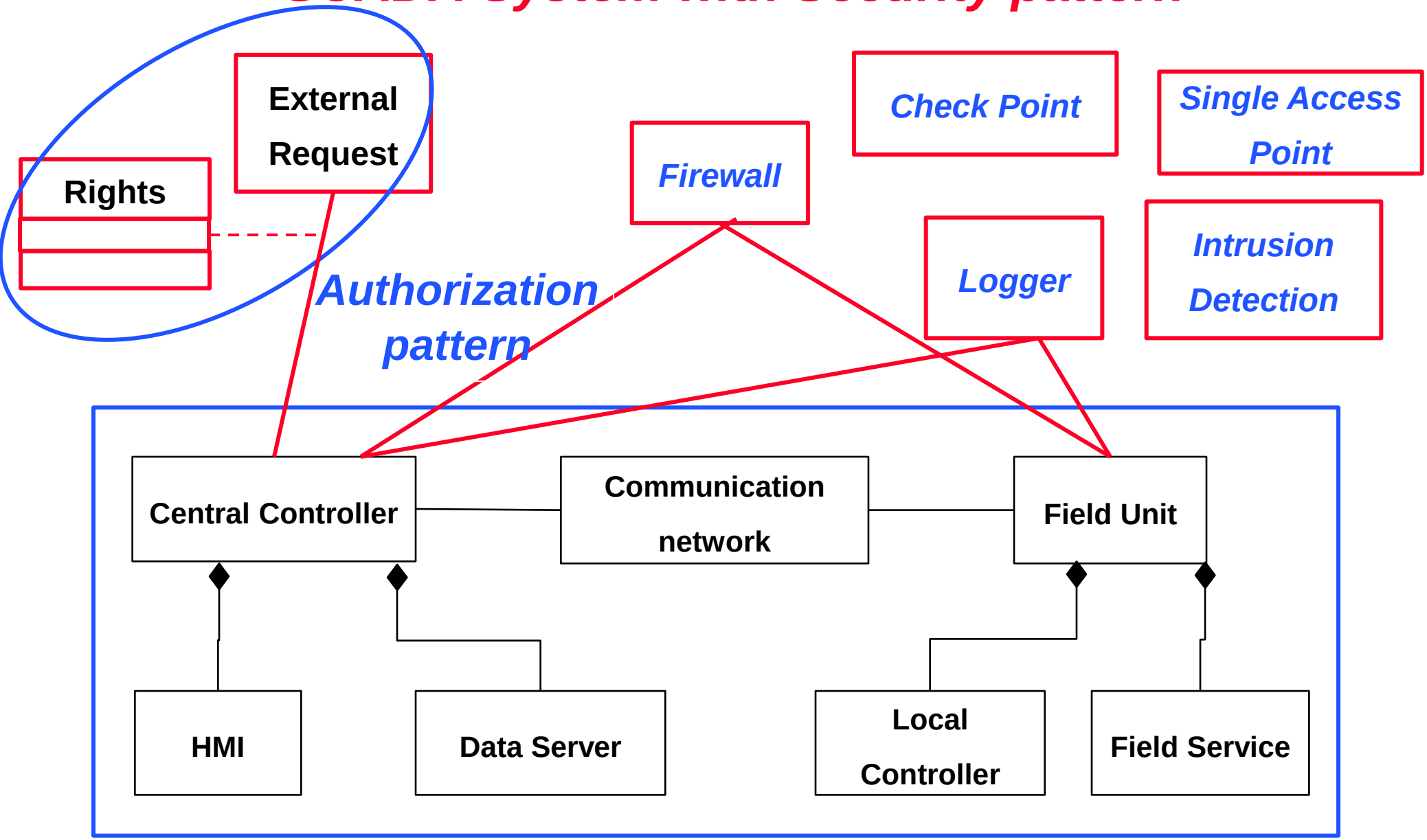
Comportement du pattern avec introduction du CheckPoint Pattern



SCADA System with Security pattern



SCADA System with Security pattern



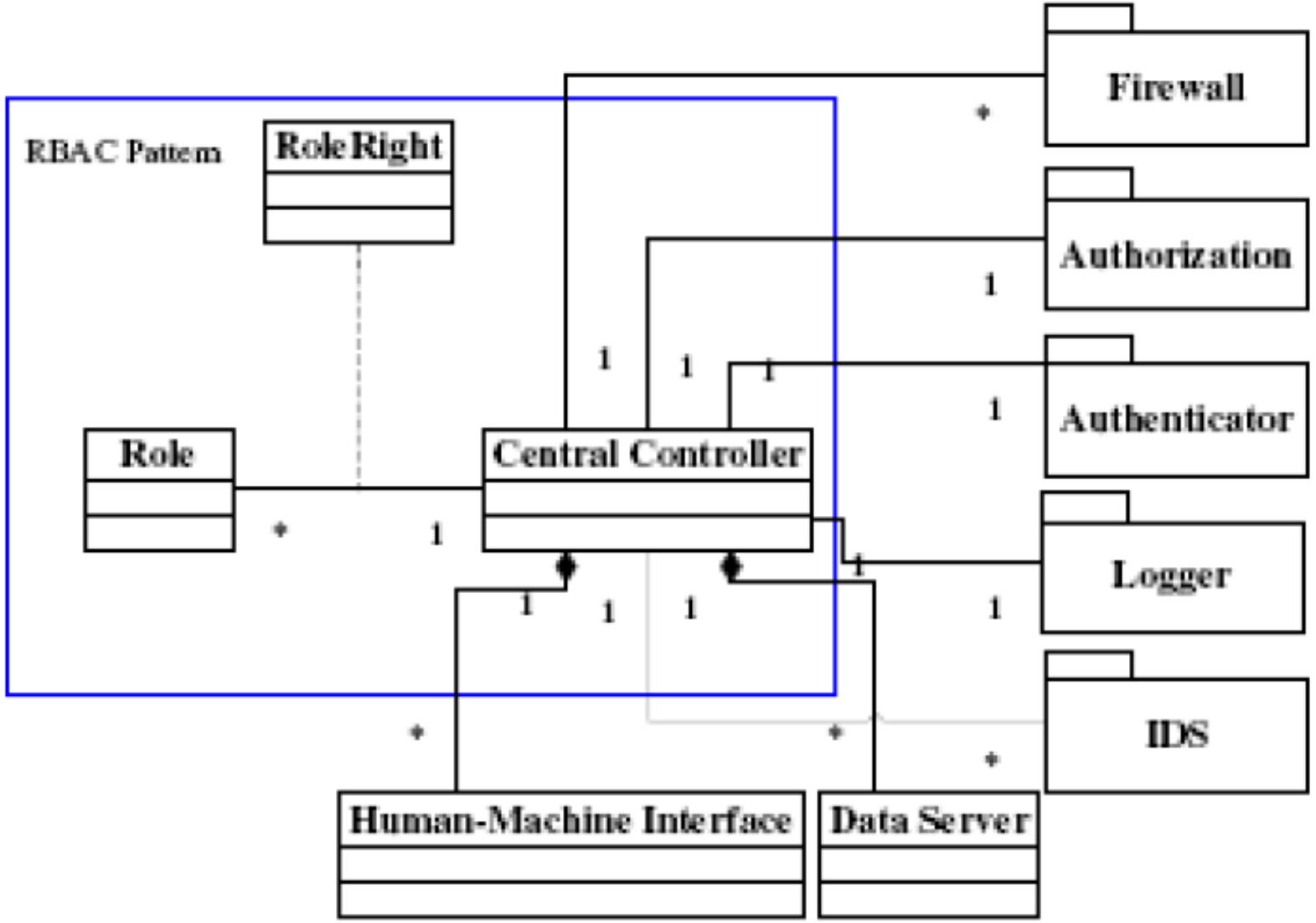
Challenges

- Combinaison de patterns
  - Complexité, impact sur les comportements et les propriétés, …
  - Introduction de comportements incompatibles => failles de sécurité

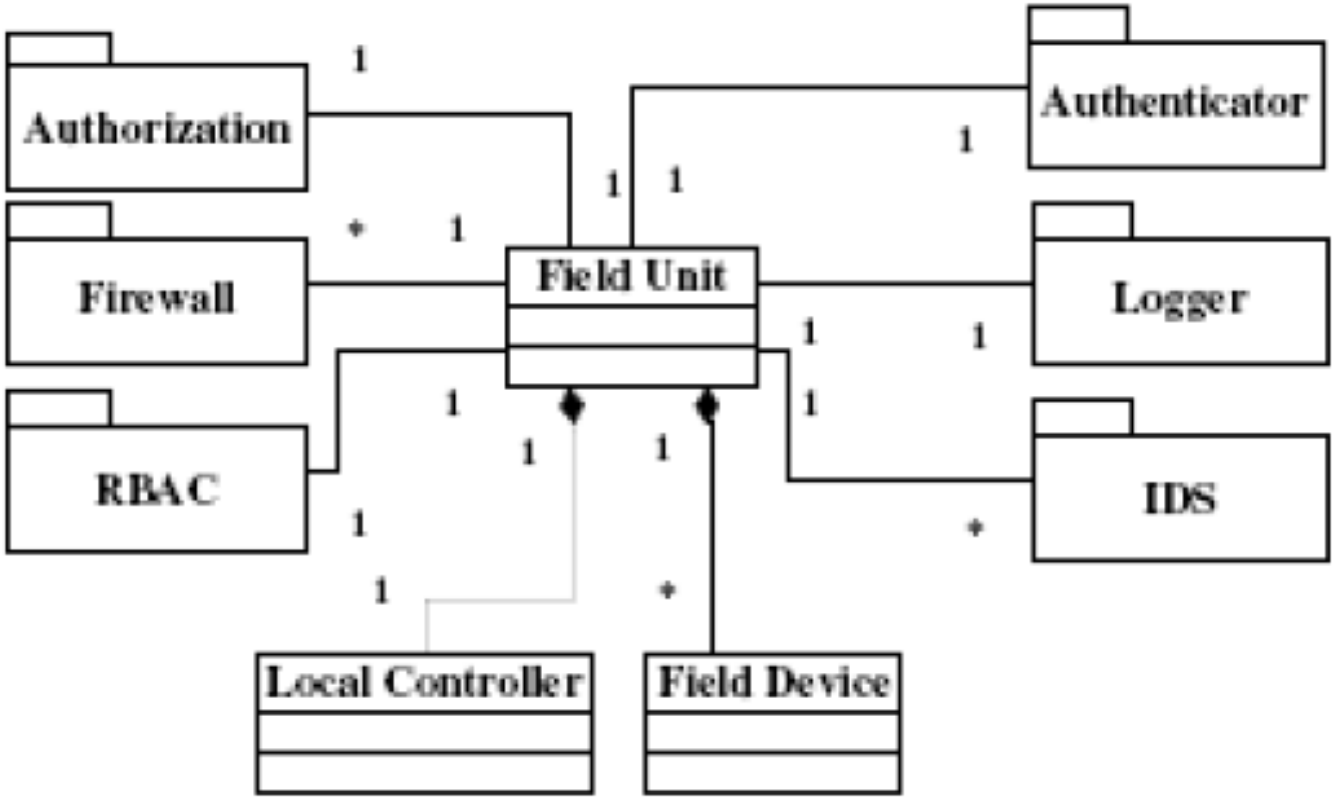
- Maintenance difficile des patterns dans les grands systèmes
  - Reconnaissance, identification des patterns dans le code et les modèles

- Garantir que les exigences sont préservées
  - Préservation des propriétés, des contraintes :
    Fonctionnelles, Sûreté de fonctionnement, Performance, …
  - Méthodologie de conception basée sur les patterns
  - Vérification formelle des propriétés (model-checking)

