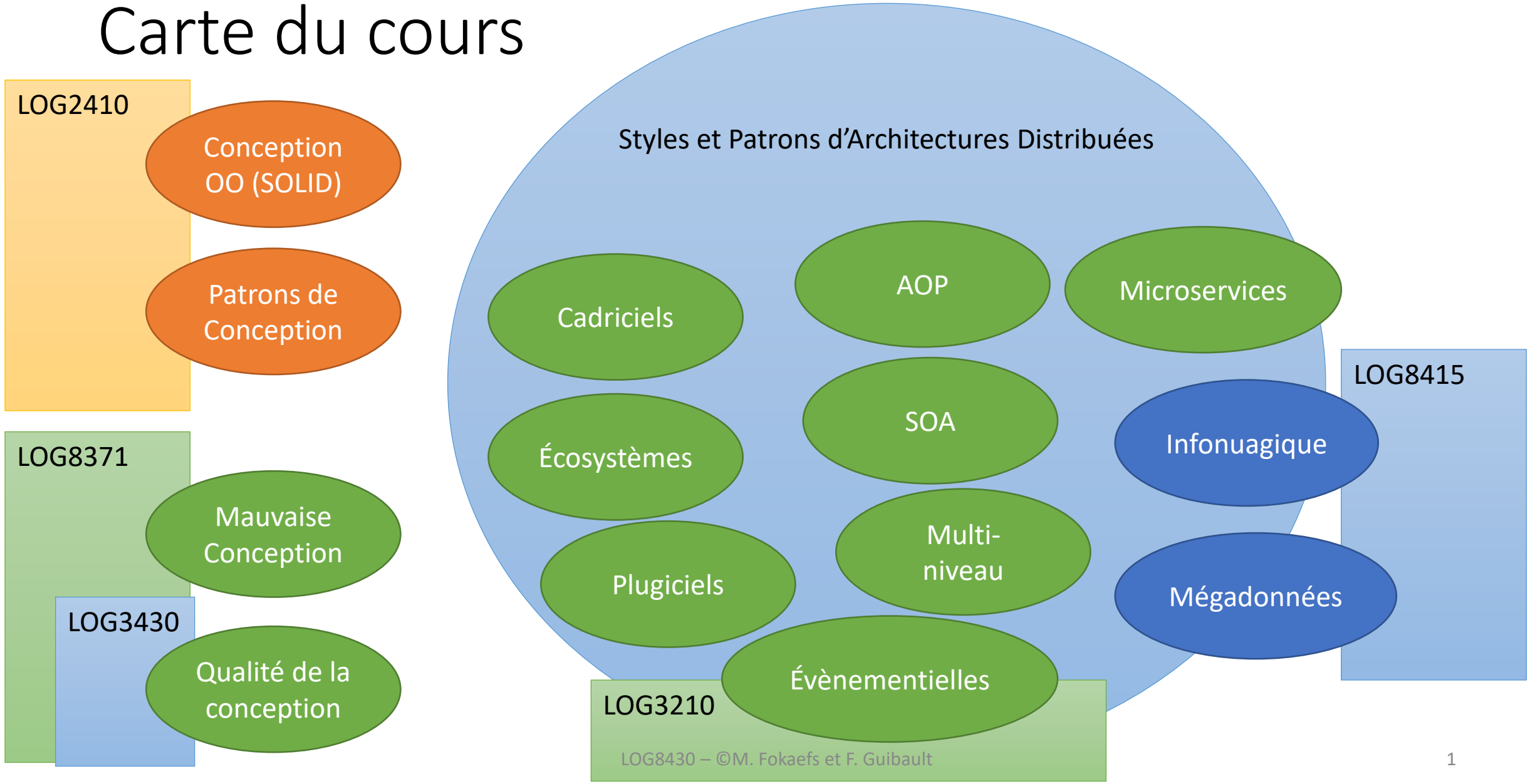
Carte du cours



POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

LOG8430: Styles d'architecture

Aujourd'hui

LOG2410

Conception OO (SOLID)

Patrons de Conception

Styles et Patrons d'Architectures Distribuées

Évolution et styles d'architecture

Monolithique

- Mainframe

Modulaire

- Pipe-Filter
- Layered
- MVC
- Blackboard
- Event-bus

Distribué

- Client-server
- Multitier
- Master-slave
- Peer-to-peer
- Broker
- SOA
- REST

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Architectures Monolithiques

Architectures Monolithiques

- Un seul programme.
  - Un seul endroit de changement.
- Il n'y a pas de distinction entre l'IU, la logique, les données etc.
  - Les éléments du programme sont fortement liés.
- Le programme est développé et maintenu seulement par une petite équipe de développeurs/experts.
  - Connaissance absolue et concentrée.
- Il y a juste un propriétaire du système entier.
  - Contrôle absolu.
- La logique et les données existent dans un seul lieu.
  - La propriété est bien protégée.
- Chaque copie du programme s'exécute sur un seul ordinateur.
  - Les frais d'exécution sont réduits.

pas de dépendance externe et des changements contrôlés

interface utilisateur IU

Un seul serveur gère la logique

de l'expertise et des données, ex: des données privées, sensibles sont bien protégés car on est le proprio de l'infrastructure

car on doit maintenir une seule infrastructure

Monolithique: Avantages

Vulnaribilité se produit slm s'il y a une connexion à l'externe

controler qui a accès à un système

- Sécurité maximale! Aucune opportunité pour une attaque externe.
- Le programme est souvent efficace, parce que tout est chargé en mémoire et il n'y a pas de délais liés au réseau ou au chargement de modules.
- Grâce à la taille de l'équipe, la communication et l'efficacité sont optimisées.
- Le développement, la maintenance et l'évolution du système sont la responsabilité d'une seule entité. il n'y a pas de dépendance à l'externe

Monolithique : Désavantages

- Pas de réutilisabilité. Il est impossible de réutiliser des parties du système, à moins de les copier en entier.
  - Cela crée des clones.
- Pas d'évolutivité ou d'extensibilité.
- Difficile à maintenir!
  - « La crise du logiciel »
- Difficile d'ajouter de nouveaux développeurs. onboarding processus est un défi
- Frais croissants pour la maintenance du système. couteux de maintenir des systèmes monolithiques

Exemples: Où des systèmes monolithiques peuvent etre applicables?
1. Flappy bird

Architectures Modulaires

Ex: Word, Excel

Architectures Modulaires

- Le système comprend plusieurs modules. plusieurs programmes
- Les fonctionnalités et les responsabilités sont plus distinctes.
- Les modules sont développés et compilés séparément.
- Les modules sont plus petits et plus facile à gérer.
- Le développement est aussi séparé entre plusieurs plus petites équipes.
- Ces architectures implémentent les principes de SRP et DIP.
- La propriété du système appartient toujours à une seule partie.

