# Le patron de conception Active Object

# Objectif du PC

- Mettre en œuvre le concept d'invocation asynchrone d'opération
- Ceci au moyen de l'appel synchrone classique

# Principes

- Emploi de la notion de Command
- Gestion asynchrone des commandes au moyen de threads

# Rôles

- Client
  - demande l'exécution d'opérations de service
  - emploie un appel classique (synchrone) mais avec retour d'un objet Future

# Rôles (2)

- Service
  - définit la liste des opérations qui peuvent être appelées de façon asynchrone
- Proxy
  - met en œuvre l'interface de Service en procédant par réification (façon Command)

# Rôles (3)

- Method invocation
  - représente une commande asynchrone
  - contient les paramètres
  - référence un Future
- Future
  - contient la valeur de retour
  - permet la synchronisation

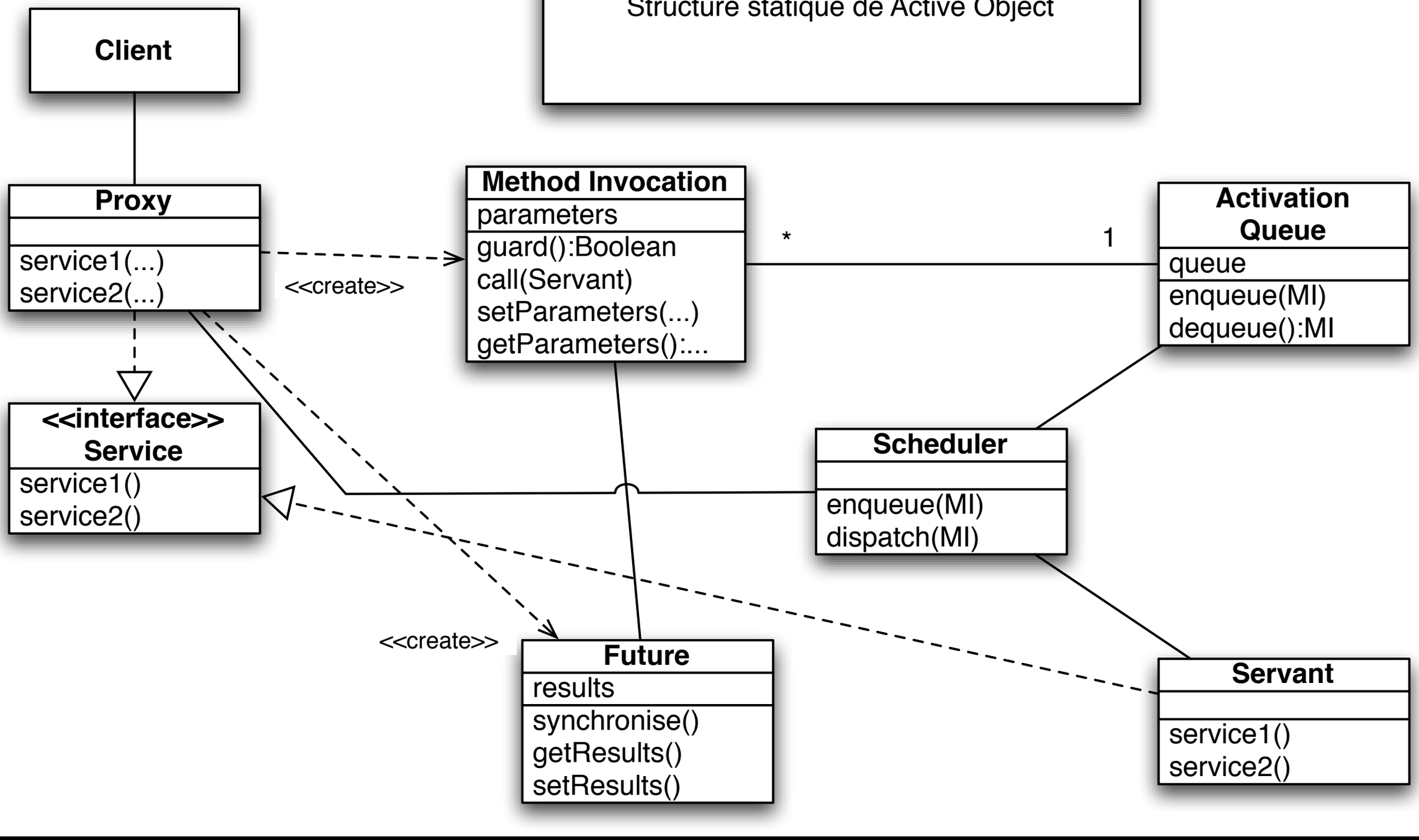
# Rôles (5)

- Scheduler
  - gère la file de Method Invocation en attente
    - sélectionne la prochaine Method Invocation (MI) à exécuter
- Activation queue
  - stocke les MI en attente