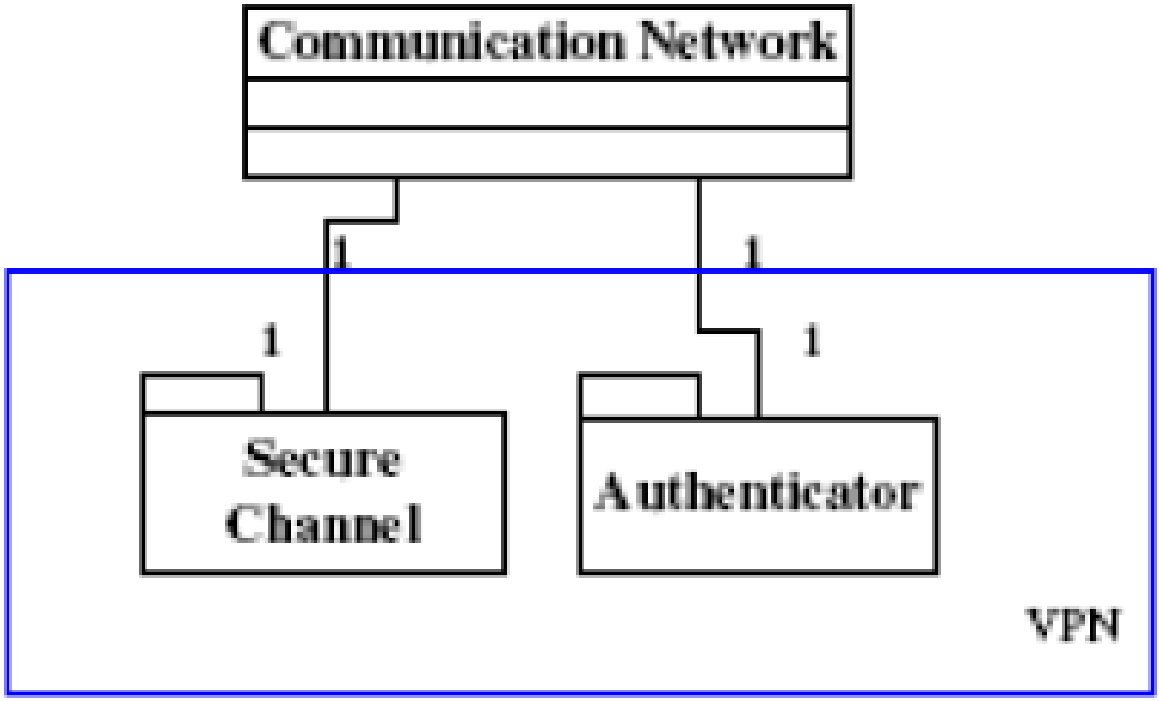
Secure central controller with security patterns



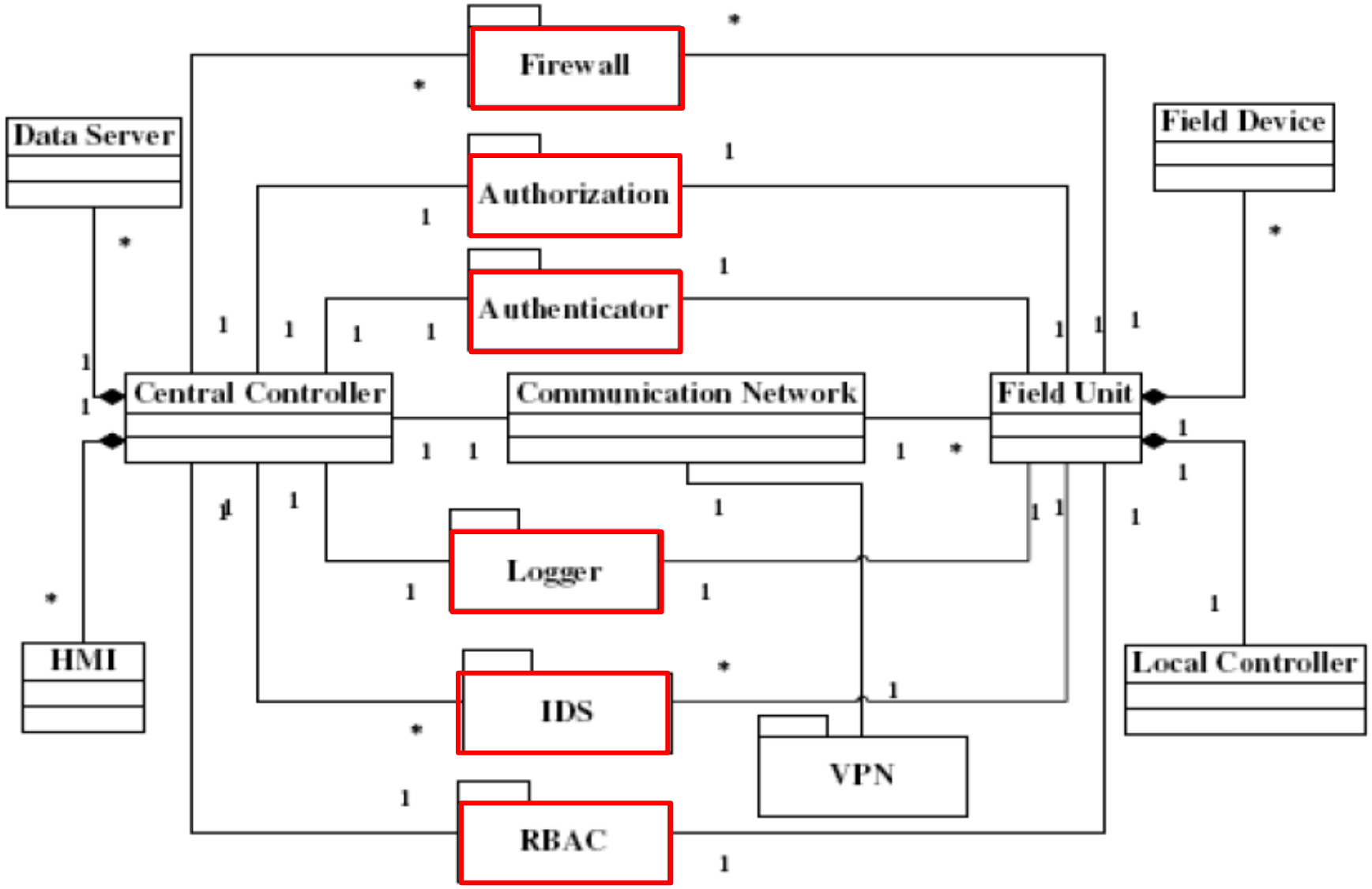
Secure field unit with security patterns



Secure communication networks with security patterns

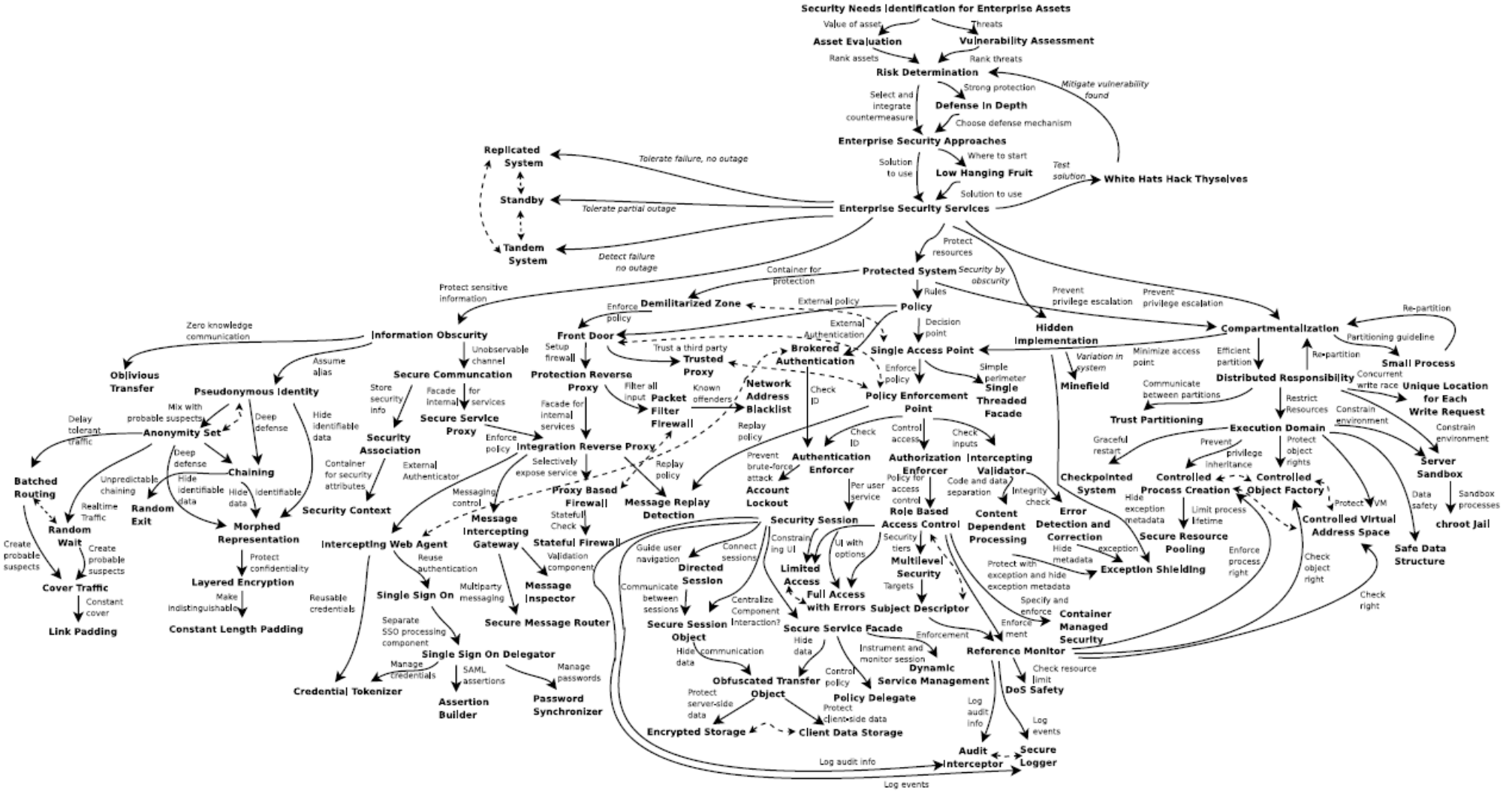


Secure SCADA with security patterns



Security patterns and pattern language

[Hafiz & al, 2012]



Modélisation, simulation et vérification formelle de patterns de sécurité Application à la cyber-défense

- Politiques et patterns de sécurité
- Patterns de sécurité et SCADA
- Modélisation et simulation UML
- Mise en œuvre de la vérification formelle de propriétés
- Travail à réaliser

Modélisation et simulation

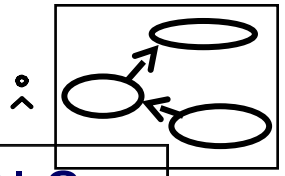
Modélisation UML :

- Exploiter les différentes vues sur le système (pattern)
- Modularité de la modélisation
- Pas de formalisme spécifique pour les propriétés
- Structure et comportement

Simulation UML

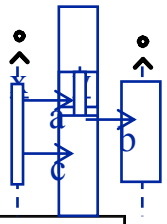
- Simulation dirigée par l'utilisateur
- Simulation à partir de scénarii de référence
- Approche par tests successifs

Modélisation avec UML



Le QUOI ?

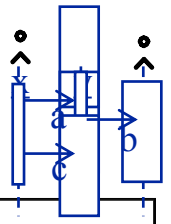
Use Case + autre



Trace

Séquence + autre

Comparaison automatique ou manuelle



Analyse

Séquence + autre

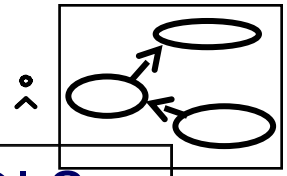
Le comportement

Opérations, Statecharts + autre

La structure

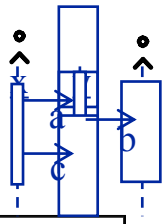
Classe, Composant + autre

Modélisation des patterns



Le QUOI ?