Modulaire : Avantages

plusieurs équipes qui s'occupent de modules spécifiques et qu'on va intégrer le tout ensemble à la fin et faire tests intégration

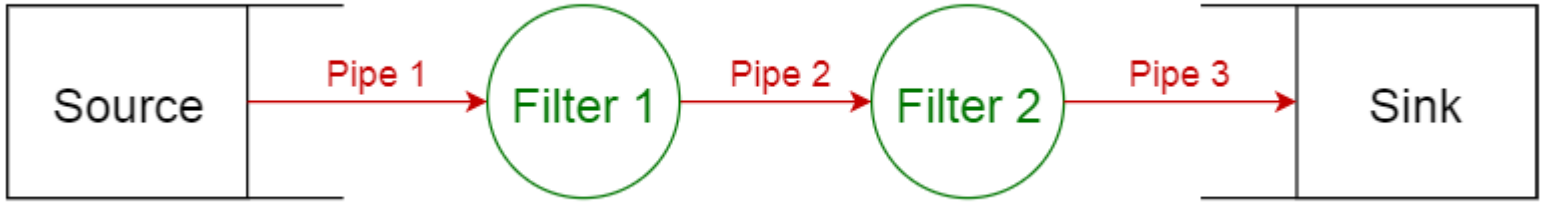
- Le développement est parallélisé, donc il est plus efficace.
- La maintenance est généralement plus facile, car les changements locaux et internes n'affectent pas les autres modules.
- Les modules sont aussi autonomes que possible au niveau des tests.
  - Tests unitaires.
- Il est plus facile d'ajouter du nouveau personnel, étant donné que les modules sont plus petits et plus facile à comprendre, et il y a moins de dépendances.
  - « Conception par contrat » : Les interfaces sont très spécifiques et connues.
- Les modules peuvent être réutilisés et changés.

Modulaire : Désavantages

- Comment assembler les modules en un seul système?
- La distribution du travail augmente le besoin de communication entre les équipes.
- La connaissance du système entier est perdue.
  - Il faut avoir une organisation hiérarchique.
- Il y a moins de dépendances, mais elles sont peut-être plus importantes.
  - Un défaut peut exister dans un module, mais cela reste inconnu par les autres modules. Son effet peut être apparent ailleurs dans le code, mais la correction n'est possible que dans un module.
- On doit investir plus d'efforts dans la gestion du système.
  - Tests, débogage, monitoring, maintenance.

Pipe-filter

Chaque algorithme change les
données et les prépare pour l'algorithme suivant

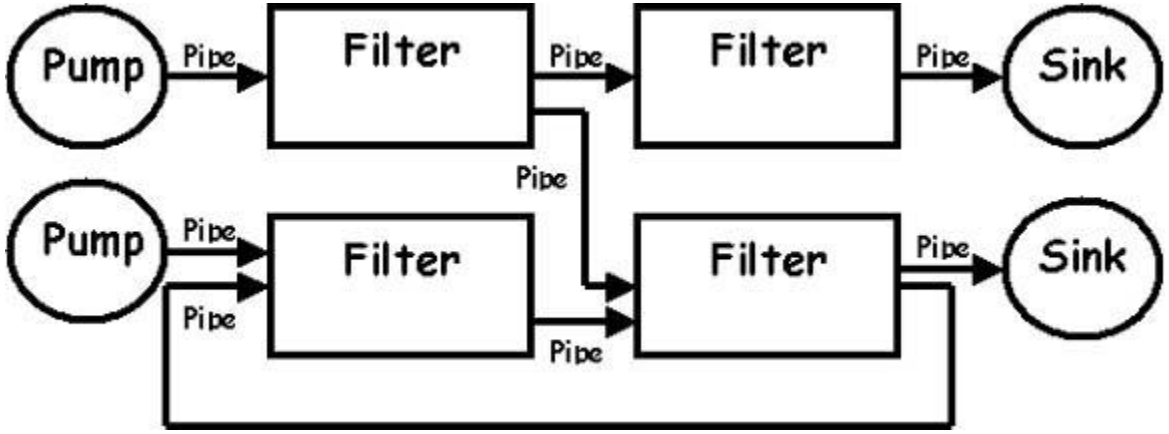


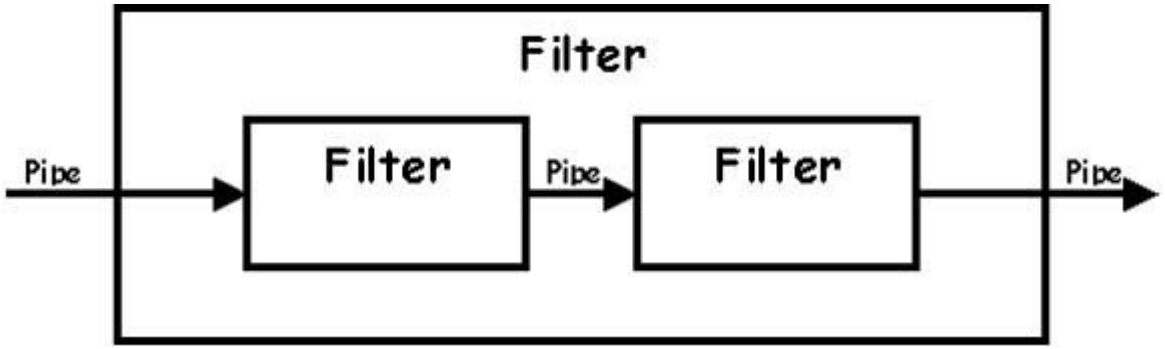
- Décomposition d'une fonctionnalité en composantes successives appliquant des transformations incrémentales sur les données.
- Un producteur envoie des données par des tuyaux et à travers de filtres à un consommateur.
- Les tuyaux sont les canaux de communication et les connecteurs entre les filtres et entre les producteurs et les consommateurs.
- Ils peuvent aussi fonctionner en tant que des agrégateurs de données ou des synchronisateurs.
- Les filtres sont des transformations appliquées progressivement aux données.
- Exemples : Programmes UNIX, compilateurs (analyse lexicale, analyse syntaxique, génération du code), apprentissage profond

Pipe-filter : Avantages

architecture efficace

- Efficace : Les filtres peuvent être parallélisés.
- Les filtres sont indépendants des producteurs et des consommateurs et ils peuvent être réutilisés.
- Il est possible d'ajouter ou de supprimer des tuyaux et des filtres en cours d'exécution.
- S'il n'y a pas suffisamment de données dans les tuyaux, les filtres attendent tout simplement.
- Il est possible d'introduire de la récursion et des compositions.





