CHAPITRE I: INTRODUCTION À L'ARCHITECTURE LOGICIELLE

Lamia Yessad

OBJECTIFS

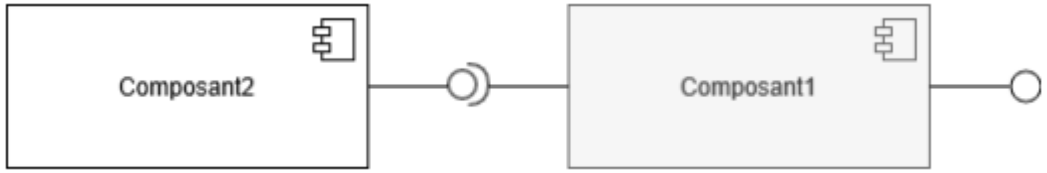
A la fin de ce cours, vous serez en mesure de:

- Comprendre l'architecture logicielle et le rôle de l'architecte.
- Utiliser UML pour décrire une architecture logicielle selon plusieurs vues.
- Expliquer d'importants styles architecturaux, leur fonctionnement et leurs principales caractéristiques de qualité.
- Comprendre comment se fait l'évaluation des architectures.

DÉFINITION

- Une architecture est « l'organisation fondamentale d'un système incarné par ses composants, les relations entre ses composants et leur lien avec l'environnement ainsi que les principes qui régissent sa conception et son évolution » (IEEE 42010: 2011).
- « L'architecture logicielle d'un programme ou d'un système informatique est la structure ou les structures du système, qui comprennent les composants logiciels, les propriétés visibles de l'extérieur de ces composants et les relations entre eux. L'architecture s'intéresse au aux interfaces publics; Les détails privés des éléments -les détails ayant uniquement trait à l'implémentation interne -ne sont pas architecturaux » (Bass et al. 2003).

CONCEPTS



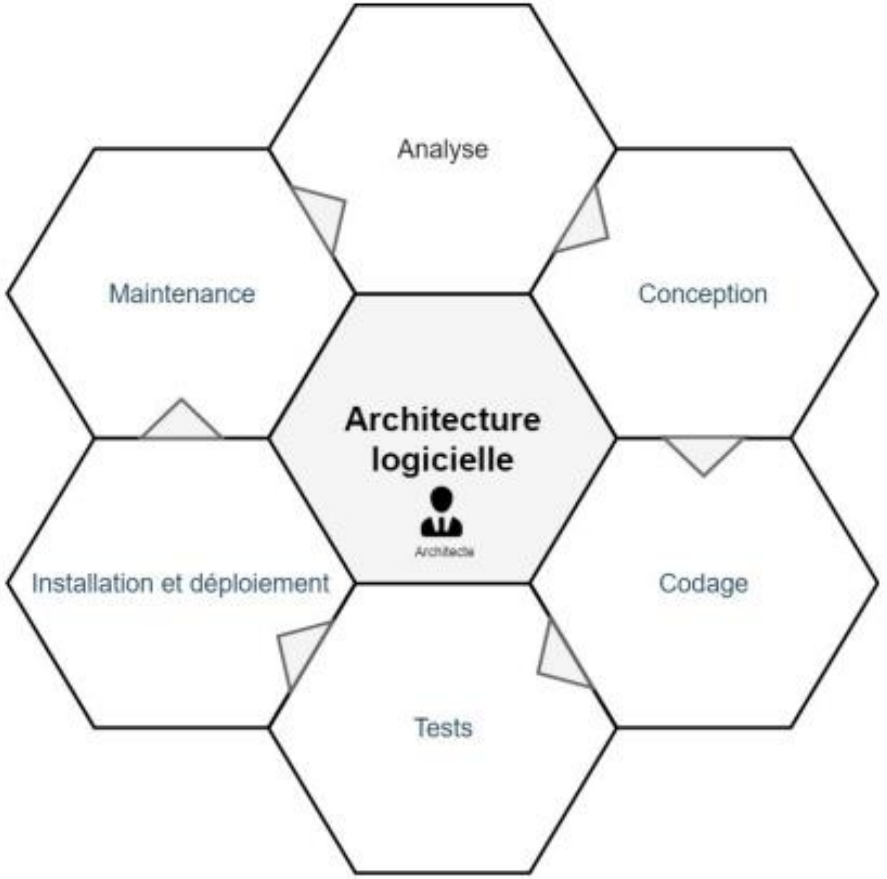
- Composant: Unité logicielle qui réalise une certaine fonction au moment de l'exécution. Exemples de composants: programmes, objets, processus, clients et serveurs, bases de données, etc.
- Interface: regroupe toutes les propriétés visibles à l'extérieur d'un composant (méthodes, qualité de service, protocole d'accès, etc). Il existe deux types d'interfaces:
  - Interface fournie
  - Interface requise