Use Case + autre



Trace

Séquence + autre

Comparaison automatique ou manuelle

Ajout objets des Patterns

Séquence + autre

Modification de Structure

La structure

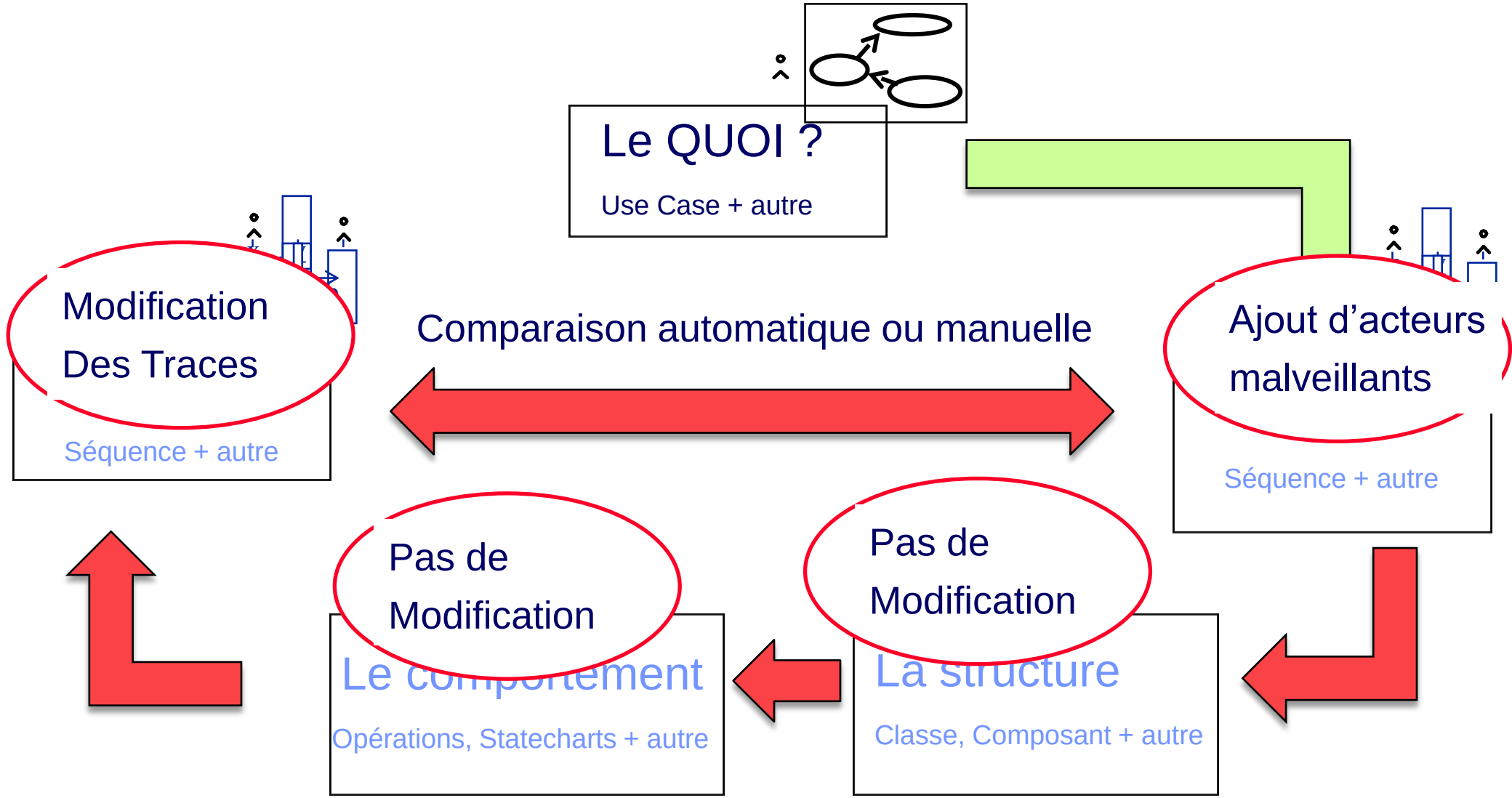
Classe, Composant + autre

Adaptation du Comportement

Le comportement

Opérations, Statecharts + autre

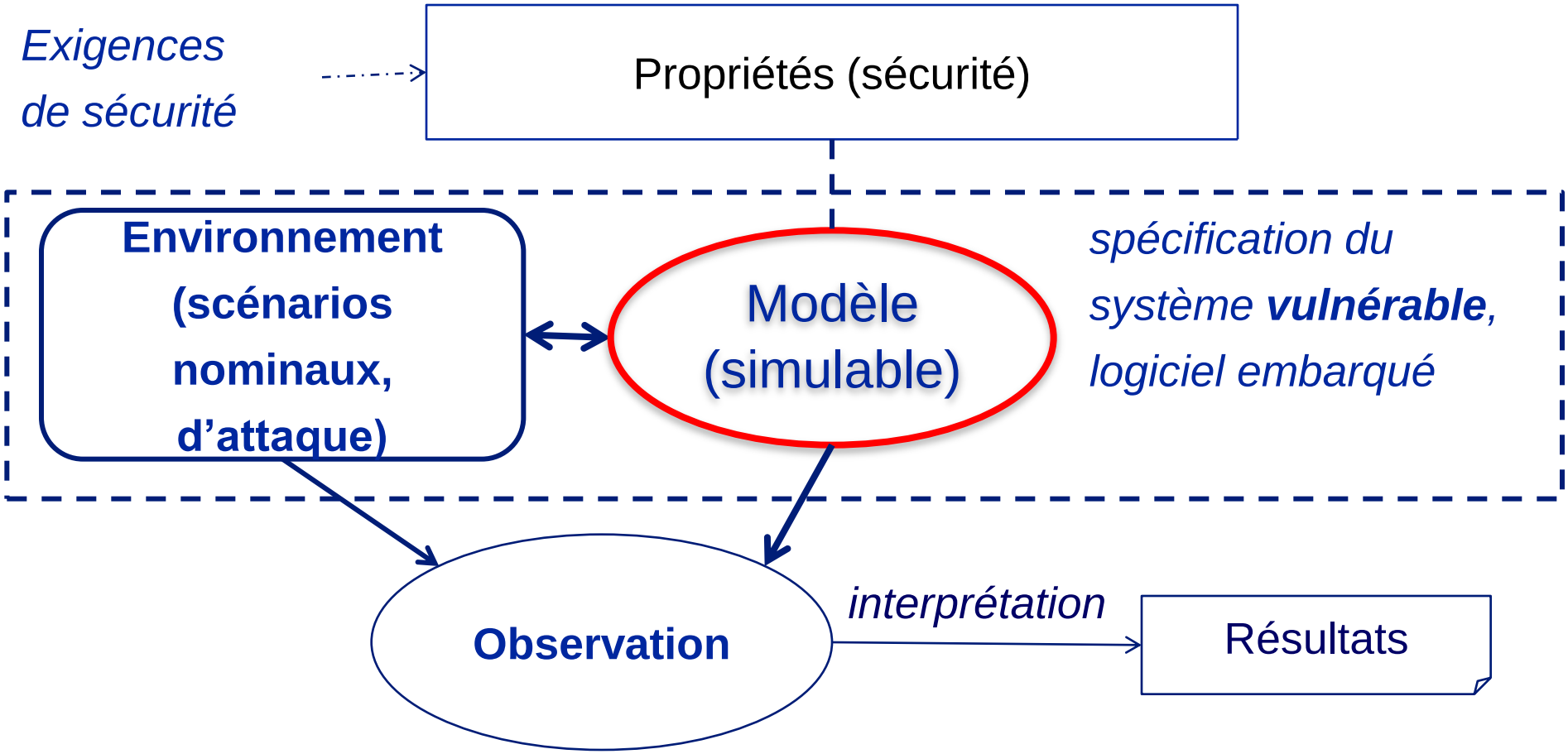
Modélisation des intrusions



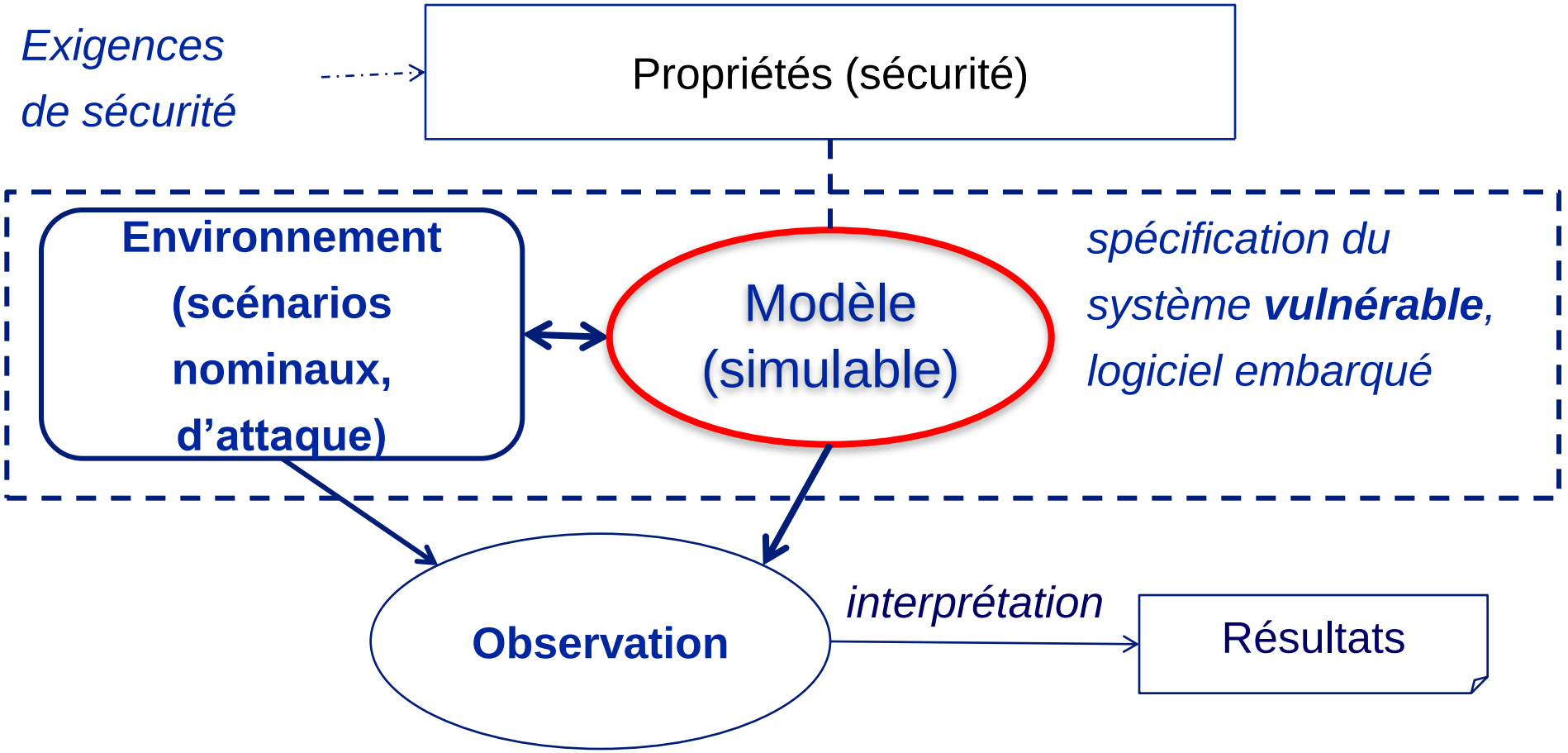
Modélisation, simulation et vérification formelle de patterns de sécurité Application à la cyber-défense

- Politiques et patterns de sécurité
- Patterns de sécurité et SCADA
- Modélisation et simulation UML
- Mise en œuvre de la vérification formelle de propriétés
- Travail à réaliser

