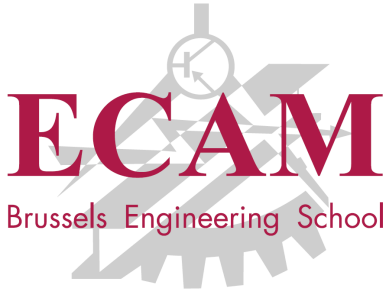


I402A Architecture logicielle

Séance 9

Architecture orientée-données

Sébastien Combéfis

2017–2018

Rappels

- Des architectures monolithiques aux (micro-)services

  Du monolithique à sa distribution jusqu'aux (micro-)services

- Architectures orientée-services
  - Principe de l'orientation services et ses caractéristiques
  - Composants des services web et mécanisme d'exécution

- Architecture REST pour proposer des services web RESTful

  Principes de base et quick tips

Objectifs

- Comprendre architecture orientée autour des données
  - Communication inter-composants exclusivement via données
  - Identification des déplacements de données
- Deux classes principales d'architecture orientée données
  - Communication par flux de données
  - Architecture centrée autour de données

Orientation donnée

- Exécution de processus selon la disponibilité de données

  Contrôle de l'exécution par les données

- Structure de l'architecture selon le mouvement des données

  Comment les données passent d'un composant à un autre

- Pas d'interaction directe entre les composants

  Les données jouent le rôle de connecteur entre composants

Flux de données

Architecture orientée flux de données (1)

- Transformations sur des données ou ensemble d'inputs

  Les données et opérations sont indépendantes

- Différents modules connectés entre eux

  I/O stream, I/O buffer, pipes...

- Plusieurs topologies d'interconnexion possibles

  Linéaire sans cycles, avec cycles, arborescent...

Architecture orientée flux de données (2)

- Augmente la réutilisabilité et la modifiabilité des applications

  Par exemple compilateur, traitement de business data

- Trois principales architectures orientées flux de données
  - Batch séquentiel
  - Pipe et filtre, pipeline
  - Contrôle de processus

Batch séquentiel (1)

- Enchainement de sous-systèmes de transformation de données

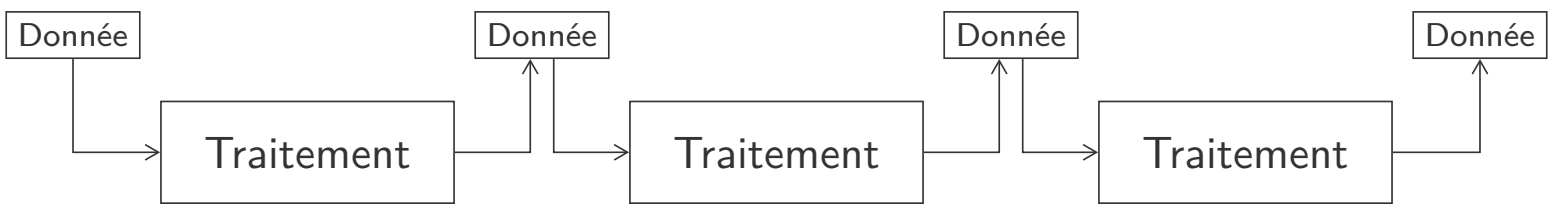
  Attente de la terminaison du sous-système précédent

- Échange de batch de données entre sous-systèmes

  Par des fichiers partagés d'une manière ou d'une autre

- Très utilisé dans tout ce qui est business data processing

  Domaine bancaire, facturation de services...



Batch séquentiel (2)

- Petit nombre de gros sous-systèmes indépendants

  Doivent être exécutés séquentiellement dans un ordre fixé

- Opèrent sur de larges fichiers « plats »