- Connecteur: Un mécanisme qui permet les interactions entre composants afin d'assurer la communication, la coordination ou la coopération entre eux.
  - Exemples de connecteurs: l'accès aux variables partagées, les appels de procédure, les appels de procédure distante, l'envoi de messages, les flux de données, etc.

DÉVELOPPEMENT LOGICIEL

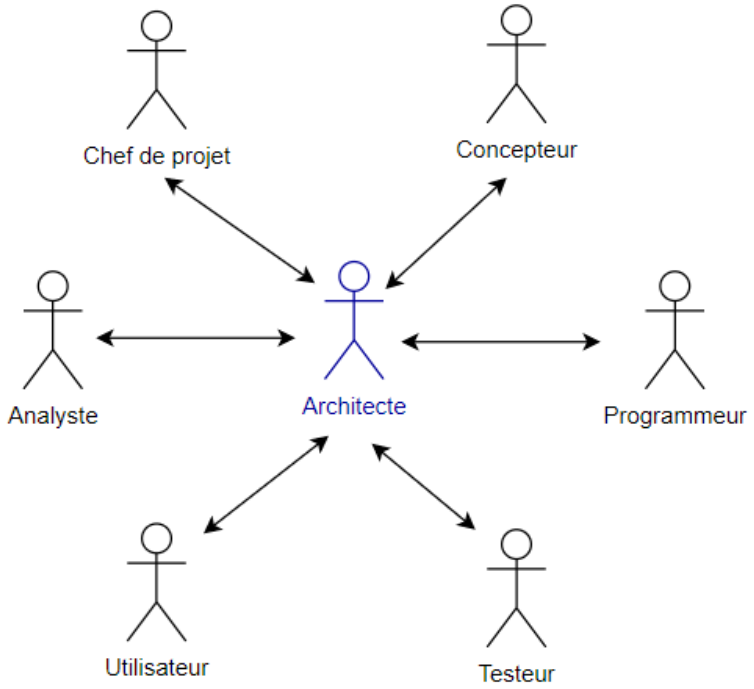
- La conception architecturale: aborde les problèmes concernant l'organisation des composants, les connecteurs entre composants, leur distribution physique, les caractéristiques de qualité, l'évolution de l'architecture et éventuellement la sélection entre plusieurs alternatives d'architecture.
- La conception détaillée: produit les algorithmes détaillés et les structures de données.
- Le stakeholder qui s'occupe de la conception architecturale est l'architecte.
- L'architecte joue un rôle central dans le cycle de développement et non seulement dans l'étape de conception architecturale.

ETAPE D'ARCHITECTURE LOGICIELLE



ARCHITECTE

- Un architecte est l'acteur qui crée une architecture et la maintient tout au long du cycle de vie.
- En plus des compétences techniques, un architecte doit:
  - Encourager les stakeholders (acteurs impliqués dans le projet) à exprimer leurs besoins et attentes
  - Comprendre la nature, la source et la priorité des besoins non fonctionnels
  - Avoir des compétences autres que techniques (Diplomatie, négociation et communication)
  - Maitriser le domaine d'application