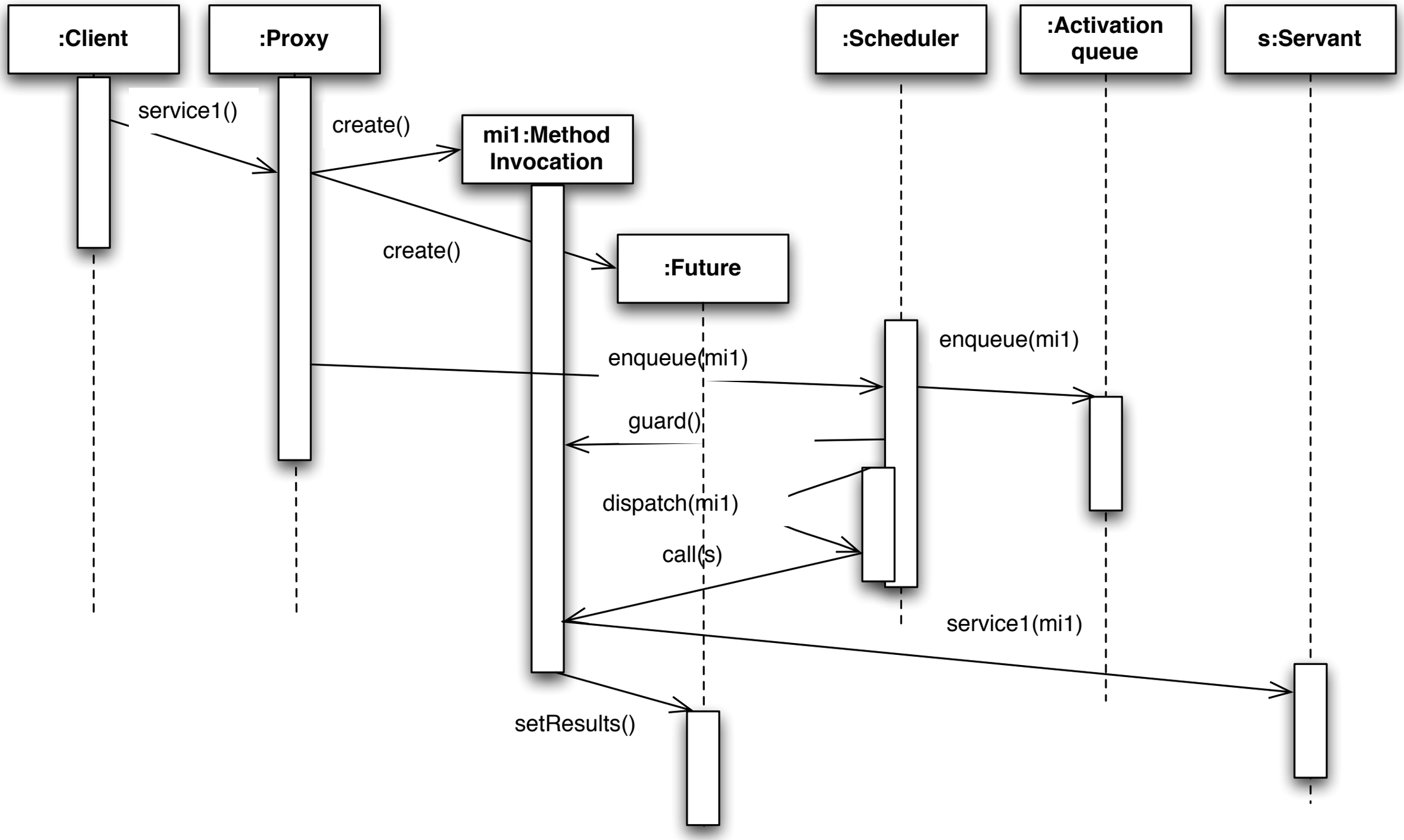
# Rôles (6)

- Servant
  - contient la mise en œuvre des service
  - chaque servant s'exécute dans un thread séparé

Structure statique de Active Object



Exemple d'exécution de Active Object



# Transition synchrone vers asynchrone

- Cœur du PC
  - un thread côté client (utilisateur)
  - un thread (ou plus) côté service
- Changement de la sémantique d'appel
  - apparence d'appel synchrone
  - prise en compte nécessaire du Future

# Asynchrone vs concurrent

- Le PC Active object supporte
  - premièrement l'appel asynchrone de service
  - secondement la gestion de de la concurrence
- Ces deux aspects sont très différents

# Intérêt de Active Queue

- Séparée de Scheduler
  - permet au proxy de donner des indications de séquencement
  - par exemple tous les appels à service1() sont traités dans une même queue et de façon séquentielle
  - communication par Method Invocation

# Gestion des interruptions

- Quel mécanisme de communication appelant/appelé
  - Comment intégrer le modèle existant (séquentiel) dans le modèle objet du langage)
- Parallèle avec RMI
  - Un appel d'opération peut échouer pour des raisons extrafonctionnelles

# Contrôle d'accès concurrent

- Active objet ne propose aucun mécanisme de contrôle de concurrence
- Compatibilité avec des verrous
  - prise de verrou lors de l'invocation
  - prise de verrou lors de l'exécution

# Cycle de vie

- Le caractère asynchrone rend visible des événements autrement confondus en mode synchrone
  - intervalle invocation/début d'exécution
  - intervalle début d'exécution/fin d'exécution
- Ceci donne des possibilités de contrôle
  - abandon, suspension

# Dépendances causales

- Mécanisme de plus haut niveau que le PC Active Object
- Permet d'indiquer un ordre partiel entre invocations de service
- L'ordonnancement est calculé par le Scheduler