  Transformations successives du premier fichier d'entrée

- Faible couplage des sous-systèmes 

  Seule connexion via les fichiers échangés

Avantage

- Division beaucoup plus simple des sous-systèmes

  Indépendants et représentant une opération plus élémentaire

- Sous-système simple sous la forme input-output process

  Pas d'état global à l'application à maintenir

Inconvénient

- Latence élevée causant un faible débit

  Un sous-système bottleneck bloque tout le processus

- Limitation à des structures complètement linéaires

  Input $\rightarrow$ *Processing* $\rightarrow$ *Output*

- Pas de concurrence par rapport à l'opération

  Sauf en multipliant les chaines de transformations

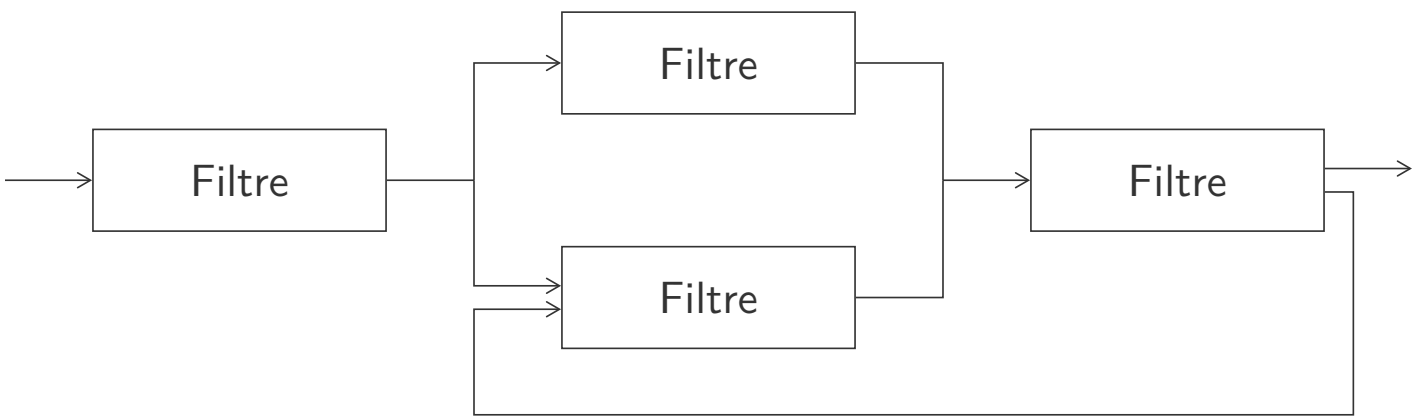
Pipe et filtre (1)

- Transformation incrémentale de données par des composants

  Combinaison de sources et puits de données, filtres et pipes

- Connexions entre composants avec flux de type FIFO

  Flux d'octets, de caractères, etc.



Pipe et filtre (2)