Vers la vérification formelle de propriétés de sécurité



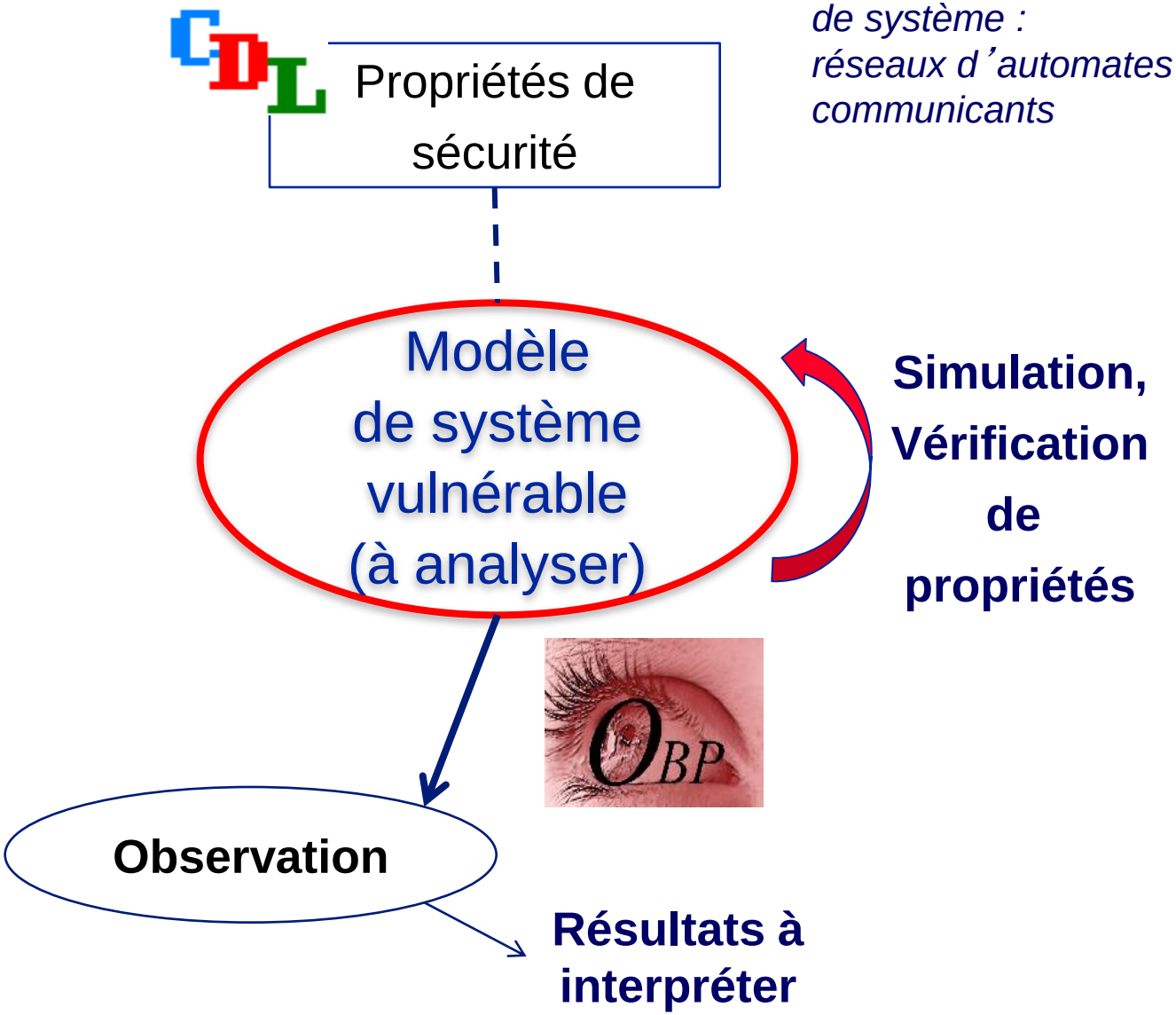
Vers la vérification formelle de propriétés de sécurité



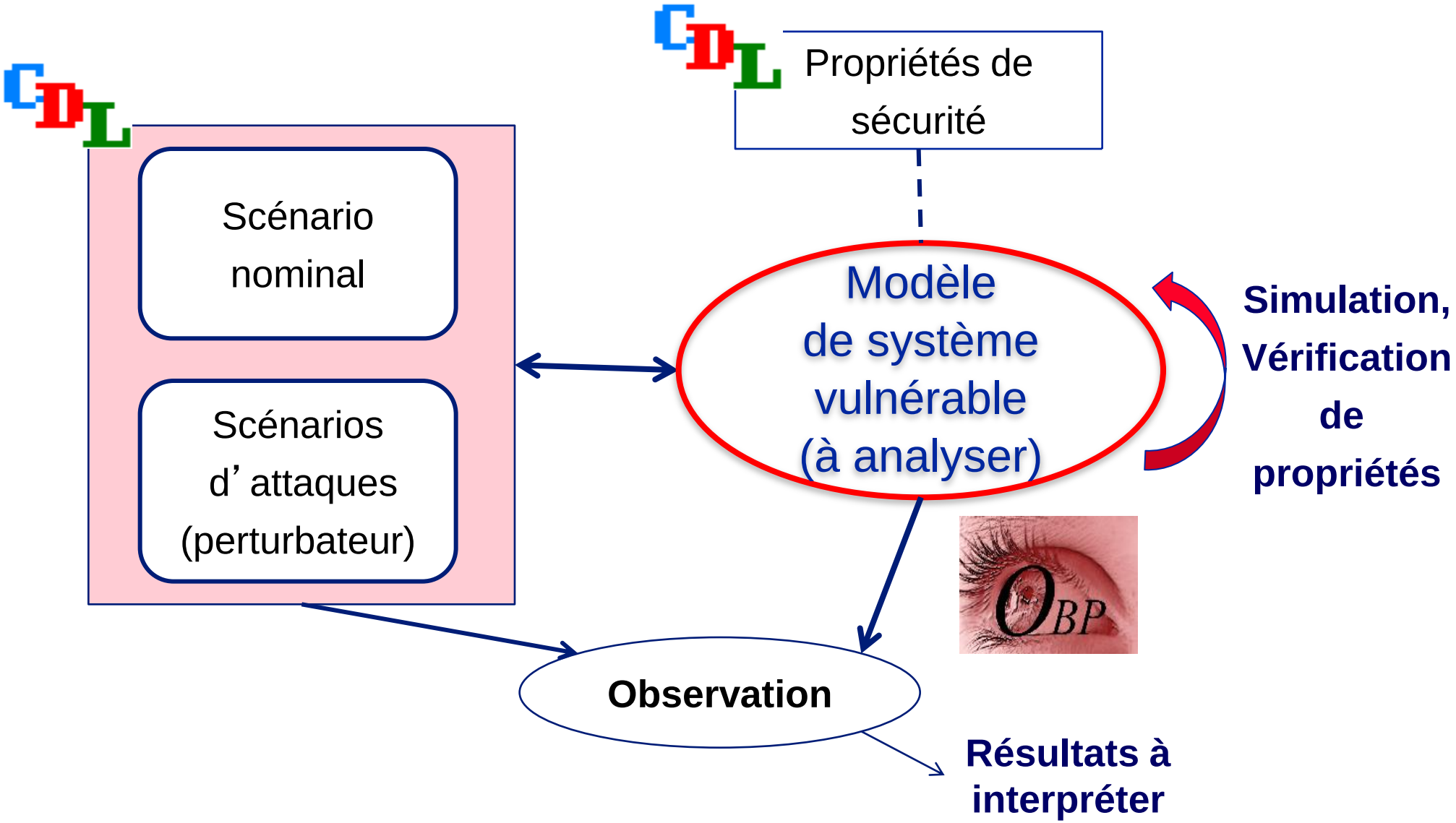
Questions scientifiques *: Formalisation des propriétés de sécurité (politique de sécurité), patrons d'architecture pour la sécurité, analyse de la robustesse, …*

Détection des attaques, analyse de la robustesse

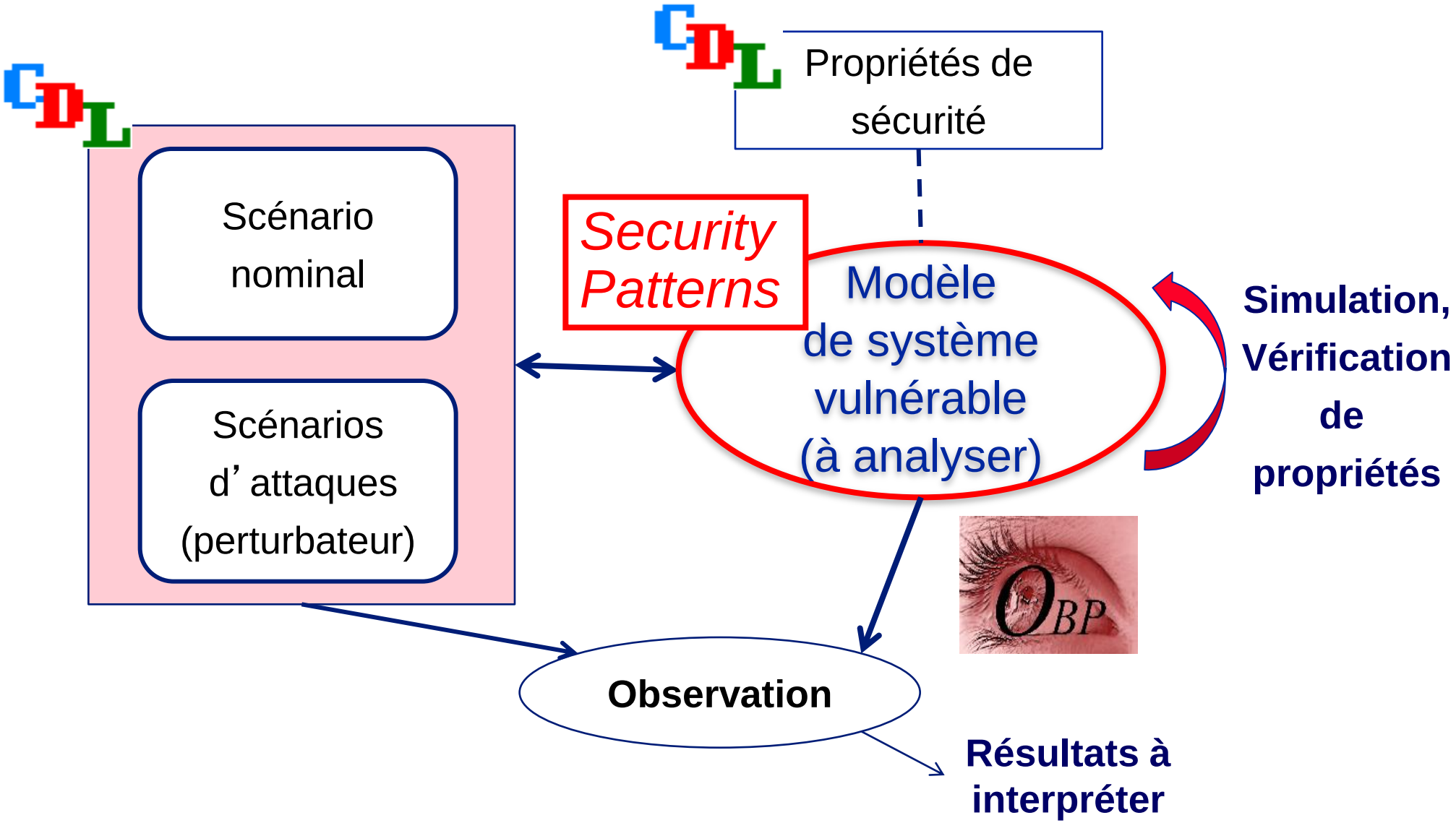
Hypothèse : *Modèle de système : réseaux d'automates communicants*



Détection des attaques, analyse de la robustesse



Détection des attaques, analyse de la robustesse



Modélisation, simulation et vérification formelle de patterns de sécurité Application à la cyber-défense

- Politiques et patterns de sécurité
- Patterns de sécurité et SCADA
- Mise en œuvre de la vérification formelle de propriétés
- Modélisation et simulation UML
- Travail à réaliser

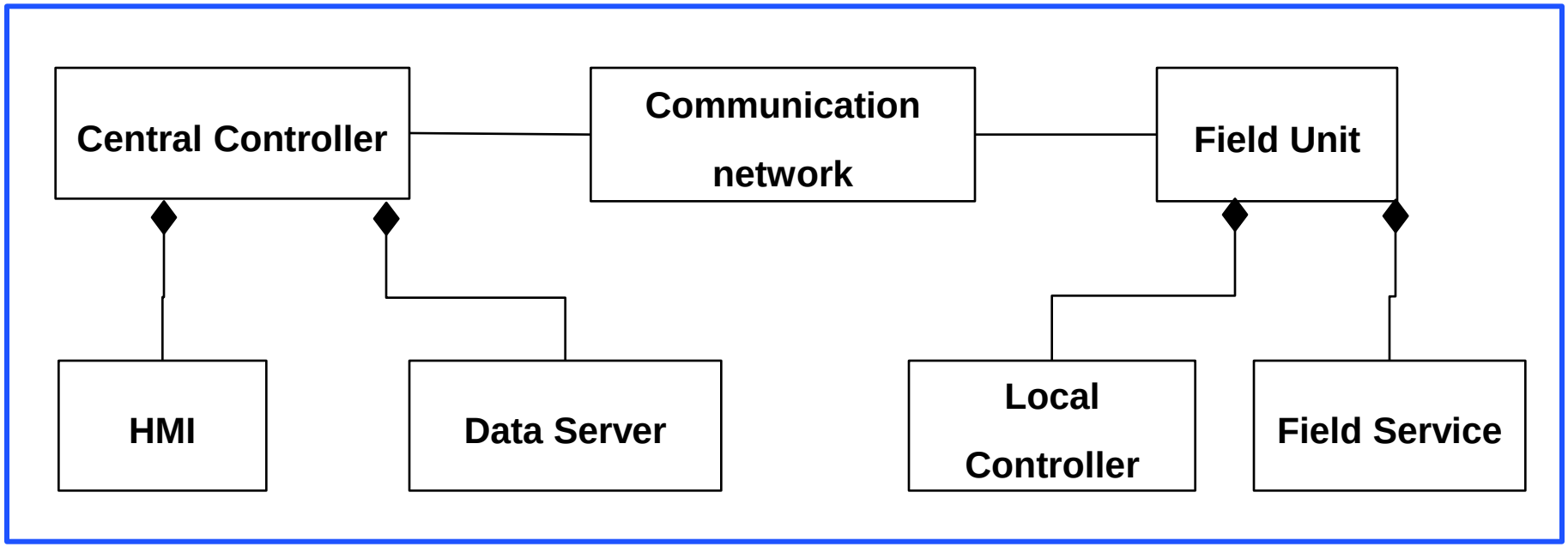
Travail à réaliser : mise en œuvre de patterns de sécurité

