Rapport COA

Thomas Daniellou & Amona Souliman

Université de Rennes 1

Enseignant : Noël Plouzeau

Table des matières

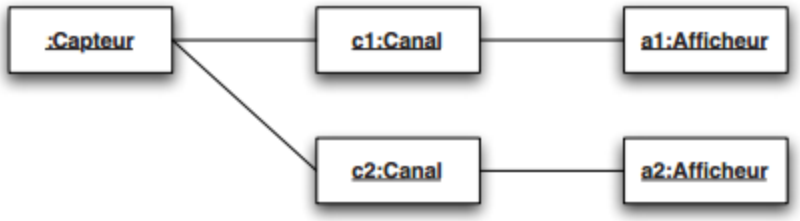
1 - Introduction

Le but de ce TP était de réaliser un service de diffusion de données de capteur en utilisant le patron de conception *Active Object* vu en cours de COA ainsi que ceux vu en cours d'AOC et ACO.

Trois versions ont été étudiées durant les TD de COA et seule la troisième fut développée pour ce TP.

2 - Description du service de diffusion

Au final, ce service permettra de diffuser un flot de valeurs vers des objets abonnés exécutés dans des threads différents de la source du service. L'objectif du TP étant la mise en œuvre parallèle d'*Observer*, les données diffusées seront une séquence croissante d'entiers (c'est à dire un simple compteur). Le compteur sera incrémenté à intervalle fixe. La transmission de l'information vers les abonnés au service emploiera un canal avec un délai de transmission aléatoire.



L'architecture comprendra donc :

- une source active (capteur), dont la valeur évolue de façon périodique
- un ensemble de canaux de transmission avec des délais variables
- un ensemble d'afficheurs réalisés en utilisant JavaFX
- un ensemble de politiques de diffusion, comprenant
    - la diffusion atomique (tous les observateurs reçoivent la même valeur, qui est celle du sujet)
    - la diffusion séquentielle (tous les observateurs reçoivent la même valeur, mais la séquence peut être différente de la séquence réelle du sujet)
    - la gestion par époque comme vu en cours.

3 - Architecture

Pour réaliser ce TP, nous avons utilisé les classes d'Oracle telles que :