- Données traitées incrémentalement alors qu'elles arrivent

  Output est produit incrémentalement, sans attendre fin de l'input

- Exécution concurrente des différents composants

  Différence entre manipuler des flux ou des batch de données

- Pipes sans état transportent les flux de données

  Rôle de connexion passive entre filtres

Filtre

- Filtre définit une étape de calcul d'un processus

  Système représenté par une chaine de filtres

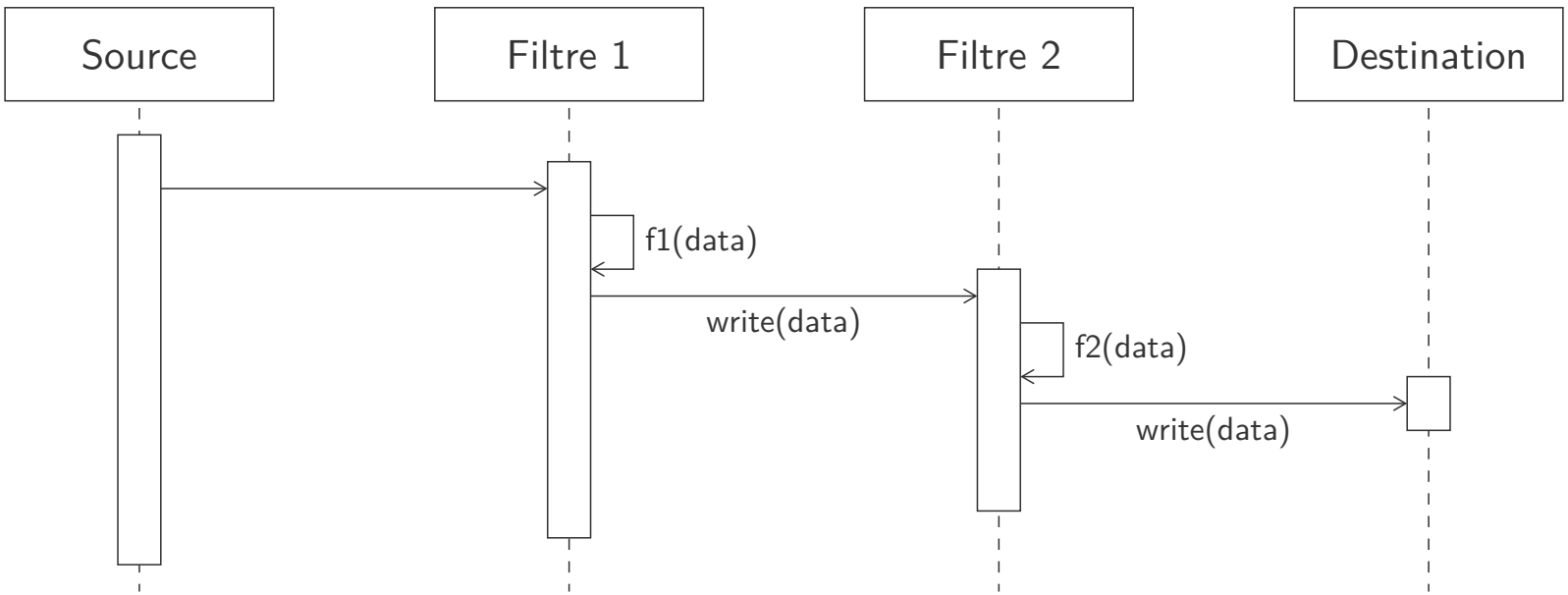
- Filtres doivent être indépendants entre eux
  - Pas de partage d'états entre différents filtres
  - Ne savent pas à qui ils ont été connectés
- Deux types de filtres selon le rapport aux données
  - **Filtre actif** récupère des données, traite et écrit en sortie
  - **Filtre passif** demande données au filtre suivant

Filtre actif

- Exemple de filtre actif avec les pipes UNIX (*pull-in push-out*)

  Source de chaque pipe pousse les données

- Par exemple : `history | grep git | wc -l | tee result.txt`



Exemple : pipes UNIX

- Enchainement de processus pour former un pipeline

  Chaine d'exécutions séquentielles de processus

- `history | grep git | wc -l | tee result.txt`

  1. `history` sort l'historique des commandes exécutées
  2. `grep` filtre les lignes contenant « git »
  3. `wc -l` compte le nombre de lignes
  4. `tee` copie l'entrée standard vers un fichier