1. Modélisation de l'architecture abstraite SCADA

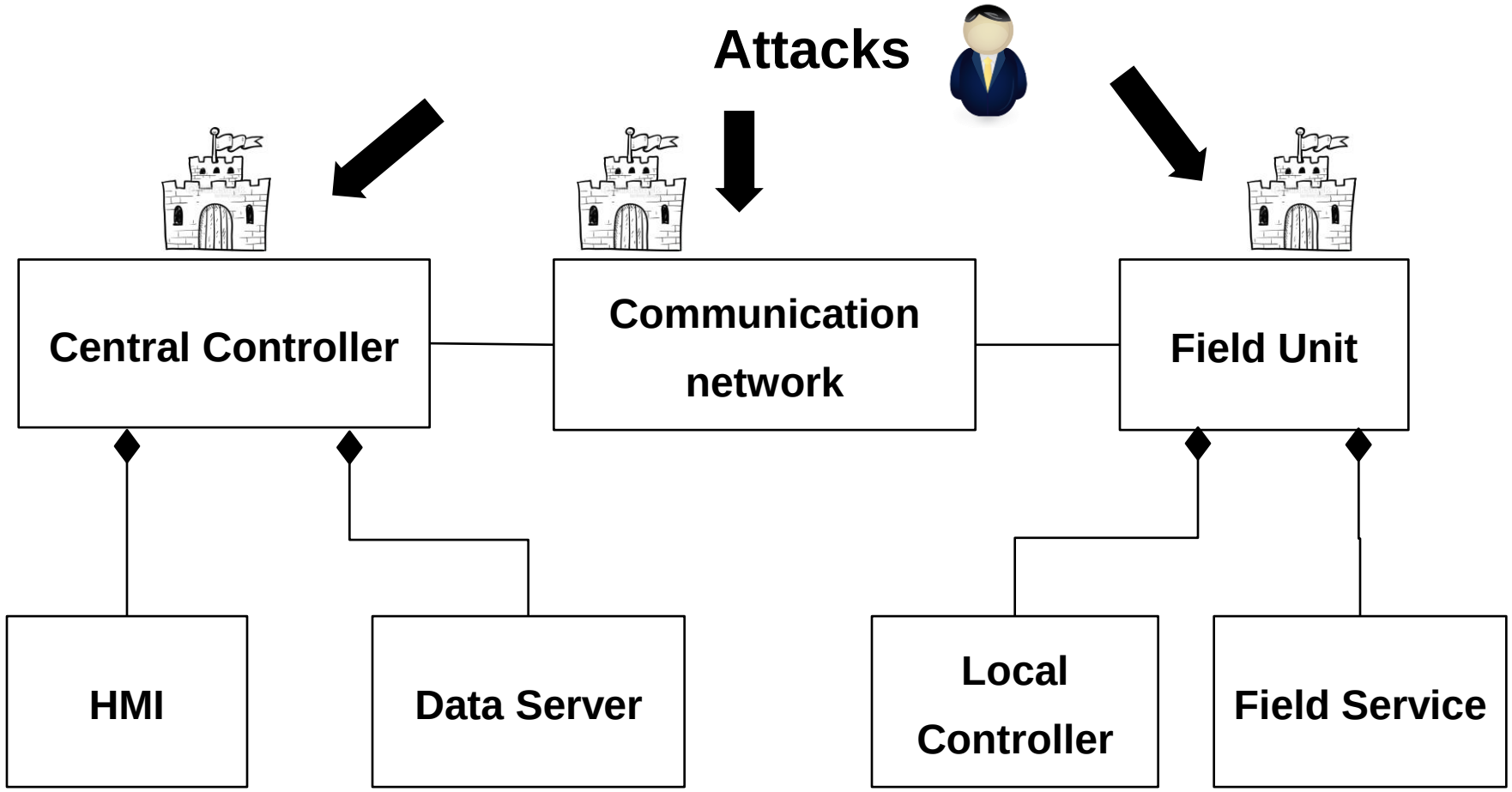
2. Composition avec 3 patterns *(Authorization, Single Access Point, Check Point)*

3. Approche générale :
   - Modélisation Rhapsody et simulation
   - Modélisation Fiacre et vérification (OBP) de propriétés (CDL)

Architecture SCADA



Système SCADA sécurisé



Objectifs du travail :

- Créer les forteresses de protection
- Exécuter le modèle
- Vérifier les propriétés

Composition de Pattern

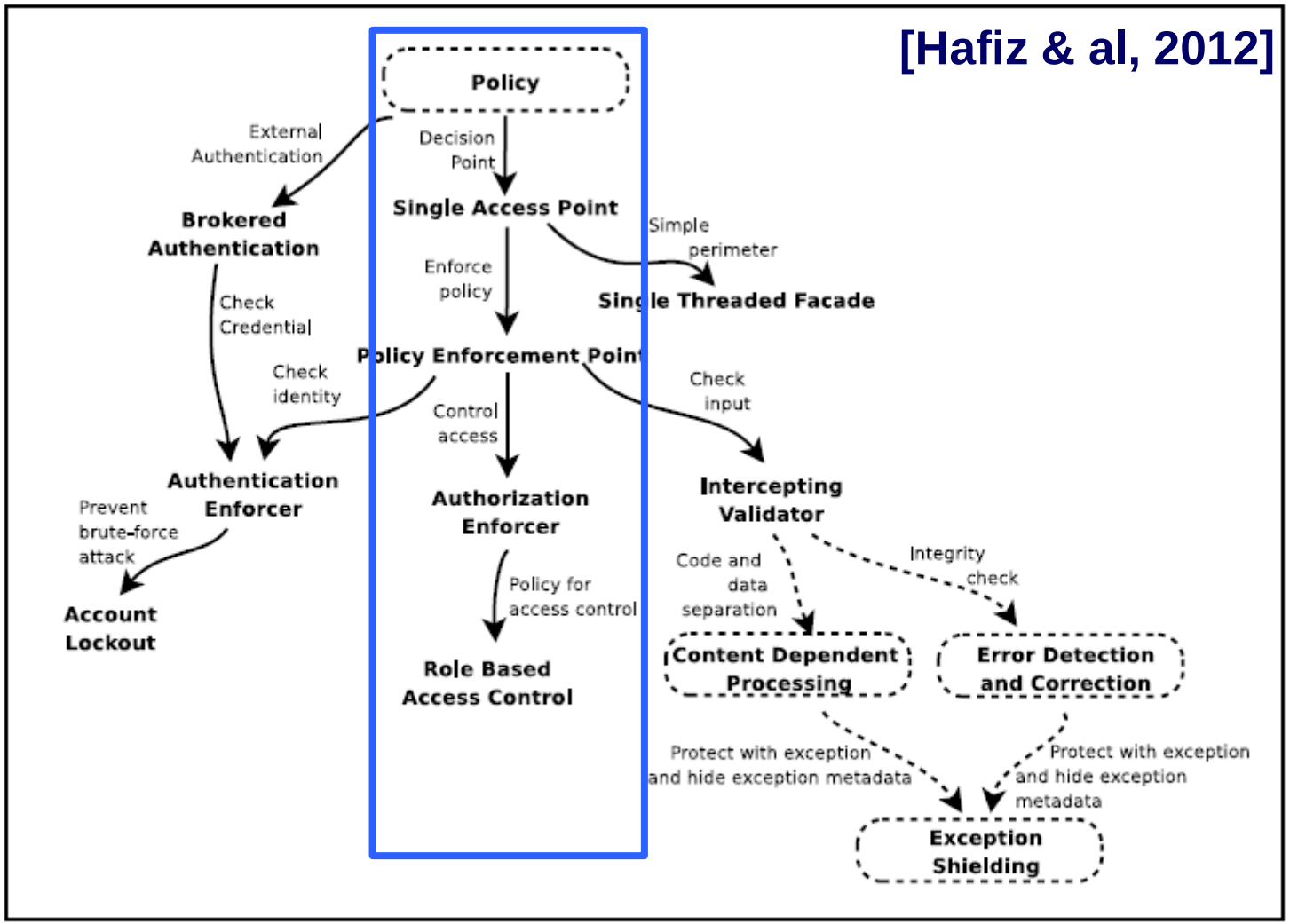
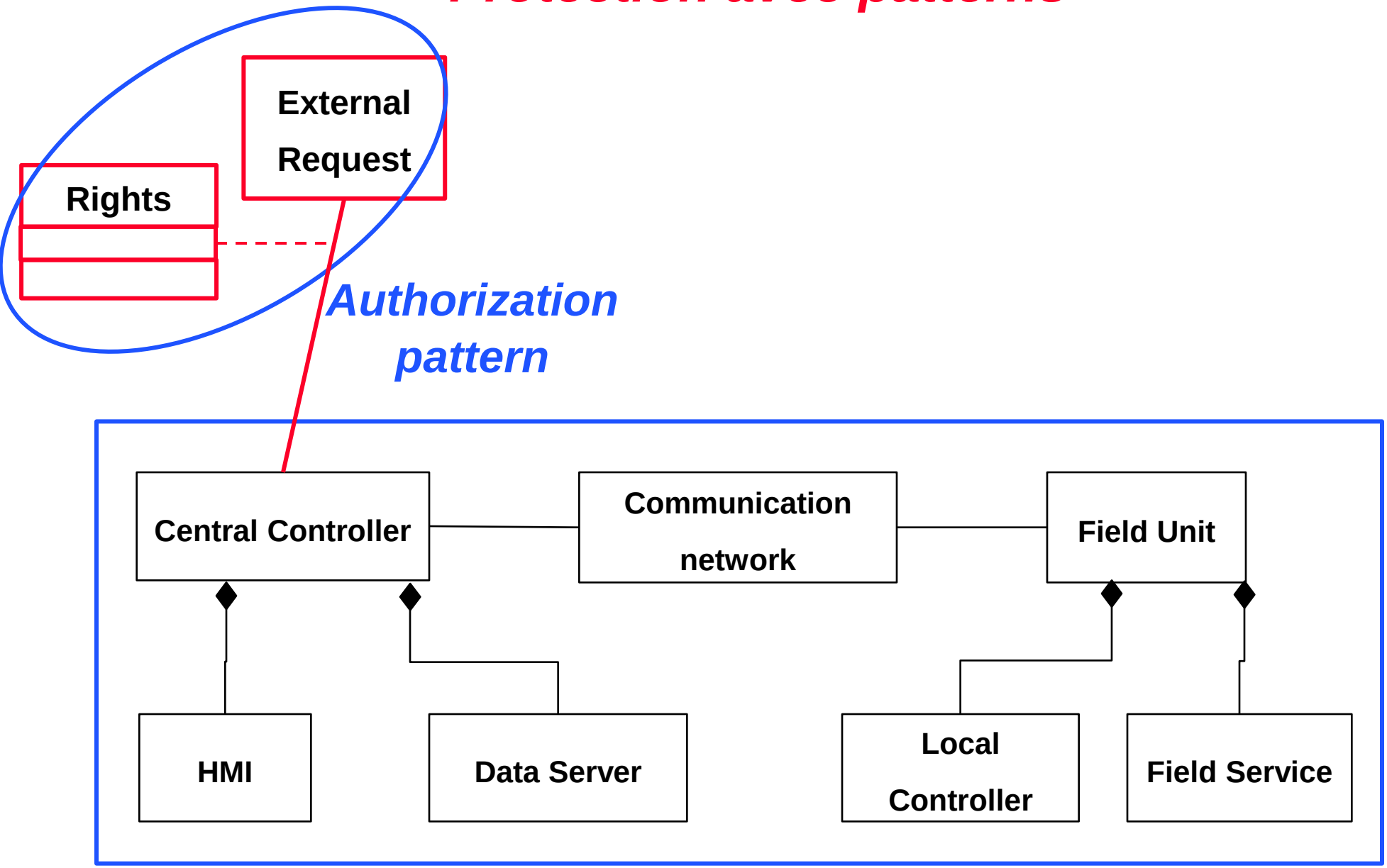
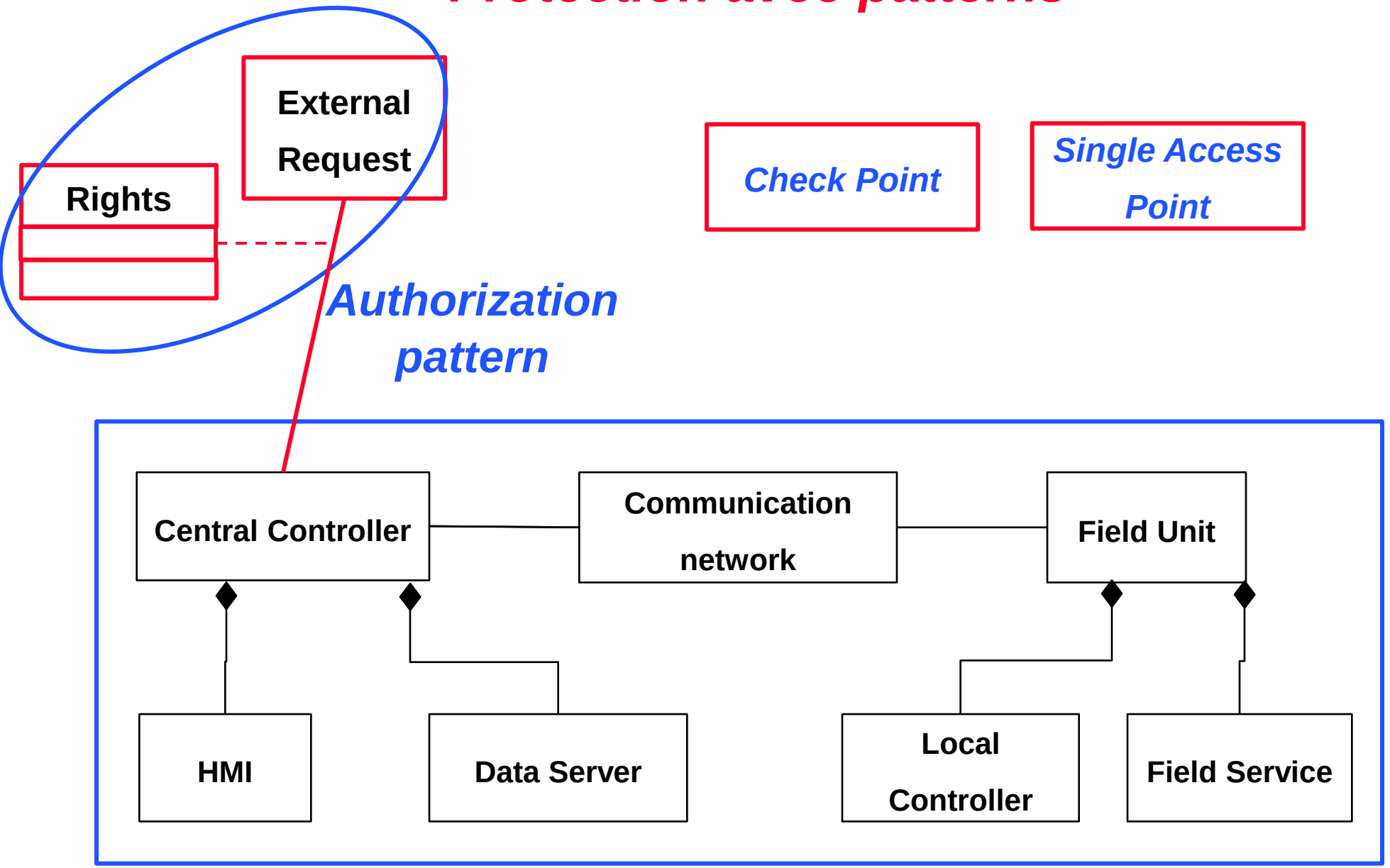