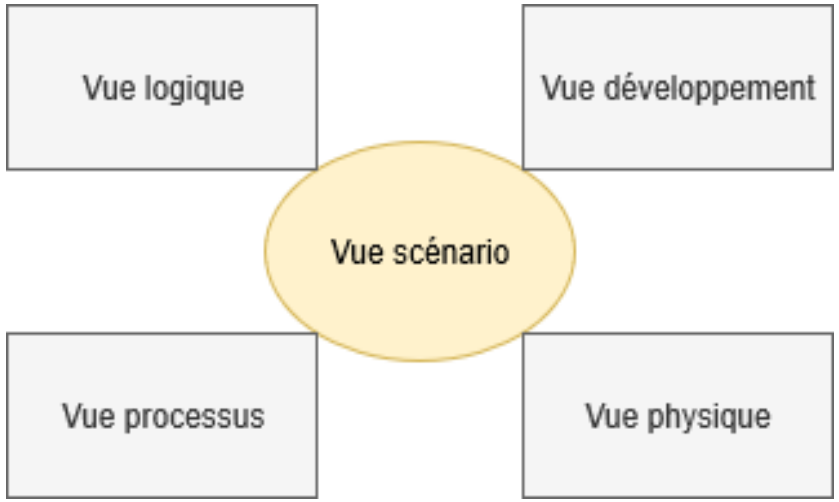


REPRÉSENTATION D'UNE ARCHITECTURE

Pour décrire une architecture logicielle, il faut:

- Décrire l'organisation générale d'un système et sa décomposition en sous-systèmes;
- Décrire les interfaces entre les sous-systèmes;
- Décrire les interactions entre les sous-systèmes;
- Décrire les composants utilisés pour réaliser les fonctionnalités des sous-systèmes;
- Décrire le déploiement des composants sur les dispositifs matériels.

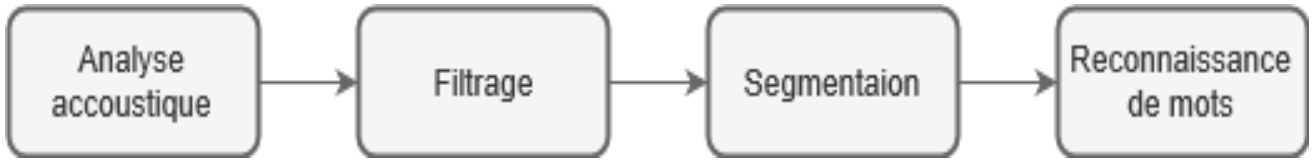
(4+1) VUES DE KRUCHTEN



- Diagramme de packages (vue logique);
- Diagramme de composants (vues développement et physique);
- Diagramme de déploiement (vue physique).

STYLE ARCHITECTURAL

- Un style architectural est un patron décrivant une architecture logicielle permettant de résoudre un problème particulier.
- Un style architectural représente un ensemble d'architectures qui ont des caractéristiques communes.
- Exemple d'un patron architectural : Cas du Système de transcription audio



STYLES EXISTANTS

Dans la suite de ce cours, quelques styles architecturaux sont étudiés. Chaque style est défini en déterminant les cinq points suivants:

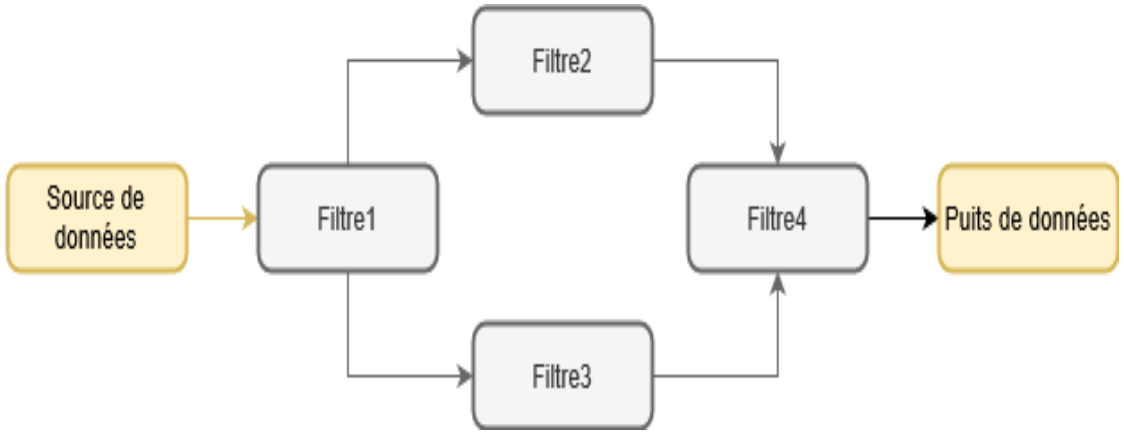
1. Le vocabulaire: déterminer quels sont les composants et les connecteurs. Dans le style de l'exemple, les composants sont des filtres et les connecteurs sont des tubes (pipes).
2. La topologie: c'est-à-dire la manière d'interconnecter les différents composants. Dans l'exemple précédent, la topologie est un graphe orienté où les nœuds sont les filtres et les arcs sont les pipes.
3. Le fonctionnement: décrire comment s'exécute le système qui suit un tel style.
4. Les plus importantes caractéristiques de qualité : ces caractéristiques sont dérivées des avantages et des inconvénients de l'utilisation du style.
5. Des exemples: Le système de transcription audio est l'un des exemples. L'exemple du Shell UNIX en est un autre et sera vu en TD.