- Attention aux filtres agrégatifs qui cassent concurrence

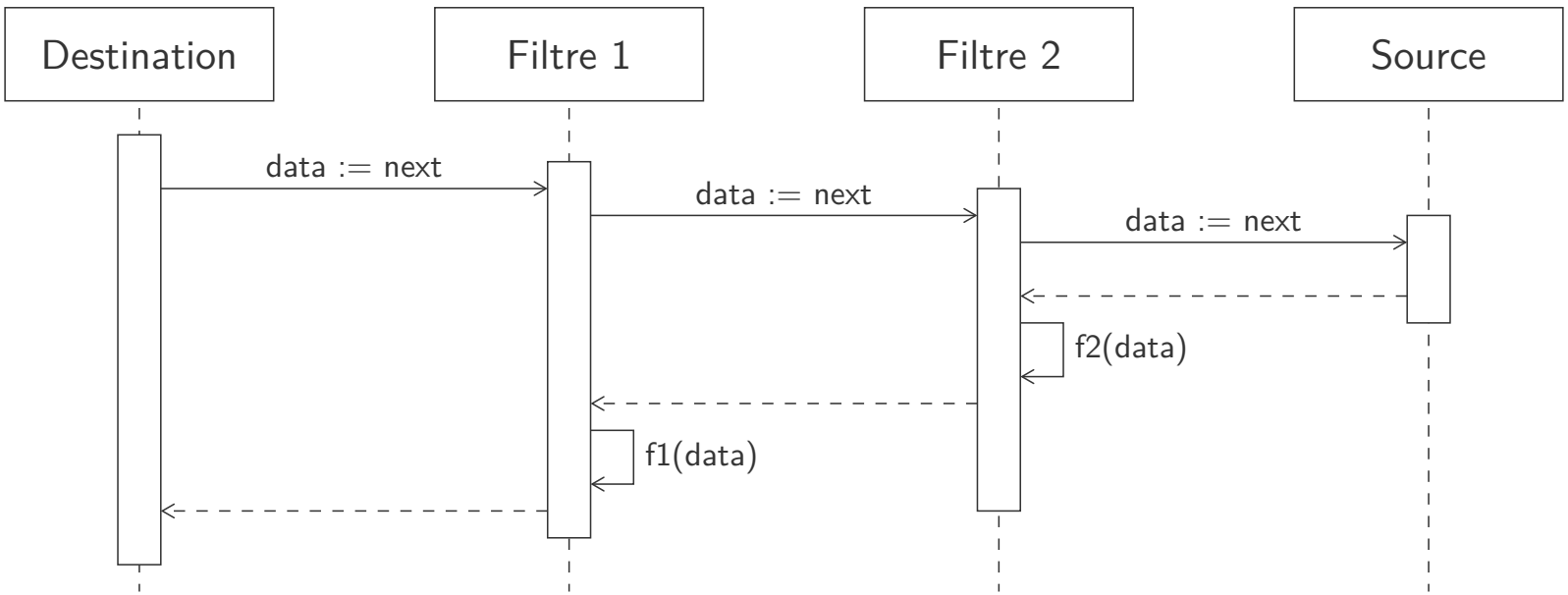
  Nécessité d'attendre toutes les données de l'input

Filtre passif

- Exemple de filtre passif avec un lexer (*push-in pull-out*)

  Destination de chaque pipe tire les données

- Par exemple : `t := lexer.next_token`



Exemple : compilateur

- Compilateur construit avec plusieurs composants
  - **Analyseur lexical** découpe chaine en tokens
  - **Analyseur syntaxique** construit arbre syntaxique avec tokens
  - ...
- L'analyseur syntaxique demande les tokens au lexer

  Pipeline à l'envers depuis la destination vers la source

Cas particulier

- Pipeline de filtres consécutifs

  Restriction à une topologie linéaire

- Pipes bornés en taille maximale gérable

  Restriction sur la quantité de données par pipe (bounded buffer)

- Pipe typé par rapport à son contenu

  Données doivent être d'un certain type particulier

Avantage

- Permet une très bonne concurrence et débit de sortie

  Grande flexibilité entre exécution séquentielle et parallèle

- Division claire, grande réutilisabilité et maintenance facilitée

  Faible couplage entre filtres, seulement connectés par pipes

- Évolution facile par ajout de nouveaux filtres

  Permet également du prototypage rapide en réutilisant des filtres

Inconvénient

- Pas d'interactions et communications entre les filtres

  Seule possibilité est de simuler avec une entrée de contrôle

- Obligation de définir un format commun de données

  Seuls des filtres compatibles peuvent être connectés entre eux

- Architecture difficilement configurable dynamiquement

  Les liens entre filtres et les pipes sont figés

- Faible tolérance aux pannes et lourdeur de parsing données

  Partager des valeurs globales est aussi très difficile

Batch versus pipe et filtre

- Batch/pipe et filtre définissent séquences de calculs simples

  Interactions se passent uniquement par l'échange de données

Batch séquentiel	**Pipe et filtre**
gros composants (*coarse-grained*)	petits composants (*fined-grained*)
grande latence	aussitôt que données disponibles
accès externe aux données	données localisées
pas de concurrence	possibilité de concurrence

Contrôle de processus (1)