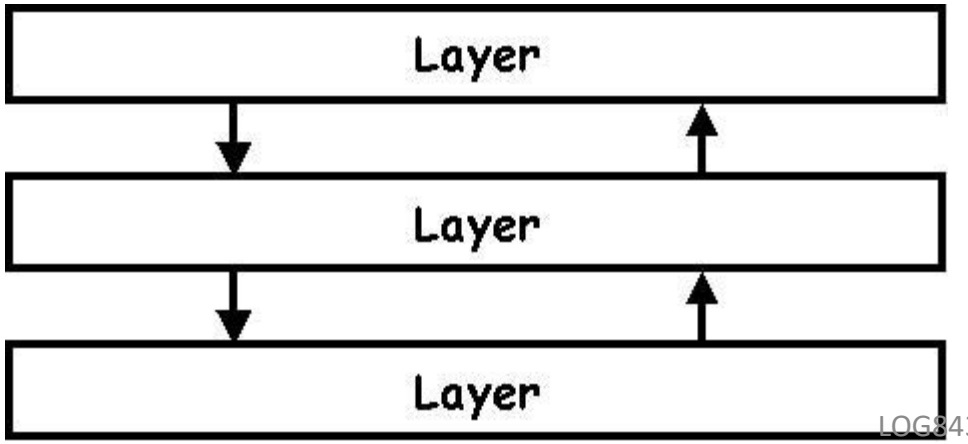
Pipe-filter : Désavantages

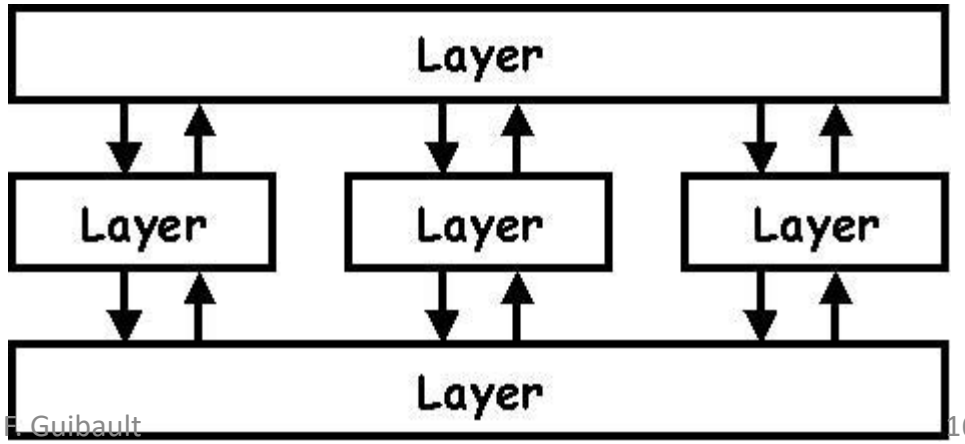
- Il est possible d'avoir des cas de dépassement de capacité ou de blocage (*deadlock*), si un filtre doit attendre pour toutes les données.
- Les tuyaux acceptent souvent seulement des type de données simples. Par conséquent, les filtres doivent faire plus de travail pour pousser et extraire les données dans les tuyaux.
- Si on implémente des tuyaux pour différents types, il n'est plus possible de connecter n'importe quel tuyau à n'importe quel filtre.

Layered

- La division d'architecture en couches.
- La couche est une notion abstraite et elle peut inclure des groupes de méthodes, de classes, de paquets etc.
- Chaque couche capture d'une responsabilité bien définie et distincte.
  - Présentation (IU), Application (Service), Logique (Domain), Accès aux Données (Persistance)
- Les éléments d'une couche ne peuvent utiliser que des modules de la couche précédente ou suivante.
  - La fonctionnalité s'exécute séquentiellement.
- Il est possible d'avoir plusieurs couches au même niveau, mais elles ne peuvent pas communiquer.

les couches ne peuvent pas communiquer





Layered : Avantages et Désavantages

- Avantages
  - La maintenance est facile et peu coûteuse.
  - On peu exploiter l'expertise des développeurs.
  - Les couches peuvent être réutilisées et partagées parmi plusieurs autres couches.
- Désavantages
  - Il est interdit (ou vraiment compliqué) d'utiliser une couche pas immédiatement adjacente.
  - L'architecture est sensible à la conception des interfaces.

Model-View-Controller MVC

- On peut dire que c'est une version de l'architecture en couches (mais pas toujours).
  - Couche Modèle : responsable des données.
  - Couche Vue : responsable de la présentation.
  - Couche Contrôleur : responsable de la logique.
- Un seul modèle peut être présenté par plusieurs vues.
  - Le modèle est responsable de connaître toutes les vues et de les avertir lors de changements. (patron Observer)
- Le modèle ne doit pas faire partie du système. Un lien à une base de données et la fonctionnalité pour accéder à la base sont suffisants.
- Le contrôleur réagit aux évènements dans la vue et il sait comment changer le modèle.

MVC

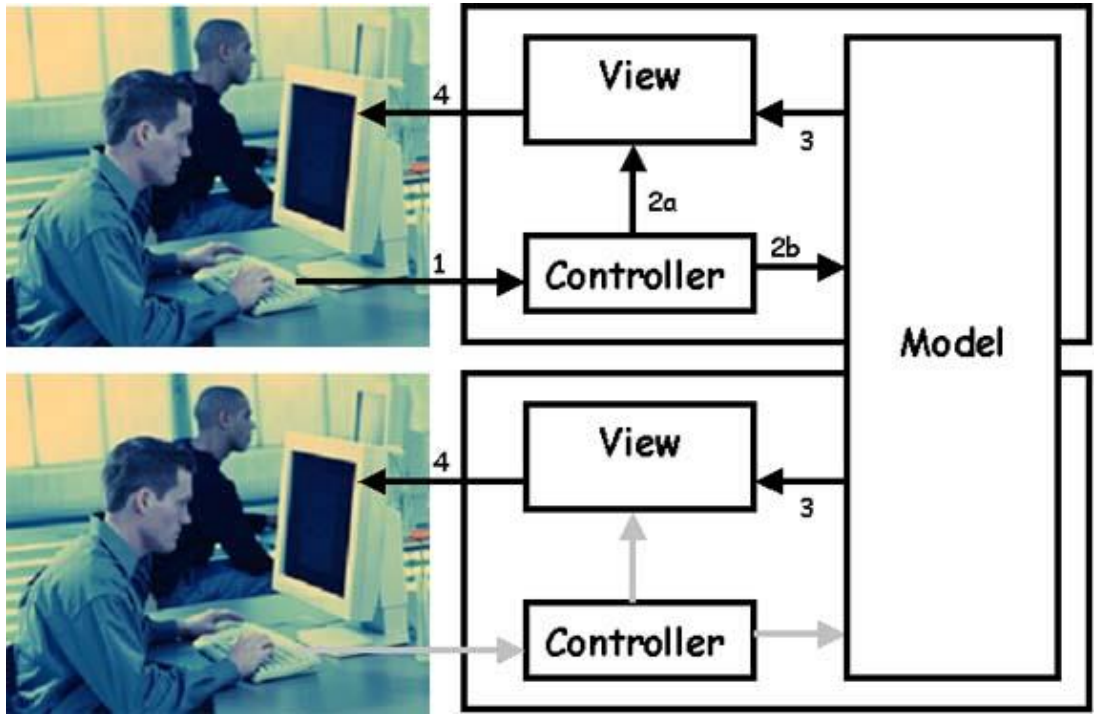
1. Le contrôleur accepte une action par l'utilisateur.