Figure 4. Pattern Language of Perimeter Patterns

Protection avec patterns



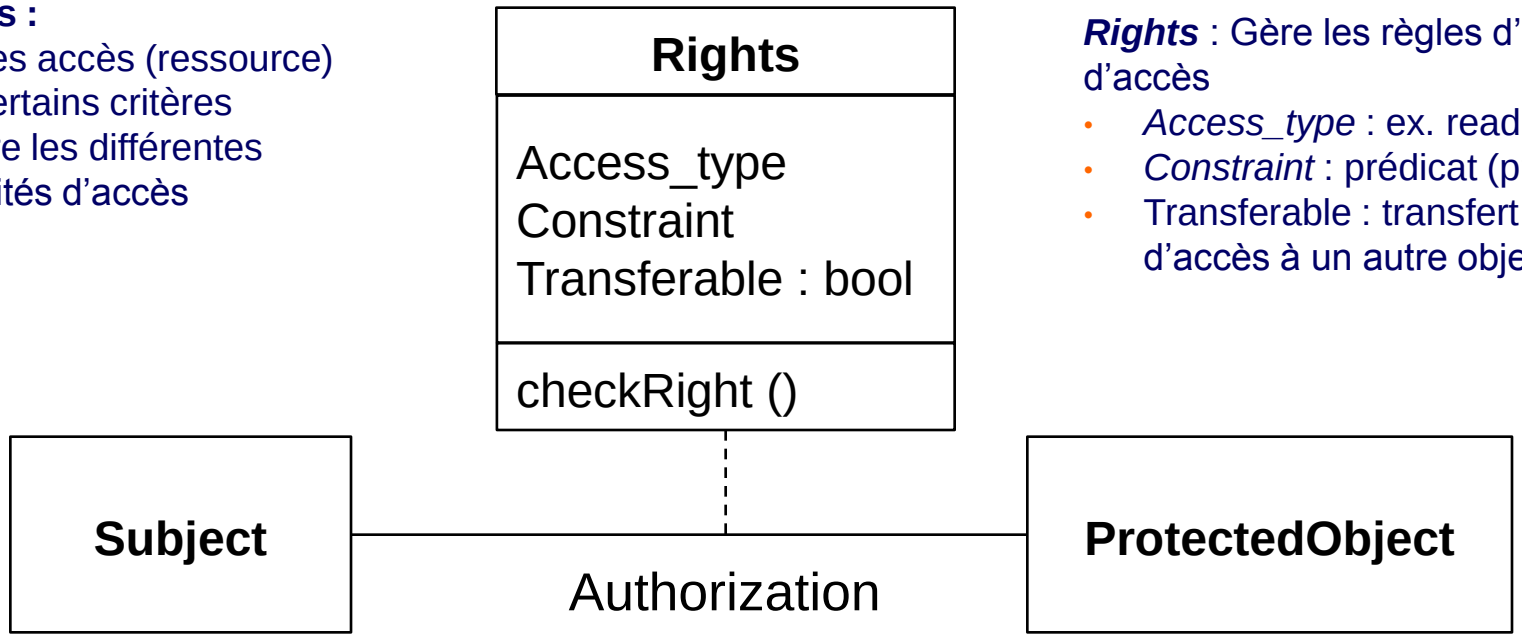
Protection avec patterns



Authorization pattern

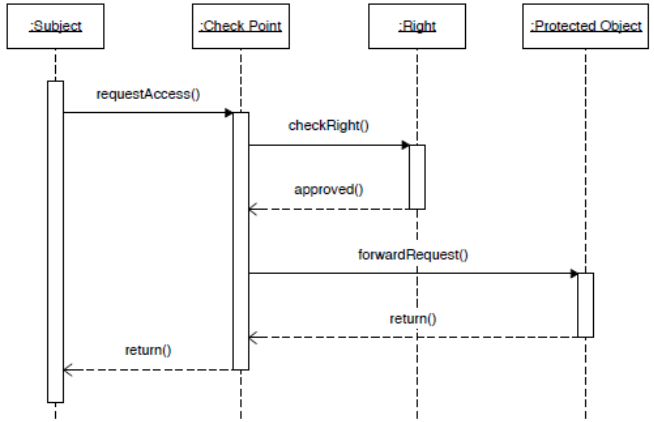
Motivations :

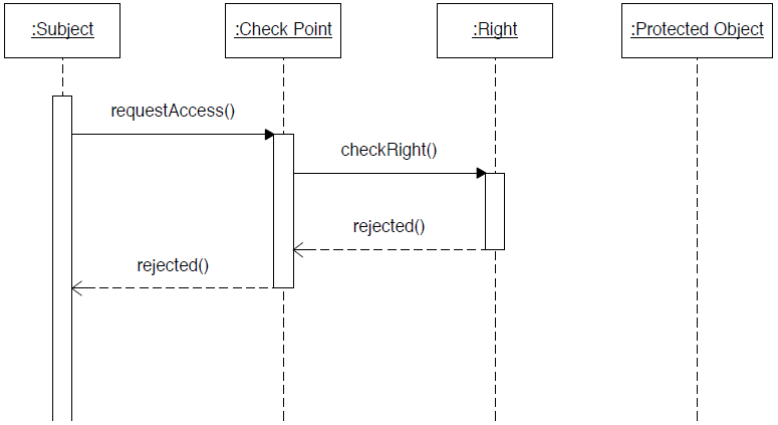
- Limite les accès (ressource) selon certains critères
- Structure les différentes possibilités d'accès



Rights : Gère les règles d'autorisation d'accès

- *Access_type* : ex. read, write
- *Constraint* : prédicat (privilège)
- Transferable : transfert du droit d'accès à un autre objet





Authorization pattern

Exemple :

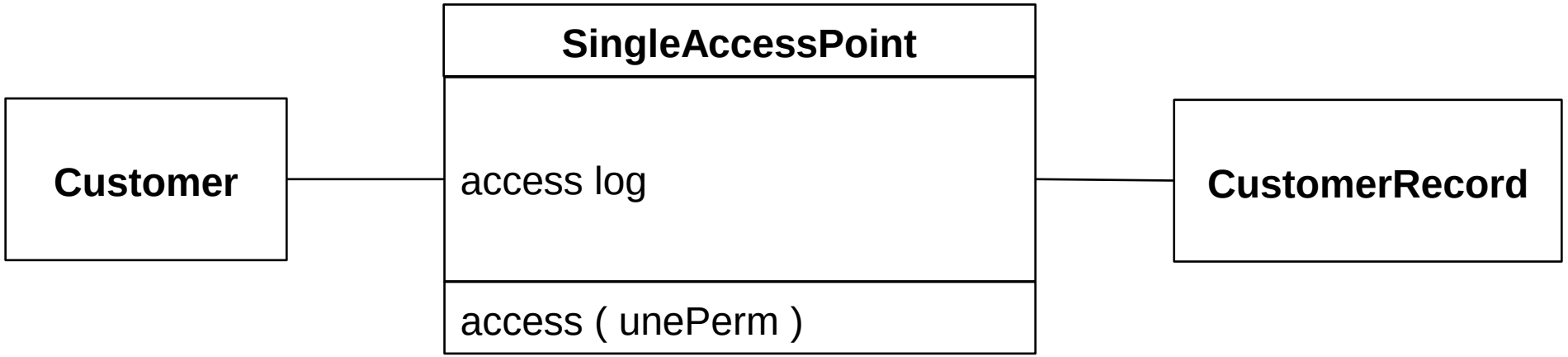
- Exception Java AccessControlException : access denied