- Architecture pour contrôler l'exécution d'un processus

  Typiquement processus physique avec d'éventuelles rétroactions

- Données proviennent d'un ensemble de variables de contrôle

  Pas de batch de données, ni de flux de données

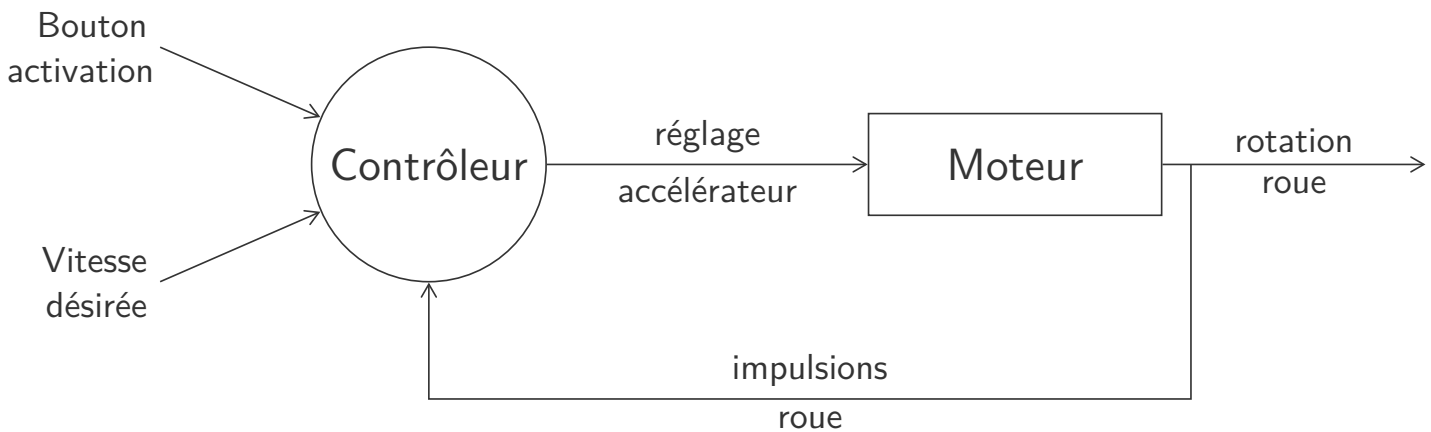
- Système décomposé en sous-systèmes ou modules

  Connexion identifient les échanges/communications des variables

Exemple : cruise control

- Variables de contrôle proviennent de différentes sources

  Bouton activation, indicateur vitesse, compteur tours de roue



Contrôle de processus (2)

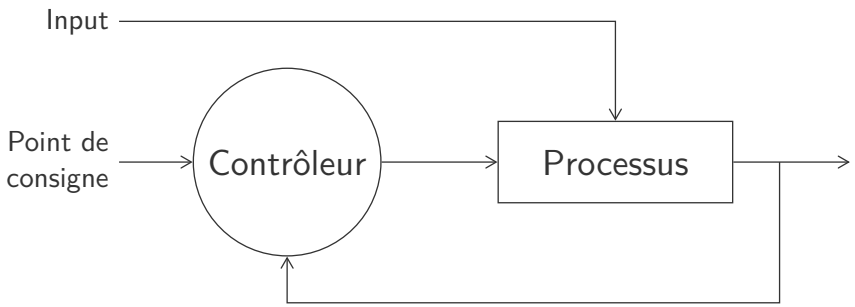
- Deux types d'unités dans architecture contrôle de processus
  - **Unité de processing** change les variables de contrôle
  - **Unité de contrôle** calcule et effectue des changements
- Trois types de connecteurs entre les unités
  - **Variable de processus** de trois types

    Contrôlée (mesurée), input, manipulée (modifiée par contrôleur)
  - **Point de consigne** valeur désirée pour variable contrôlée
  - **Senseur** pour obtenir valeurs variables

Type de boucle

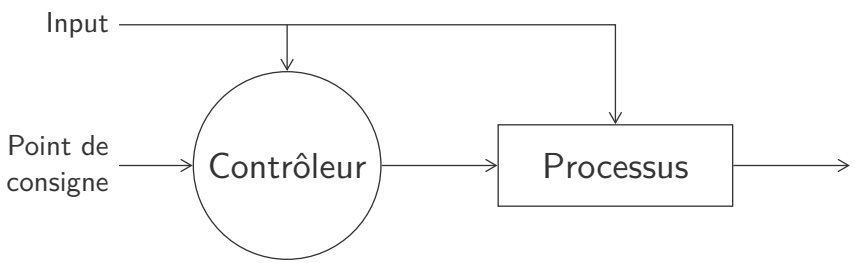
- Système de contrôle en boucle de rétroaction (*feedback*)