PIPE-AND-FILTER

- Le vocabulaire: Le filtre est le composant qui possède les propriétés suivantes:
  - Indépendance: les filtres ne doivent pas partager d'état ni de variables globales.
  - Anonymat: Un filtre ne connait pas les filtres en amont et en aval.
  - Concurrence: plusieurs filtres peuvent s'exécuter en parallèle.

  Le pipe est un mécanisme qui permet de rediriger la sortie d'un filtre vers l'entrée d'un autre.

- La topologie:

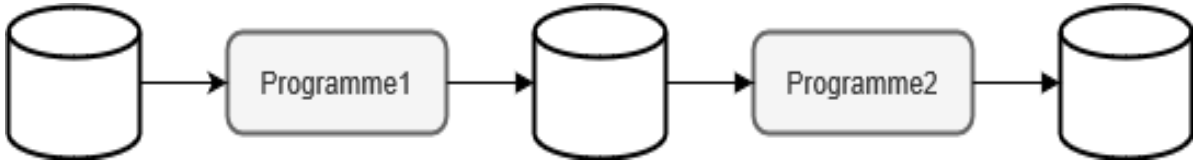


PIPE-AND-FILTER (SUITE)

- Le fonctionnement: la source de données produit les données en entrée (Inputs) qui seront acheminées (grâce aux pipes) de filtre en filtre subissant à chaque fois des transformations (Outputs) jusqu'à destination (puits de données).
- Les caractéristiques de qualité: réutilisabilité, maintenabilité, évolutivité, traitement d'énormes flux de données (audio, vidéo, fichiers) et parallélisme. Difficulté de synchronisation et manque d'interactivité.
- Exemple:

BATCH SEQUENTIAL (SÉQUENTIEL BATCH)

- Le vocabulaire: les composants sont des programmes indépendants et les connecteurs sont des médias (fichiers, bandes magnétiques, …).
- La topologie:



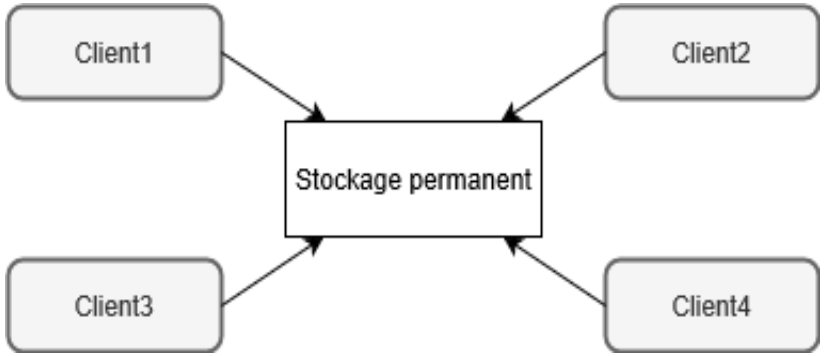
- Le fonctionnement: similaire au pipe-and-filter avec la différence qu'un programme ne peut commencer son exécution avant que le programme qui le précède ne termine la sienne.

BATCH SEQUENTIAL (SUITE)

- Les caractéristiques de qualité: réutilisabilité et maintenabilité mais sans possibilité de parallélisme ce qui peut réduire considérablement les performances. Aussi, ce style n'est pas conçu pour traiter des flux de données illimités. Cependant, il élimine les problèmes liés à la synchronisation.
- Exemple: Ce style est utilisé dans les traitements scientifiques et traitements métier (transactions d'un type particulier survenant à des intervalles réguliers).