- ScheduledExecutorService
- Future
- ou encore Callable

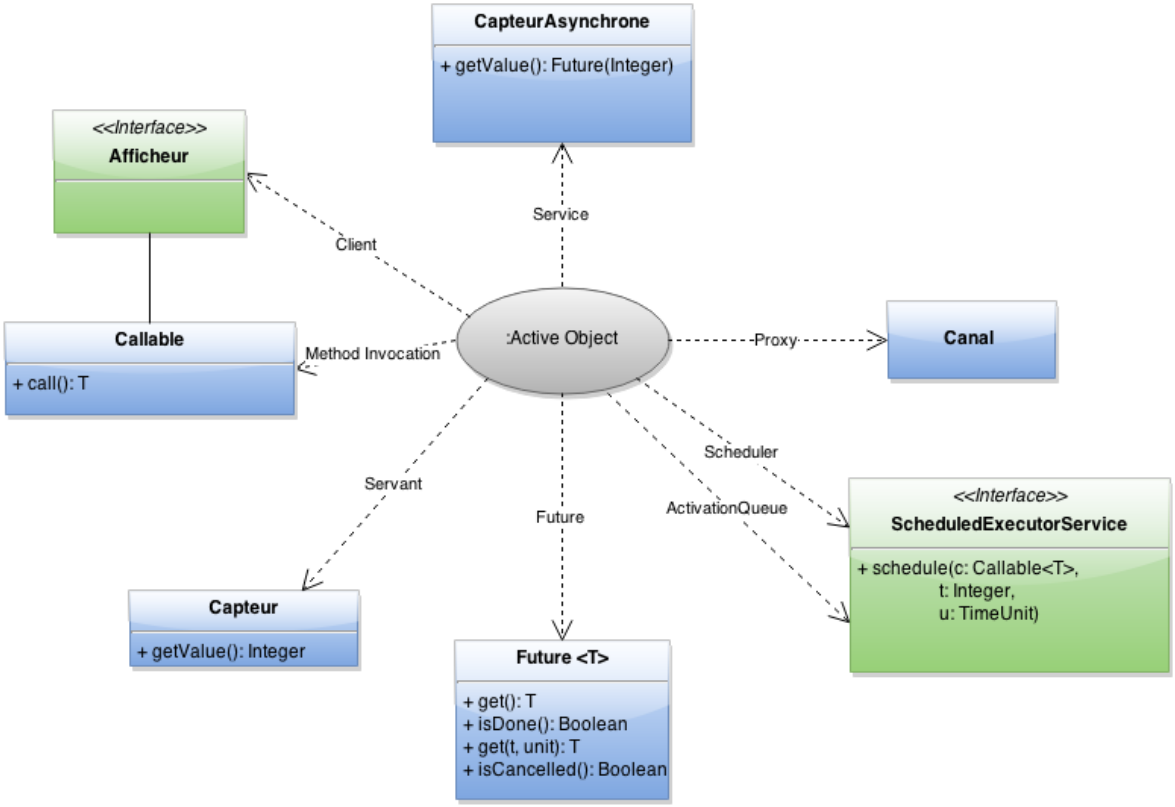


Diagramme de classe du patron Active Object appliqué au problème.

Lorsque la valeur du capteur est changée, une mise à jour des afficheurs est effectuée par l'intermédiaire des canaux.

Le patron de conception *Observer* a été utilisé pour la mise à jour des afficheurs lorsque la méthode *tick()* est appelée. C'est cette méthode *tick()* qui va permettre l'incrémentation de la valeur du capteur.

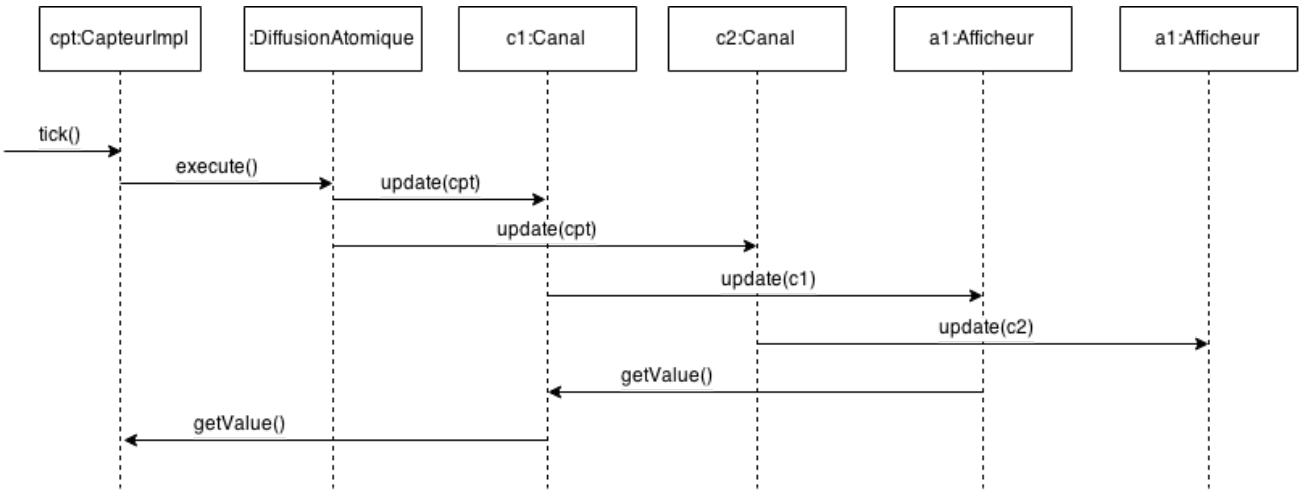


Diagramme de séquence illustrant l'appel à la méthode tick()

Cette mise à jour est faite de manière asynchrone en utilisant la classe `Future`.