  Variables de contrôle pour manipuler variable processus



- Système de contrôle en boucle ouverte (*feedforward*)

  Pas d'utilisation variables processus



Centré sur les données

Architecture centrée données (1)

- Données centralisées souvent accédées par composants

  Les données sont également modifiées

- Objectif principal est d'assurer intégration des données

  Intégration pour l'ensemble des composants

- Composants communiquent via dépôts de données partagés

  Le plus indépendants possibles, peu de communication directe

Architecture centrée données (2)

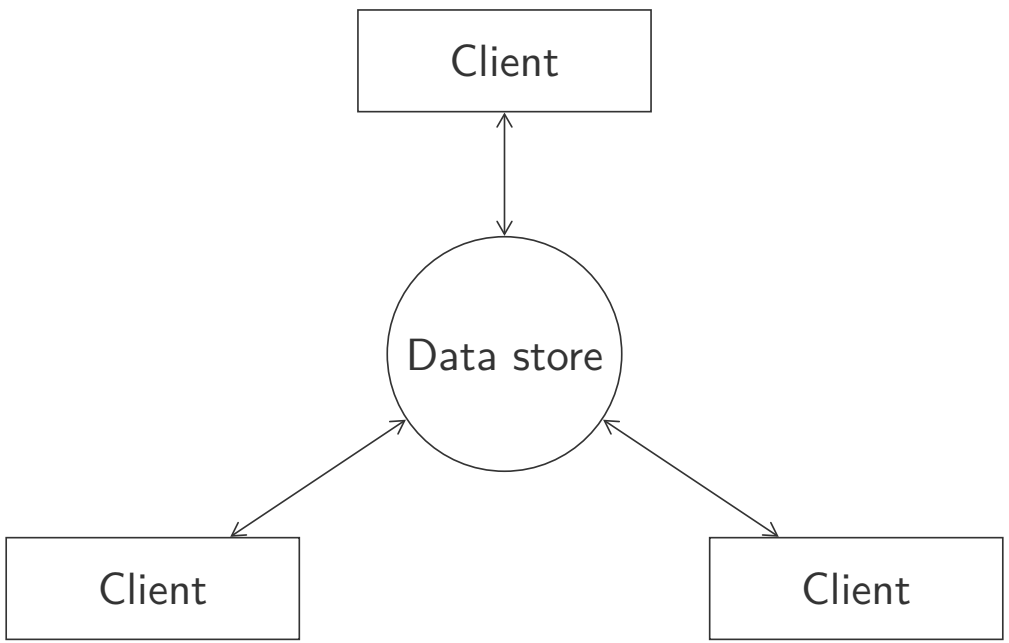
- Deux exemples connus d'architecture centrée données
  - Base de données avec définition de tables et types de données
  - Web-services de donnnées partagés suivant modèle hypermédia
- Deux types de composants qui vont interagir entre eux
  - **Data store** offre le stockage permanent de données (*état*)
  - **Data accessor** demande et envoie des données vers le store
- Deux catégories en fonction du flux de contrôle

  Dépôt et blackboard

Dépôt (1)

- Data store est complètement passif, aucune initiative

  Les client sont actifs et définissent le flux de contrôle



Exemple : compilateur

- Données partagées entre modules d'un compilateur

  Analyseur lexical, syntaxique, préprocesseur...

- Données mises à jour et accédées par les composants

  Table des symboles, arbre syntaxique abstrait (AST)...

Dépôt (2)

- Vérification de changements du store faite par clients

  Le client envoie une requête au système pour faire actions

- Utilisé largement dans plusieurs types de systèmes
  - Systèmes de gestion de bases de données (SGBD)
  - Répertoire d'interfaces dans CORBA
  - Environnement *Computer Aided Software Engineering* (CASE)
  - ...