2a. Selon cette action, le contrôleur met à jour les caractéristiques de la vue…

2b. … et/ou le modèle.

3. Si le modèle est changé, il notifie toutes les vues enregistrées.

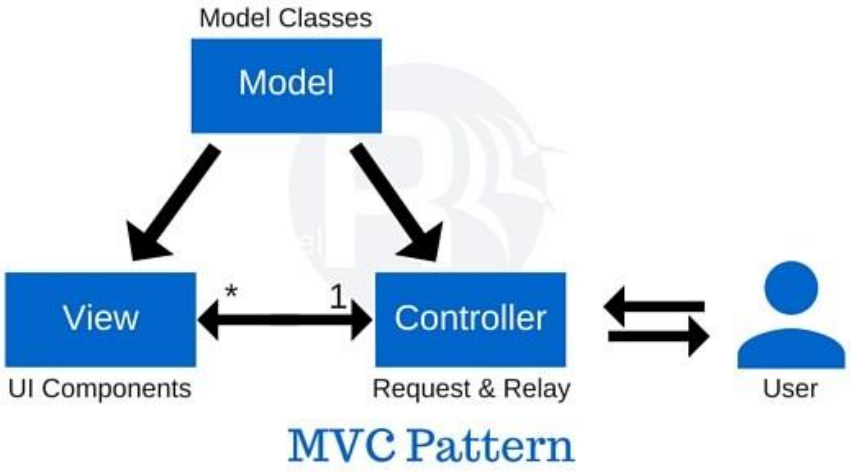
4. Les vues se mettent à jour.

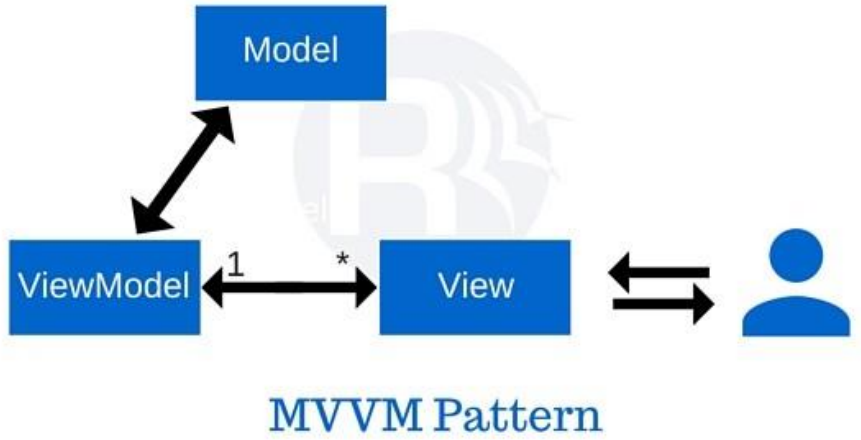


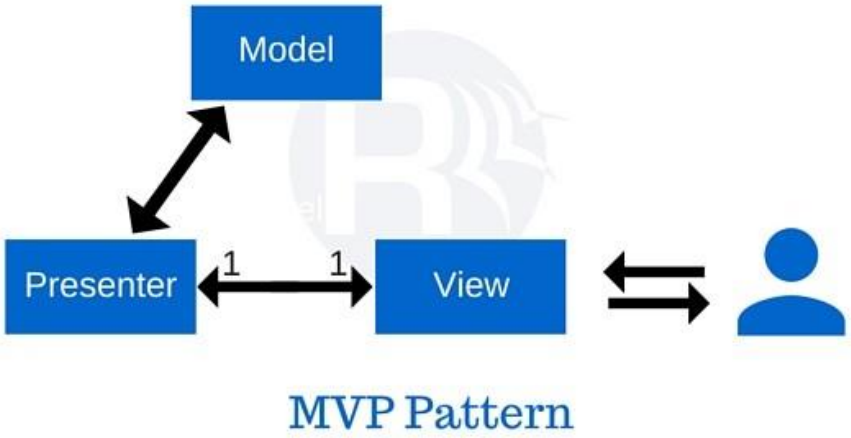
MVC - alternatifs

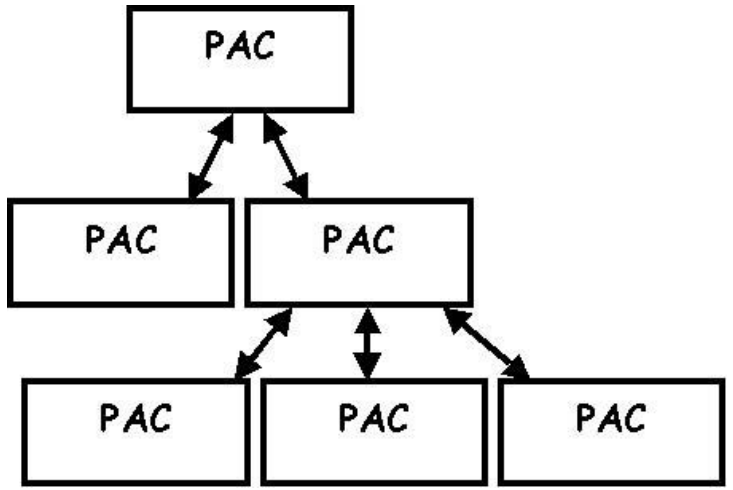
- Il y a plusieurs implémentations alternatives de MVC.
  - MVP – Model-View-Presenter (IBM)
    - L'utilisateur interagit avec le présenter, pas la vue. Un présenter par vue.
  - MVVM – Model-View-ViewModel (Microsoft)
    - ViewModel peut contrôler plusieurs vues.
  - MVA – Model-View-Adapter
    - Le modèle et la vue ne sont pas connectés.
  - PAC – Presentation-Abstraction-Control
    - Si le modèle est modulaire. Les composantes sont organisées dans une hiérarchie.