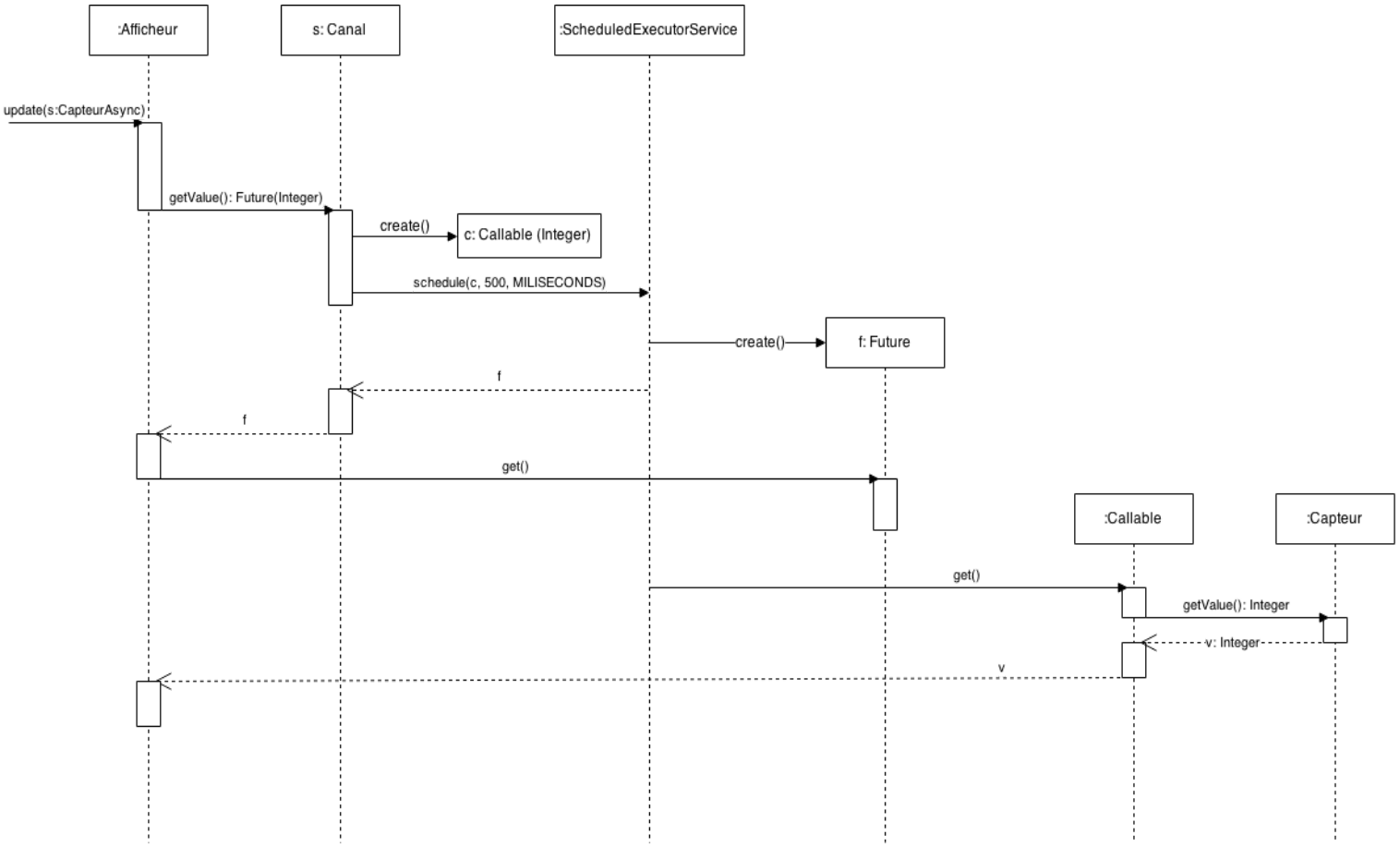


Diagramme de séquence illustrant l'appel à la méthode update()

D'autres patrons de conception tels que *Adapter*, *Strategy*, *Proxy* ou encore *MVC* ont été utilisés pour ce TP.

4 - Conclusion

Nous avons donc pu réaliser ce TP grâce aux classes `ScheduledExecutorService`, `Callable` et `Future` d'Oracle. Cela nous a donc permis de nous familiariser avec le patron de conception Active Object et de l'ajouter à notre liste de patrons appris durant ces deux années à l'ISTIC.

Le patron de conception Active Object nous a permis d'appréhender le parallélisme en Java d'une manière différente de celle vu en cours de Systèmes en M1.

De plus, l'étude précise de l'architecture du programme en TD de COA nous a permis de cerner le fonctionnement et l'implémentation nécessaire à la réalisation du TP.