Exception in thread "AWT-EventQueue-1" java.security.AccessControlException: access denied (java.io.FilePermission characteroutput.txt write) at java.security.AccessControlContext.checkPermission(AccessControlContext.java:323) at java.security.AccessController.checkPermission(AccessController.java:546) at java.lang.SecurityManager.checkPermission(SecurityManager.java:532) at java.lang.SecurityManager.checkWrite(SecurityManager.java:962) at java.io.FileOutputStream.<init>(FileOutputStream.java:169) at java.io.FileOutputStream.<init>(FileOutputStream.java:70) at java.io.FileWriter.<init>(FileWriter.java:46)

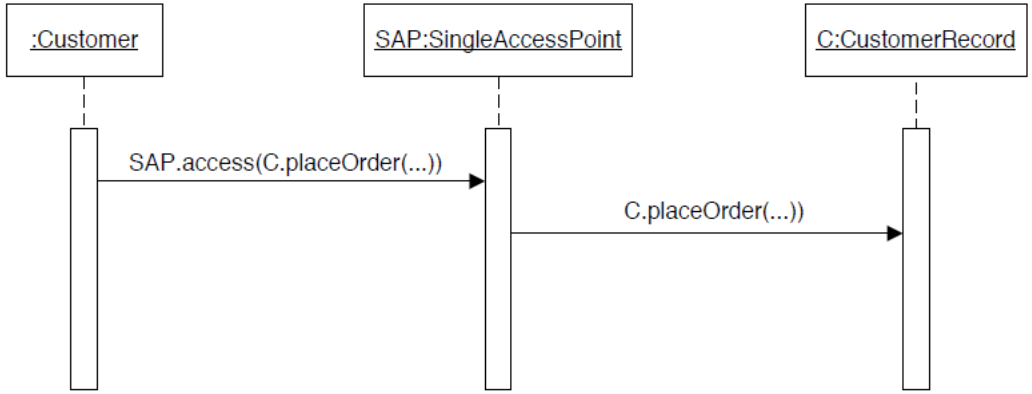
Single Access Point pattern

Motivations :

- **Limite l'accès à un seul point d'entrée**
- **Signature de chaque message qui passent pour tester la transaction au retour (proche Multilevel Security Pattern et Subject Descriptor Pattern)**
- Effectue un log.



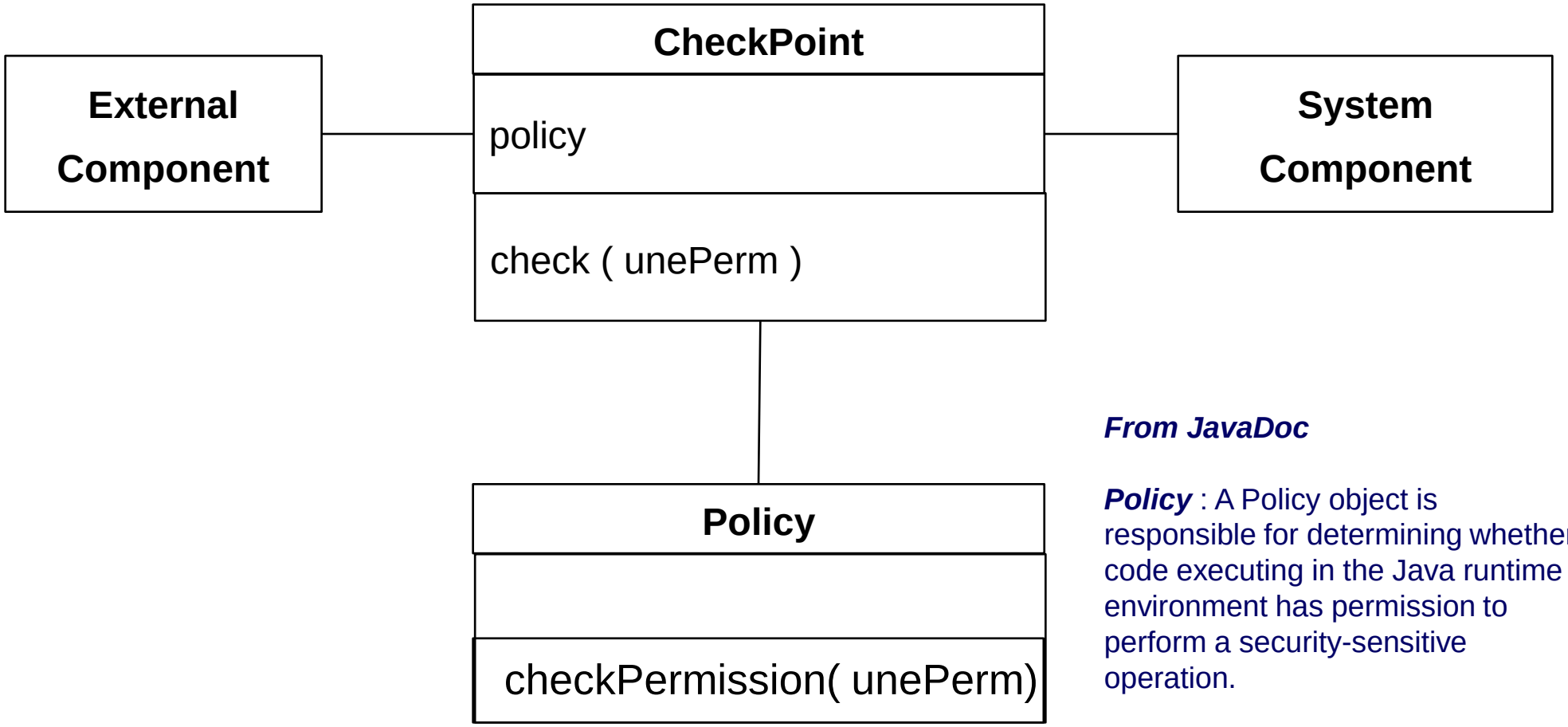
Access avec paramètres
ou
readAccess et writeAccess



Check Point pattern

Motivations :

- Contrôle le type d'accès selon une politique de sécurité
- Prend des mesures si besoin (violation)



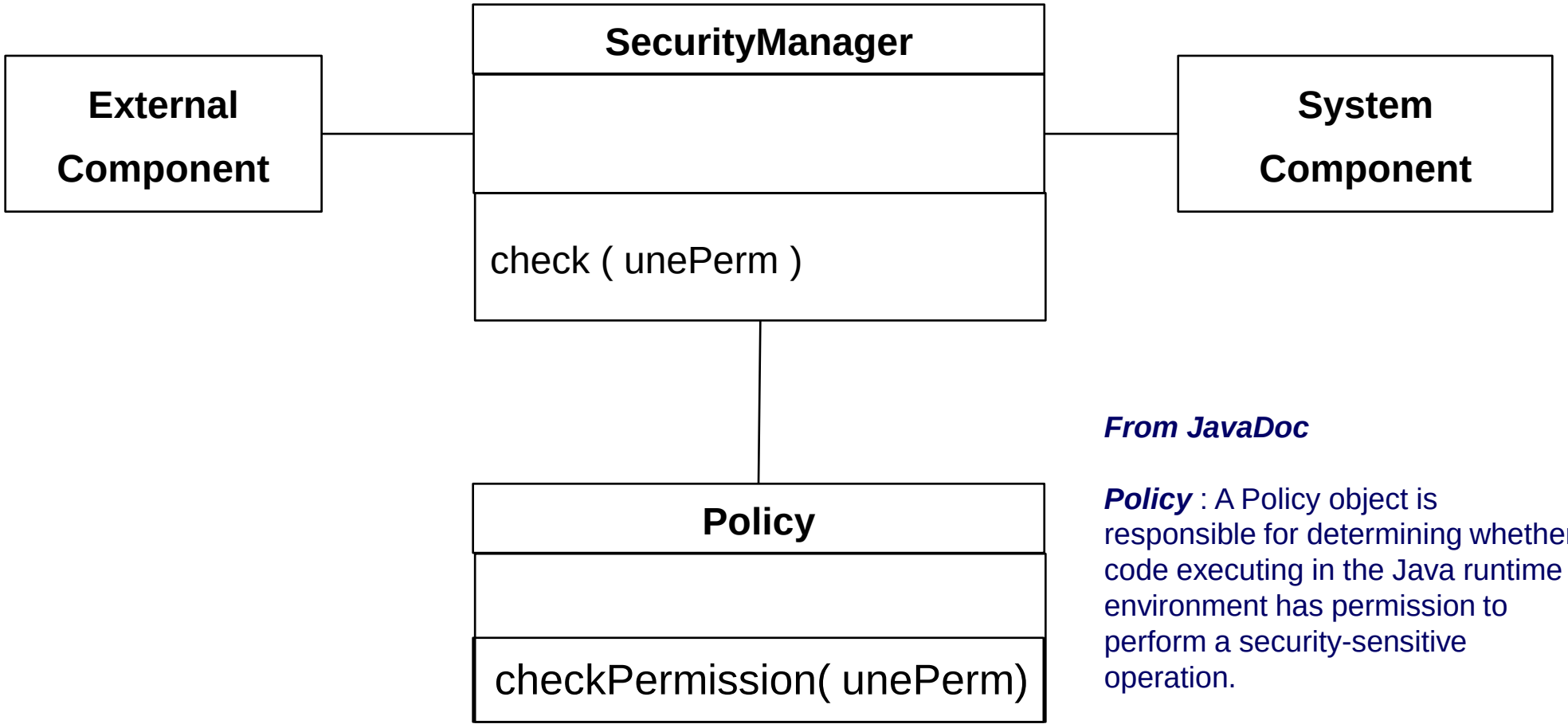
From JavaDoc

Policy : A Policy object is responsible for determining whether code executing in the Java runtime environment has permission to perform a security-sensitive operation.

Check Point pattern

Exemple :