MVC - alternatifs







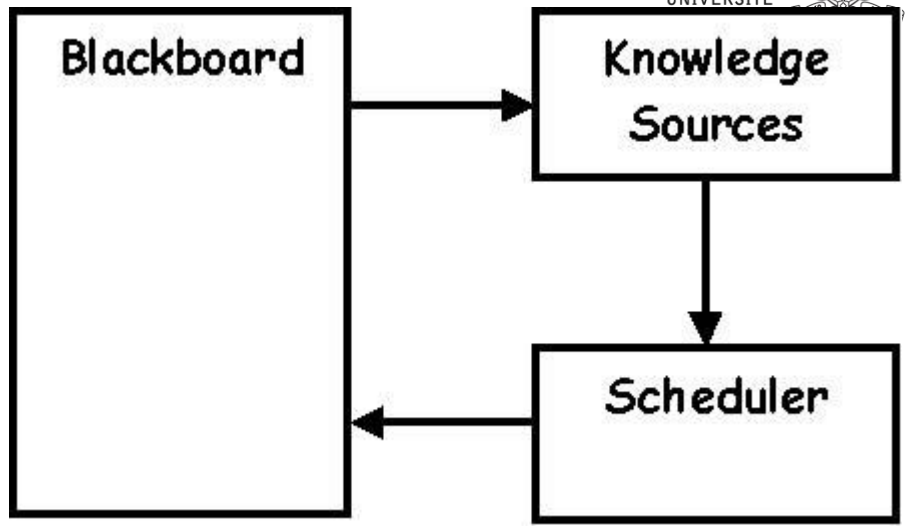


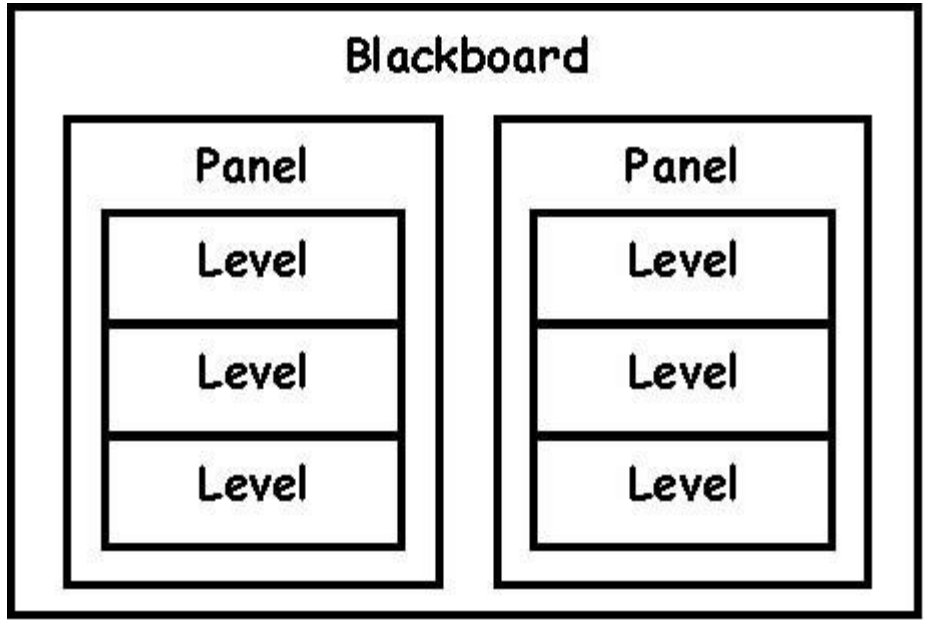
MVC – Avantages et Désavantages

- Avantages
  - Séparation des responsabilités.
  - Le modèle peut être partagé parmi divers vues et contrôleurs.
  - Il y a déjà plusieurs cadriciels prêts à étendre pour implémenter le MVC.
- Désavantages
  - L'architecture peut augmenter la complexité du système.
    - Il y a la possibilité de plusieurs mises à jours des composantes, pas vraiment nécessaires.
  - Les diverses alternatives et définitions peuvent créer de la confusion.

Blackboard

- Imaginez qu'on a un grand problème sur le tableau et un groupe d'étudiants. Chaque étudiant peut résoudre une partie du problème. On a aussi un professeur qui décide quel étudiant va écrire sur le tableau. Quand l'étudiant a fini, le processus est répété.
- Le tableau représente un grand problème qui peut être divisé en petites parties. Il fournit la structure de données partagée.
  - La structure peut être complexe avec divers représentations, des données décomposées et organisées en différents niveaux.
- Les étudiants sont les sources de connaissance. Ils ont l'expertise pour résoudre une partie.
  - Chaque source de connaissance ajoute à la solution du problème.
- Le professeur est le planificateur qui décide quelle expertise est nécessaire selon le travail qui est déjà complété.
  - Il réagit aux changements sur le tableau pour mettre les décisions.
- Exemple : reconnaissance de la parole
  - Radar Defence System (Microsoft): https://social.technet.microsoft.com/wiki/contents/articles/13461.blackboard-design-pattern-a-practical-c-example-radar-defence-system.aspx