Avantage

- Assure l'intégration des données par cloisonnement de celles-ci

  Facilité de backup et restauration des données

- Évolution, mise à l'échelle, réutilisabilité d'agents

  Puisque pas de communication directe entre agents

- Réduction de la quantité de données transitoires

  Grâce au dépôt centralisé de données, moins de copies locales

Inconvénient

- Système plus vulnérable à une panne ou à une attaque

  Nécessité de bien sécuriser le dépôt central 

- Structure des données du dépôt central critique

  Forte dépendances entre clients et store, changement difficile

- Difficile et cher de faire évoluer les données

  Notamment cout pour déplacer sur réseau en donnée distribuée

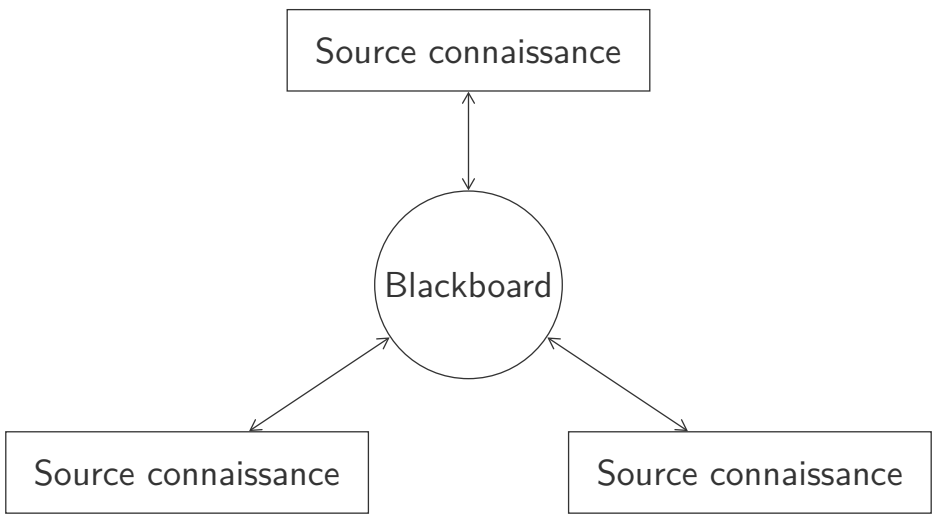
Blackboard (1)

- Data store actif avec des clients passifs

  État du store détermine complètement le flux de contrôle

- Un composant blackboard est central à l'architecture

  Source de connaissances interagissent via le blackboard



Exemple : Salle de chat

- Salle de chat en ligne avec des clients connectés

  Stockage de tous les messages sur un serveur central

- Discussion à plusieurs clients dans une seule salle

  Ajout d'un message depuis un client vers le blackboard

- Rafraichissement automatique des messages par clients

  Les clients doivent d'abonner à la salle de discussion

Blackboard (2)

- Trois composants dans l'architecture blackboard
  - **Sources de connaissances** répond à changement dans le store
  - **Structure de données blackboard** donnée d'état
  - **Contrôle** réalise le lien sources et blackboard
- Utilisation du modèle avec listeners (ou subscribers)

  Pour être notifié des changements d'états du blackboard
- Communication inter-source de connaissances via blackboard

  L'écriture d'une donnée va activer d'autres sources

Avantage

- Évolutivité/mise à l'échelle par ajout sources de connaissance

  Il suffit de s'enregistrer auprès du blackboard pour être notifié

- Possibilité forte d'avoir de la concurrence

  Travail parallèle des sources de connaissance indépendantes

- Réutilisabilité des sources de connaissances

  Facilité pour réaliser des expériences pour tester des hypothèses

Inconvénient

- Changement difficile de la structure du blackboard

  Lien fort entre source de connaissances et blackboard

- Très difficile de tester un système avec blackboard

  Notamment pas évident de savoir quand l'exécution est terminée

- Problèmes lors de la synchronisation de plusieurs agents

  Mécanismes lourd de protection du blackboard

Crédits

- https://www.flickr.com/photos/jeffgmoore/5463638799
- https://www.flickr.com/photos/small_realm/11189803153