- CheckPoint = SecurityManager in Java

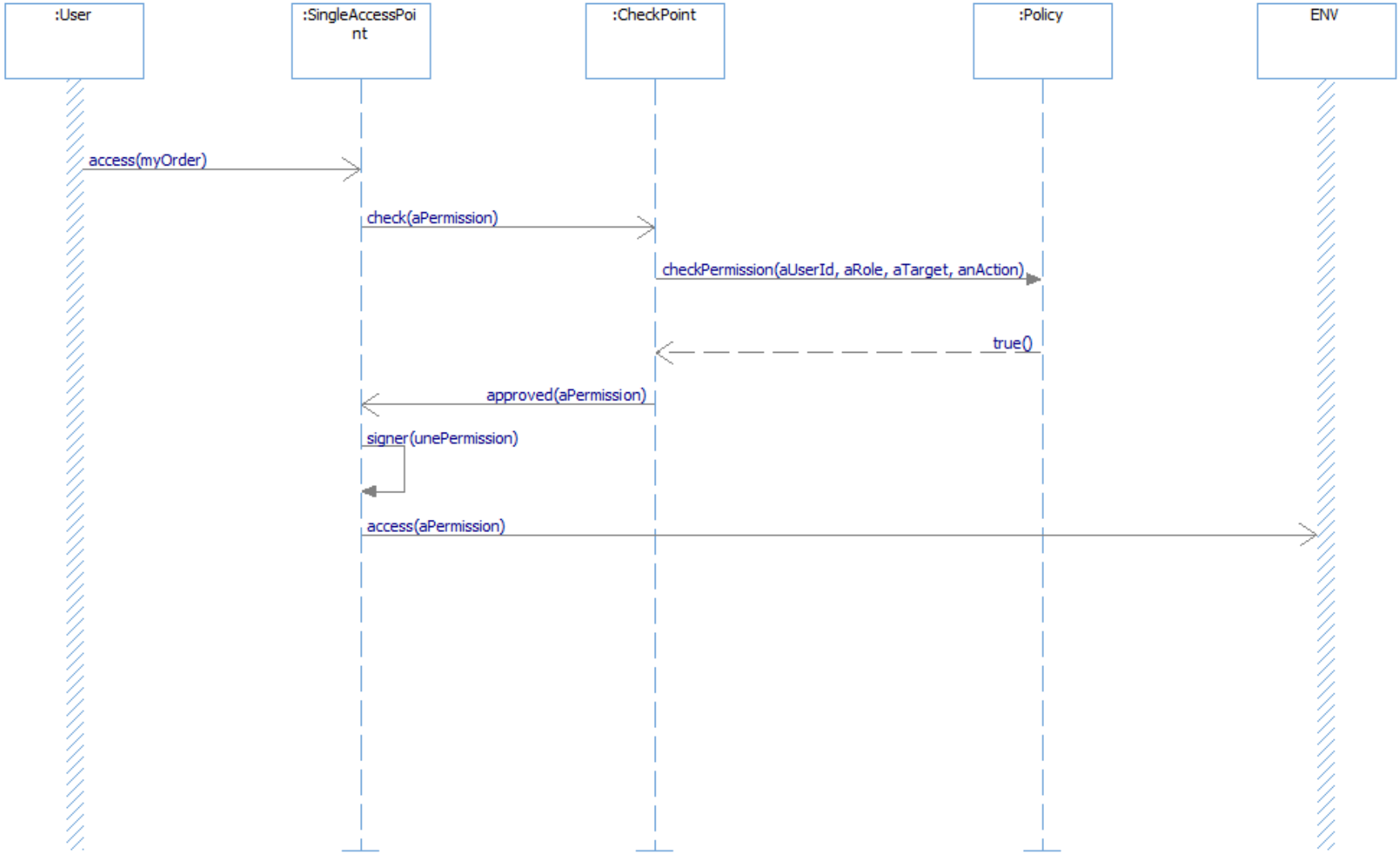


From JavaDoc

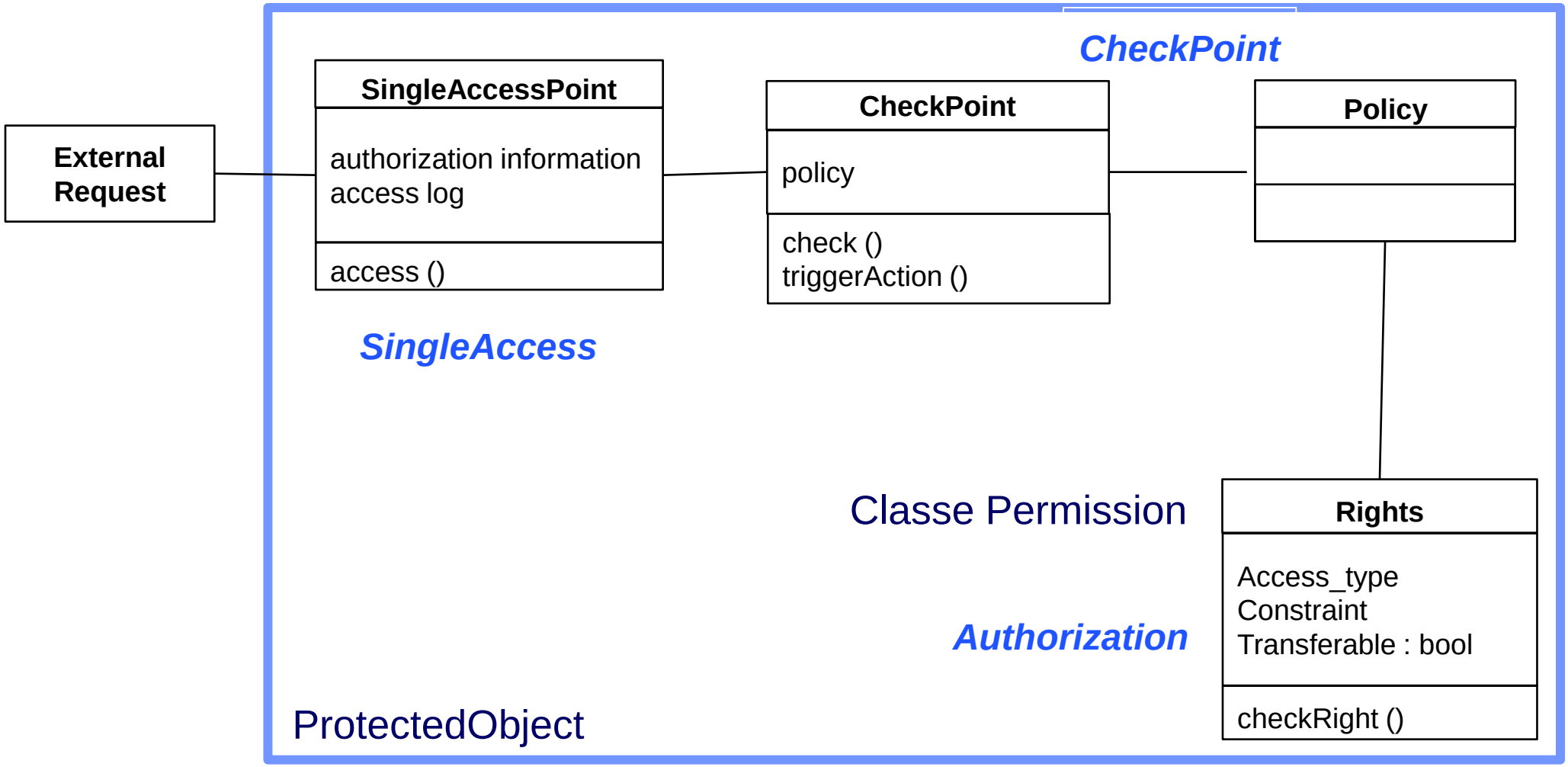
Policy : A Policy object is responsible for determining whether code executing in the Java runtime environment has permission to perform a security-sensitive operation.

Check Point pattern

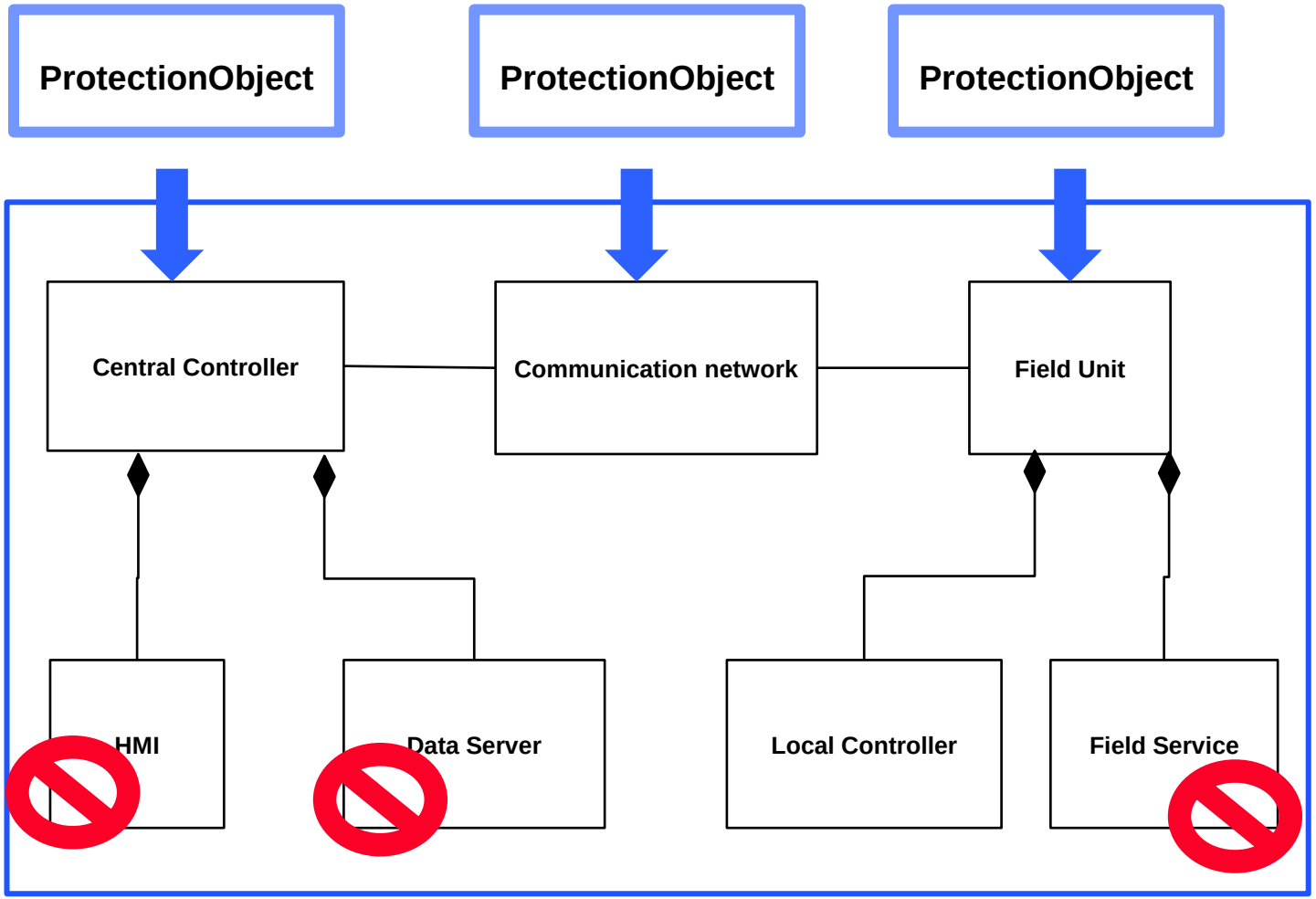
Access granted



Composition des patterns

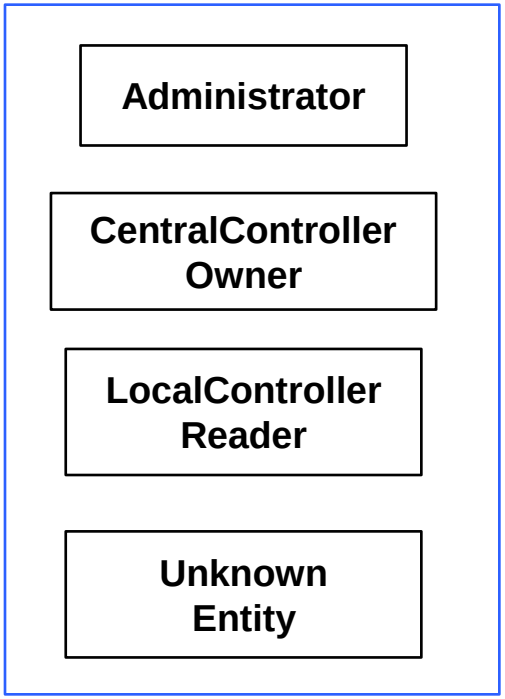


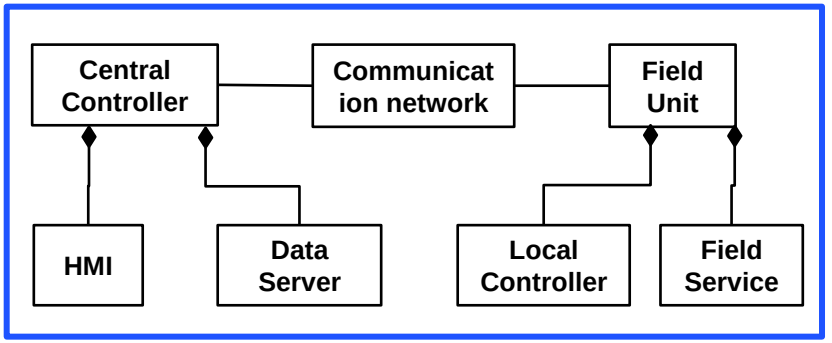
Application des patterns



Propriétés à respecter

Entités externes





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

Exemples de propriétés à vérifier (1/2)

	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
	Autho 1	*Autho 3*	*Autho 2*	*Autho 4*

(**Confidentiality**)

Autho 1 : *Si : not Administrator ET not CC Owner ET access = read ET access to CC*
alors : access not allowed

Autho 2 : *Si : not Administrator ET not LC Owner ET access = read ET access to LC*
alors : access not allowed

(Integrity)

Autho 3 : *Si : not Administrator ET not CC Owner ET access = write ET access to CC*
alors : access not allowed

Autho 4 : *Si : not Administrator ET access = write ET access to LC*
alors : access not allowed

Exemples de propriétés à vérifier (2/2)

	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

Autho 5 *Autho 7* *Autho 6* *Autho 8*

(**Availbility**)

Autho 5 : *Si : (Administrator OU CC Owner) ET access = read ET access to CC*
alors : access allowed

Autho 6 : *Si : (Administrator OU LC Owner) ET access = read ET access to LC*
alors : access allowed

Autho 7 : *Si : (Administrator OU CC Owner) ET access = write ET access to CC*
alors : access allowed

Autho 8 : *Si : Administrator ET access = write ET access to LC*
alors : access allowed