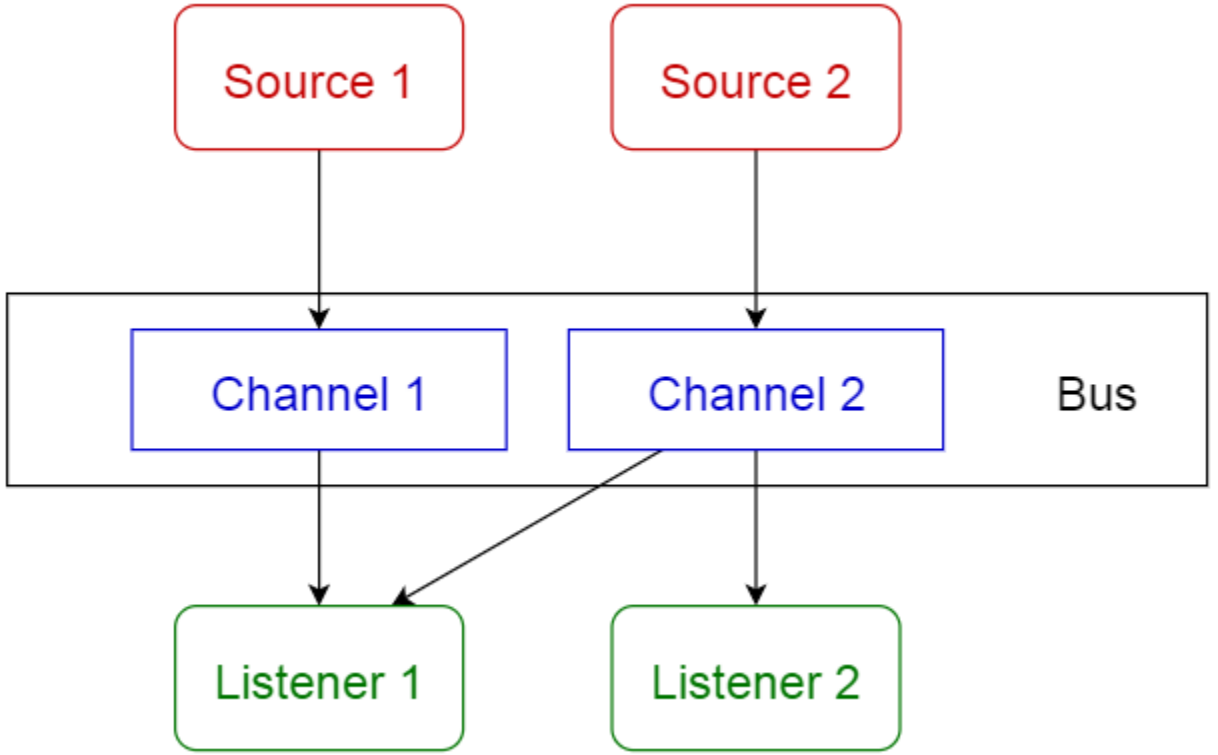


Blackboard – Avantages et Désavantages

- Avantages
  - Il peut fonctionner avec des problèmes grands, mais décomposables.
  - Il peut accommoder plusieurs « expertises ».
  - Il est facile d'ajouter de nouvelles sources de connaissance et d'étendre les structures de données.
- Désavantages
  - Le planificateur peut être lourd et complexe.
  - Comment va-t-on décider de l'ordre d'exécution?
  - La synchronisation et le contrôle d'accès sont nécessaire.
  - Il est difficile de modifier les structures de données.

Event-bus

- Si on a plusieurs modules qui veulent communiquer entre eux, le bus d'évènements peut gérer la communication.
- Les modules peuvent soumettre des messages directs, diffuser des messages ou attendre des messages d'un type spécifique (Publish-Subscribe).
- Les messages ont une en-tête, qui peut spécifier le receveur et le type du message, et un corps.
- Exemples : systèmes de notification, monitoring, interfaces graphiques
- Un peu comme le patron Observer.



Event-bus – Avantages et Désavantages

- Avantages
  - On peut ajouter ou supprimer des modules en cours d'exécution.
  - Les modules ne sont pas responsables de la livraison des messages.
- Désavantages
  - Problèmes d'évolutivité : Il y a un risque de dépassement de capacité, s'il y a beaucoup de messages pour un seul bus.
  - La synchronisation est aussi une considération critique.

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Architectures Distribuées

Architectures Distribuées

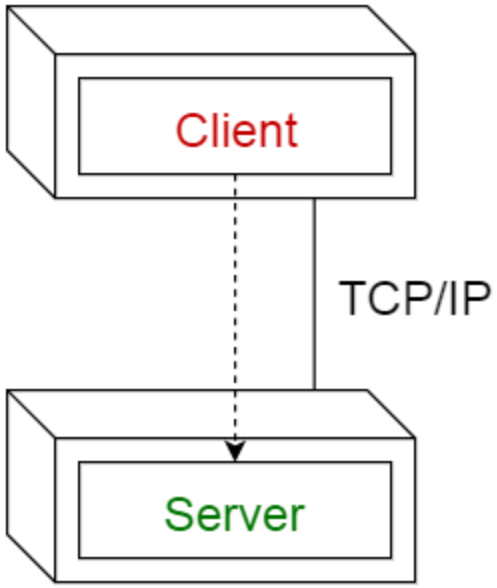
- Ce sont aussi des architectures modulaires, mais dans ce cas, les modules sont distribués parmi plusieurs fournisseurs et situés à différents endroits.
- L'importance du réseau est critique.
- Le développement est partagé parmi divers organisations, ce qui a un impact sur les aspect économiques liés au logiciel.
- Le logiciel devient plus flexible et dynamique.
- Les technologies avancent rapidement pour répondre aux défis posés par ces architectures.

Distribué – Avantages et Désavantages

- Avantages
  - Transparence : Plusieurs détails sont cachés, ce qui facilite le développement.
  - Partage des ressources (matériel et logiciel).
  - Ouverture : Flexibilité accrue pour utiliser divers fournisseurs de matériel et de logiciel.
  - Traitement concurrent pour améliorer la performance.
  - Évolutivité : Capacité augmentée en ajoutant de nouvelles ressources.
  - Tolérance aux fautes : Capacité de continuer à fonctionner après une la détection et la résolution d'une faute.
- Désavantages
  - Complexité augmentée.
  - Sécurité diminuée.
  - L'effort de gestion est augmenté.
  - Il y a des réponses et des effets imprévisibles par des parties du système.

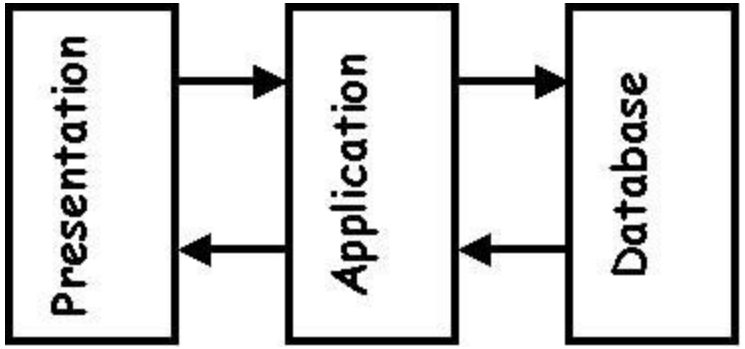
Client – serveur

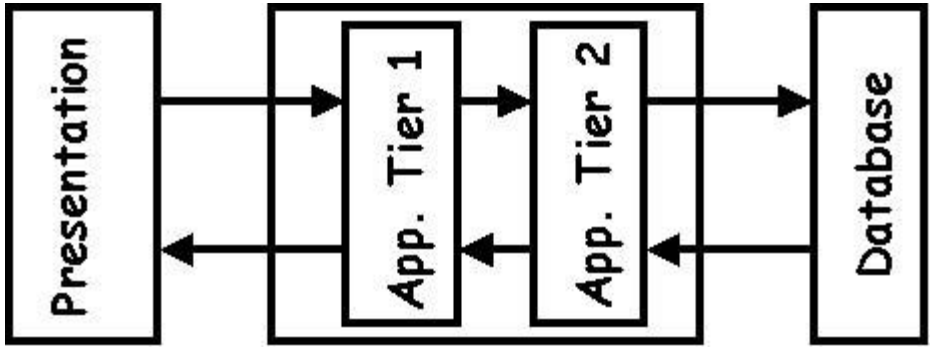
- Deux entités :
  - Un serveur, qui fournit la fonctionnalité (un service) et le client, qui est un consommateur du service.
  - Le client fait des demandes et le server répond.
- Le server peut servir plusieurs clients.
- La communication se passe par un réseau.
- Deux types possibles de client:
  - Client-léger (thin-client) : juste le rendu de l'interface graphique. Aucune logique. Terminal
  - Client-lourd (thick-client) : Le rendu de l'interface graphique et quelque partie de la logique. Navigateur
- Exemples : Presque toutes les applications web.
- L'architecture fondamentale de l'Internet.
- On ne peut pas parler d'avantages ou de désavantages parce que l'architecture est vraiment générique.
  - Des styles plus spécifiques ont des avantages ou des désavantages plus concrets.