REPOSITORY

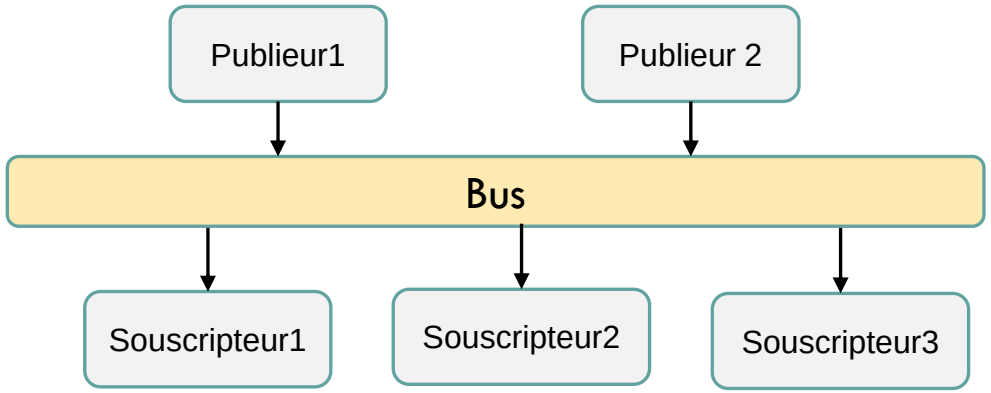
- Le vocabulaire: Les composants sont le stockage partagé et les différents clients (ou accesseurs de données). Le connecteur est l'accès direct au stockage.
- La topologie:



- Le fonctionnement: Le stockage est passif alors que les clients sont actifs. Chaque client est un composant indépendant qui accède au stockage.

EVENT-BUS

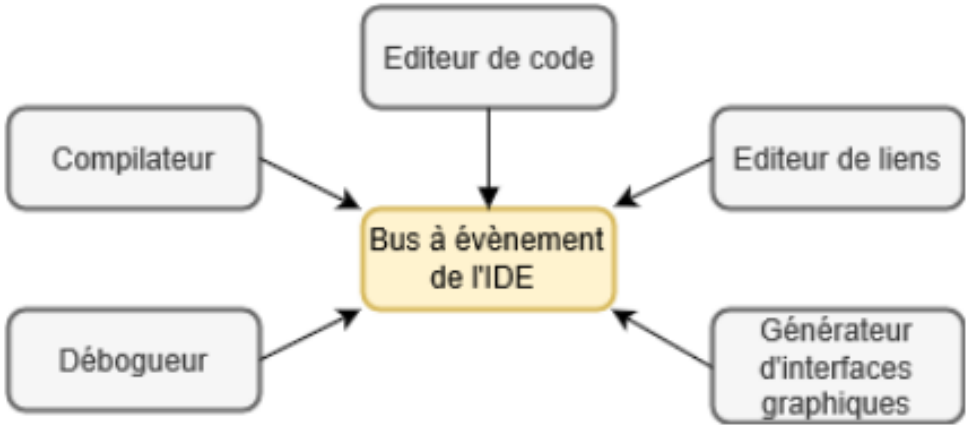
- Vocabulaire: Les composants sont les publieurs, les souscripteurs et le bus et les connecteurs sont des annonces d'événements et des appels de routines.
- Topologie:



- Fonctionnement: Les publieurs annoncent des événements et ne connaissent pas quels souscripteurs sont concernés par ces événements. Aucune hypothèse n'est possible sur la nature et l'ordre des traitements déclenchés. Les communications sont donc asynchrones et implicites.

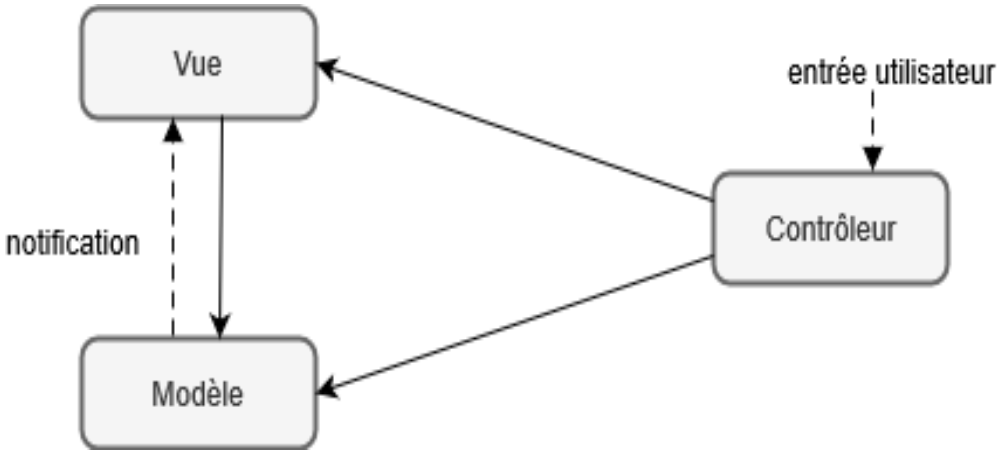
EVENT-BUS (SUITE)

- Les caractéristiques de qualité: Les différents composants peuvent être ajoutés et remplacés facilement et dynamiquement (réutilisation et maintenabilité). Cependant, le contrôle est absent et la surcharge du bus.
- Exemple: les interfaces utilisateur et les environnements de programmation intégrés.



MVC

- Le vocabulaire: Acronyme de Model View Controller, on a donc les trois composants:
  - Le modèle (Model): contient les fonctionnalités et les données;
  - La vue (View): affiche les informations à l'utilisateur (plusieurs vues peuvent être définies);
  - Le contrôleur (Controller): traite l'entrée de l'utilisateur.
- La topologie:

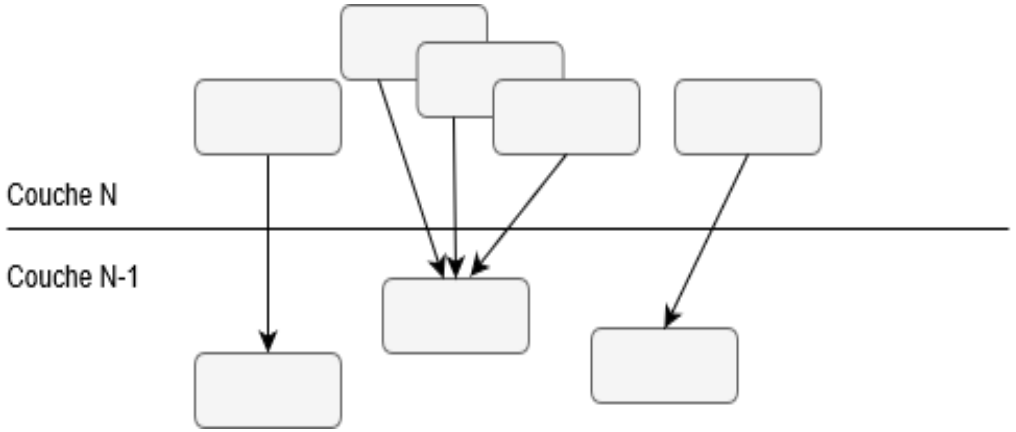


MVC (SUITE)

- Fonctionnement: la vue et le modèle communique indirectement à travers le contrôleur ou directement grâce à la notification. La vue accède au modèle pour se mettre à jour.
- Les caractéristiques de qualité: MVC s'adapte aux applications interactives. Il favorise la séparation des tâches et la maintenance. Cependant, la vue et le contrôleur sont étroitement liés et l'augmentation de la complexité de l'application peut être problématique.
- Exemple: les applications où les utilisateurs interagissent avec le système à travers une interface graphique.

STYLE EN COUCHES

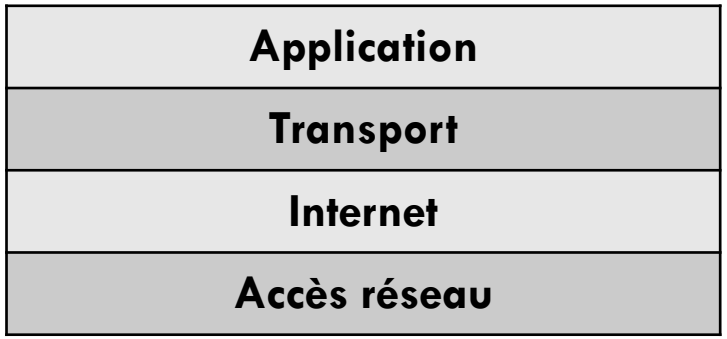
- Le vocabulaire: les composants sont les sous-tâches appartenant aux différentes couches et les connecteurs sont les appels entre tâches.
- La topologie:



- Fonctionnement: chaque couche contient un ensemble de sous-tâches qui rendent des services à la couche au-dessus.