Multitier

- C’est une extension de client – serveur.
- Au lieu de deux niveaux (client, serveur), on peut en avoir plusieurs.
  - Présentation (interface), Application (plusieurs niveaux), Base des données.
- Comme l’architecture en couches mais:
  - Les niveaux peuvent communiquer indépendamment entre eux.
  - Les niveaux peuvent être distribués sur plusieurs ordinateurs ou même entre plusieurs propriétaires.
- Les modules sont développés complètement indépendamment et ils sont très spécialisés.
- Exemples : Applications de commerce électronique, analyse de données, apprentissage automatique





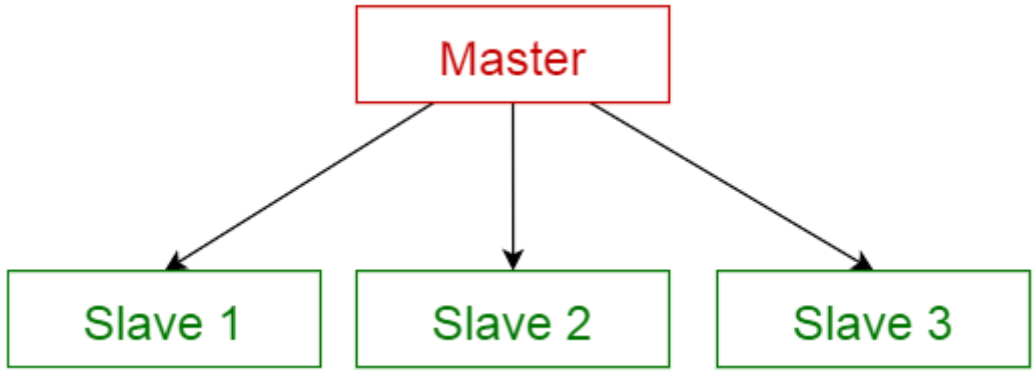
Multitier – Avantages et Désavantages

- Presque les mêmes que l'architecture en couches
- Avantages
  - Les technologies existent depuis longtemps et elles sont robustes.
  - Les modules sont réutilisables. Ils ne sont plus autonomes, mais vraiment indépendants.
  - L'échange dynamique des modules est possible et facile.
- Désavantages
  - Les modules peuvent être spécifiques à des technologies particulières et cela diminue la flexibilité.
  - La fonctionnalité et la disponibilité dépendent du réseau.
  - La sécurité devient complexe et réduite.

Master-slave

Pas bcp de fonctionnalités!

- Aussi, une architecture à deux niveaux, mais les rôles sont différents.
- On a un grand travail, qui peut être divisé et partagé.
- Le « maître » est responsable de diviser le travail et de distribuer les parties aux esclaves.
- Les « esclaves » exécute le travail sur des parties des données. En fait, les esclaves ont des programmes identiques.
- À la fin, le maître assemble les résultats et les stocke dans une base de données.
- Exemples : MapReduce, bases des données distribuées (NoSQL)



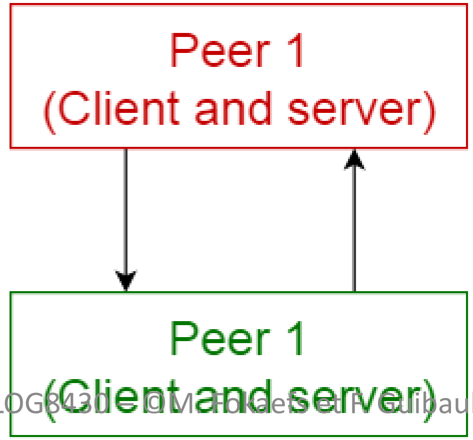
Master-slave – Avantages et Désavantages

- Avantages
  - La parallélisation augmente l'efficacité.
  - La présence du maître augmente la fiabilité du système.
  - Le résultat est garanti.
- Désavantages
  - L'architecture est sensible à la fiabilité du maître.
    - Il y a des solutions qui ont plusieurs maîtres (backup).
  - Les esclaves sont isolés. Il y a un risque de redondance.
  - L'architecture n'est pas utilisable sans une capacité suffisante de calcul.
  - N'est pas applicable à des problèmes qui ne sont pas décomposables.

Peer-to-peer

Peer to peer: les noeuds communiquent entre eux pour identifier des erreurs. Pas centralisée, robuste aux échecs des participans = tolérance aux fautes.

- Les données ne sont pas centralisées, mais tout le monde peut avoir toutes les données.
- Les données sont partagées par des connections directes entre les participants du réseau.
- Une structure centrale est possible, mais juste pour les adresses des participants ou pour conserver l'information sur qui possède quoi.
- Exemple : applications de partage de fichiers

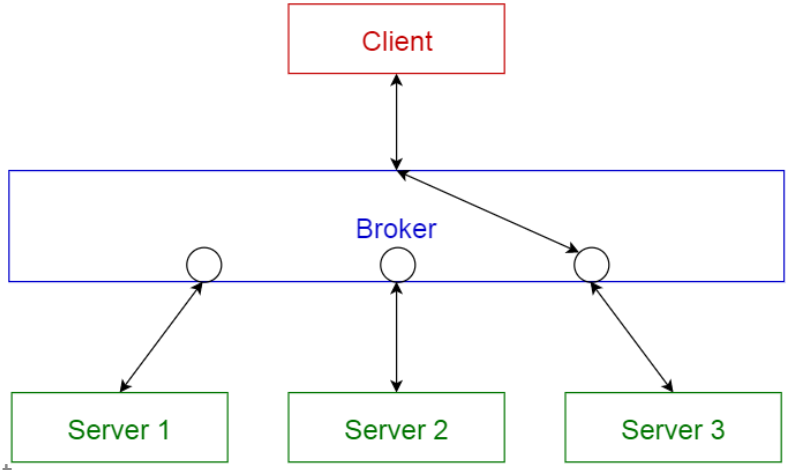


Peer-to-peer – Avantages et Désavantages

- Avantages
  - Le système est robuste aux échecs des participants.
  - L'architecture offre une évolutivité augmentée.
- Désavantages
  - La qualité, la performance et la sécurité ne sont pas garanties.

Broker

- On sait tout ce qu'on veut faire, mais on ne sait pas qui peut le faire et comment on peut les contacter.
- Le « courtier » est un service de découverte qui connait tous les fournisseurs et leurs services, dont la description est publiée.
- Les clients soumettent leurs exigences au courtier et le courtier est responsable de trouver le service qui remplit les exigences et d'établir une connexion directe entre le client et le service.
- Il est nécessaire pour les serveurs et les clients de soumettre les fonctionnalités de leurs services et les exigences respectivement dans un format spécifique
- C'est le prédécesseur de l'architecture orientée service.