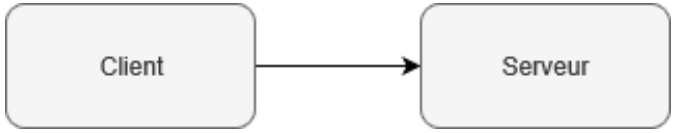
STYLE EN COUCHES (SUITE)

- Les caractéristiques de qualité: ce style s'adapte aux logiciels qui ont plusieurs niveaux d'abstraction. La modification dans l'une des couches n'affectent que la ou les couches supérieures. Le remplacement de l'implémentation d'une couche avec les mêmes interfaces est facile. Cependant, la décomposition en couches d'abstraction n'est pas évidente et la performance risque d'altérer si le choix de composition n'est pas judicieux.
- Exemple: modèles de communication (OSI, TCP/IP), les systèmes d'exploitation et les SGBD.



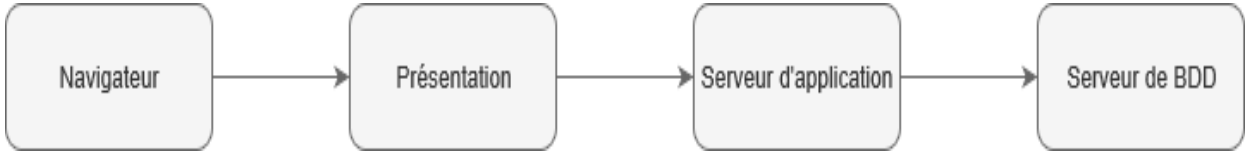
STYLE EN TIERS

- Le vocabulaire: les composants sont les tiers (clients et serveurs) et les connecteurs sont les protocoles ou les appels de procédures,…
- La topologie:
  - 2-tiers (Client/Serveur)



  - 3 tiers (N-tiers)

  - 4-tiers

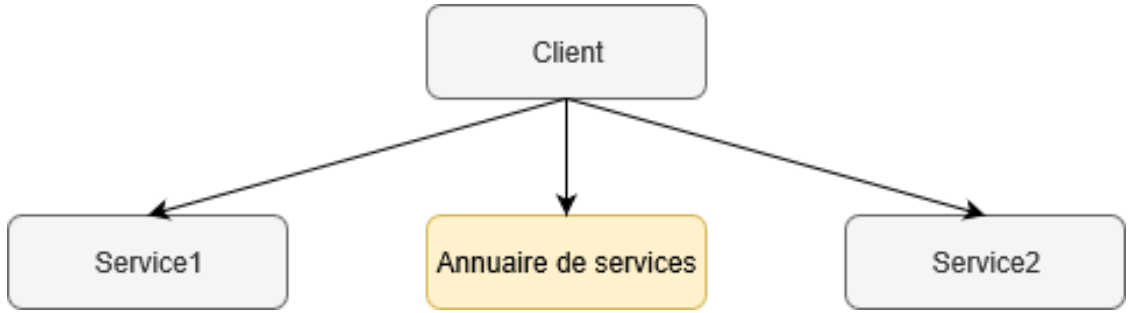


STYLE EN TIERS (SUITE)

- Fonctionnement: chaque tier communique avec le tier adjacent pour obtenir un service. Un tier peut être décomposé en couches. Une couche peut être déployée sur un ou plusieurs tiers.
- Les caractéristiques de qualité: Ce style s'adapte aux applications Web. Facile à mettre en œuvre mais la complexité s'élève à l'ajout des niveaux intermédiaires.
- Exemple: Web App.