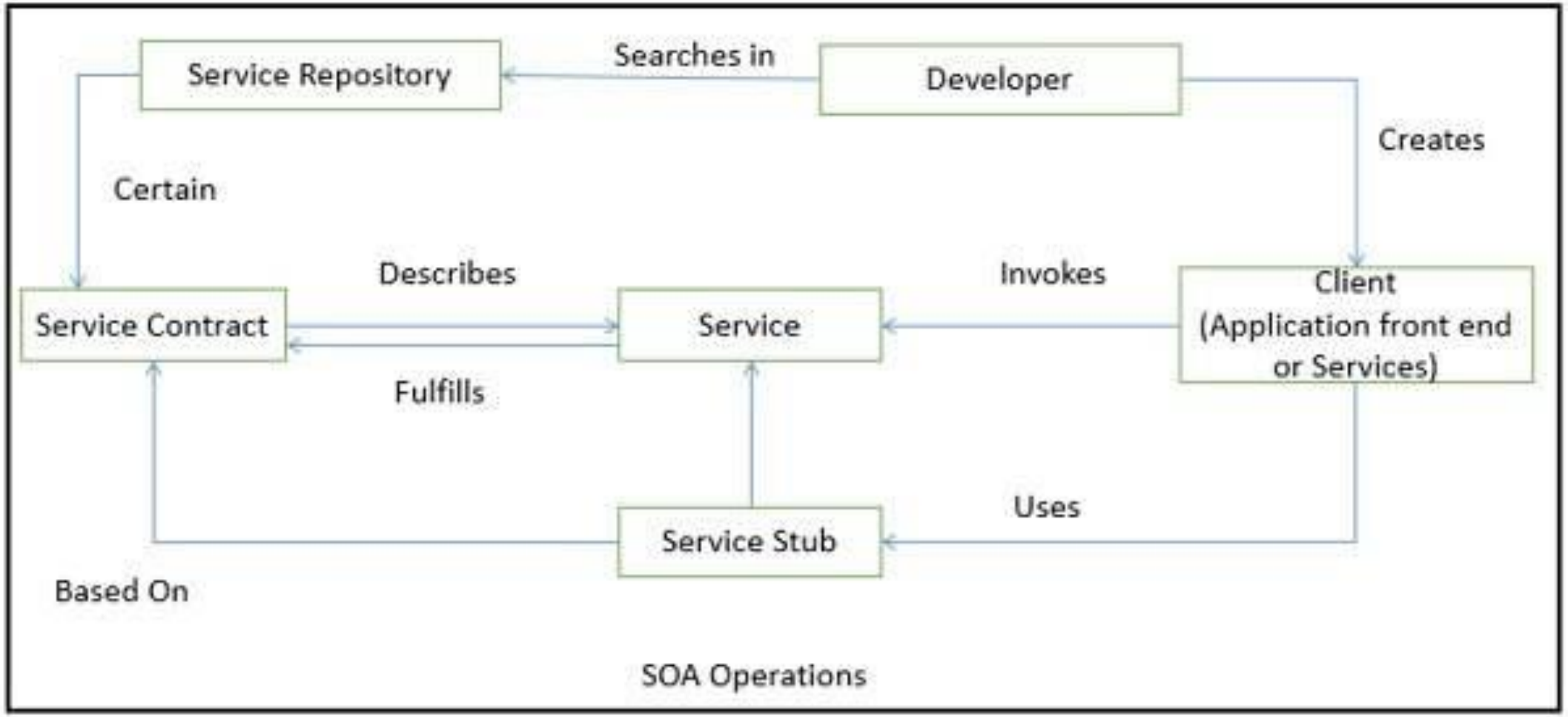
Broker – Avantages et Désavantages

- Avantages
  - Il est possible de dynamiquement ajouter, modifier ou supprimer des serveurs.
  - La distribution est transparente pour le développeur.
  - Le serveur et le client sont découplés.
- Désavantages
  - Les transactions et les exceptions doivent être gérées.

Service-oriented architectures SOA

- Tout est un service!
- Les services sont des logiciels autonomes et atomiques.
- Ils rendent possible l'exposition de l'expertise (fonctionnalité) et des données.
- SOAP (Simple Access Object Protocol) a été développé pour remplacer les vieilles technologies du courtier.
  - En fait, il a remplacé les messages en format simple par le XML.
- Le protocole est accompagné de plusieurs standards.
  - Pour la sécurité, la composition, le web sémantique.
- Le service est publié en tant qu'une interface XML.
  - L'interface spécifie la fonctionnalité et les types des données disponibles.
  - L'entente au niveau du service (SLA) spécifie les propriétés non-fonctionnelles.
- Le client utilise l'interface pour créer un « stub » dans le langage de l'application locale pour invoquer le service.

Service-oriented architectures



SOA – Avantages et Désavantages

- Avantages
  - La technologie promeut la composition, la réutilisation, l'interopérabilité.
  - Les clients ne dépendent pas d'une technologie spécifique.
    - Le service peut être implémenté en python, et le client en Java.
  - Il y a une abondance d'options et c'est possible d'échanger les services dynamiquement (en théorie).
  - Il y a aussi une abondance d'outils automatiques pour faciliter le développement et la maintenance des services et des clients.
- Désavantages
  - SOAP est très lourd.
  - Plusieurs technologies n'ont jamais marché.
    - La découverte des services automatique, le dépôt universel (UDDI) etc.

Representational state transfer

L'utilisateur peut simplement voir un produit, ajouter un produit, supprimer un produit ou mettre à jour un produit

- Tout est une ressource!
- En théorie, les services REST exposent juste des données et des structures des données (identifiées uniquement) et permettent des opérations simples sur ces structures (GET, PUT, POST, DELETE).
- La technologie utilise plusieurs formats de données (texte, XML, JSON).
- Une interface explicite n'est pas nécessaire.
- Une communication vers HTTP est requise.

SOAP vs REST

SOAP

- Standardisé
- Langage, plateforme, communication indépendant
- Support des outils augmenté
- Gestion des fautes et des exceptions fournie

REST

- Plus facile
- Plus simple
- Plus performant

REST (Respresentation state transfer) car nous sommes certain qu'on cherche pour une sorte d'architecture web, donc plus probablement client-serveur. Plus spécifiquement, c'est un bon cas pour le SOA ou multi-serveur (GUI, service, BD). Basé sur les identifiants uniques des produits, nous sommes amenés vers une solution SOA particulière; le REST qui fonctionne avec des URIs. Basé sur l'ensemble des opérations spécifiques (CRUD), SOAP serait trop complexe pour cette application. La différence enntre Microservices et REST peut être perçue comme l'une entre une architecture et son implémentation.

La prochaine fois

LOG2410

- Conception OO (SOLID)
- Patrons de Conception