STYLE SOA

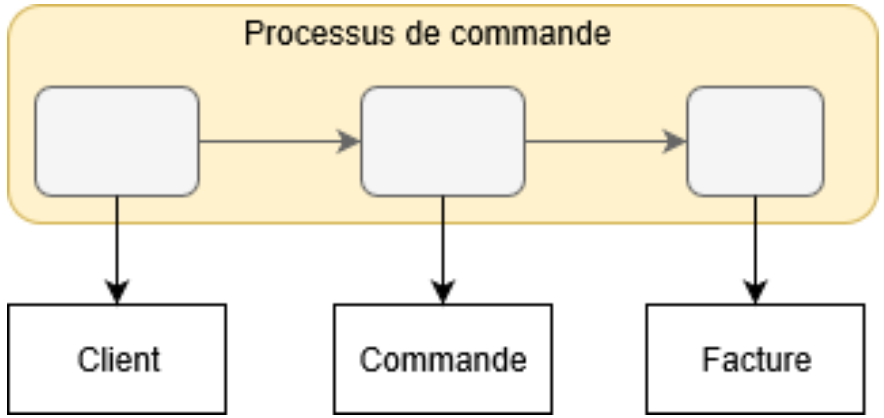
- Vocabulaire: Les composants de la SOA (son acronyme) sont des services (y compris l'annuaire) et les connecteurs entre services sont des protocoles orientés message tels que SOAP.
- Topologie:



- Fonctionnement : Un client appelle un service soit directement s'il possède les informations techniques de ce service ou bien accède à l'annuaire pour récupérer ces informations. Ces dernières sont utilisées pour pouvoir effectuer l'appel requis.

STYLE SOA (SUITE)

- Caractéristiques de qualité : Le service possède un certain nombre de caractéristiques (indépendant, neutre, sans état, faiblement couplé, ...) ce qui favorise le couplage faible et la réutilisabilité. L'utilisation des standards du Web rend cette architecture interopérable et scalable .
- Exemple : Dans les systèmes d'information, un service est une fonctionnalité métier bien définie et autonome. Il est indépendant des autres service, publié et prêt à être utilisé via une interface standard.



ÉVALUATION D'UNE ARCHITECTURE

- Les besoins non fonctionnels sont principalement un sous ensemble de caractéristiques ou d'attributs de qualité. Ils répondent à la question: Comment le système réalise ses besoins fonctionnels?
- Il existe plusieurs modèles de qualité: ISO 9126, ISO 25000, McCall, Boehm, indicateurs KPIs, ...
- Par exemple, le modèle ISO 25000:2014 définit les caractéristiques suivantes: Fonctionnalité, Fiabilité, Utilisatibilité, Efficience, Maintenabilité et Portabilité.
- Il existe plusieurs méthodes d'évaluation d'architecture telles que: ATAM, SAAM et ARID.
- L'architecture sélectionnée est celle qui répond, au mieux, aux besoins de qualité exprimés par les différents stakeholders. Un compromis est souvent réalisé par l'architecte.

QUALITÉ LOGICIELLE: UNE NOTION MULTIVALUÉE

- Fiabilité : la capacité d'un logiciel à rendre des résultats corrects dans des conditions de fonctionnement normales.
- Efficience : la capacité d'un logiciel à fonctionner en optimisant l'utilisation de ressources. Il faut faire la distinction avec l'efficacité qui désigne la réalisation des objectifs à atteindre.
- Utilisabilité : la capacité d'un logiciel à être facilement utilisé.
- Maintenabilité : la capacité d'un logiciel à être facilement maintenue. La maintenance peut être corrective, évolutive, adaptative (maintenance curative) et/ou préventive.
- Portabilité : la capacité d'un logiciel à être porté d'un environnement (ex. Système d'exploitation) à l'autre.

QUALITÉ LOGICIELLE (SUITE)

- Réutilisabilité: La capacité d'un logiciel à être utilisé pour développer d'autres logiciels.
- Evolutivité: La capacité d'un logiciel à prendre en compte les évolutions futures de besoins.
- Interactivité: La capacité d'un logiciel à laisser interagir l'utilisateur.
- Scalabilité: La capacité d'un logiciel à s'adapter à un changement d'ordre de grandeur de la demande (demandes de plus en plus croissantes des utilisateurs). Cette caractéristique est aussi appelée « Passage à l'échelle ».
- Interopérabilité: la capacité d'un logiciel à communiquer avec d'autres logiciels hétérogènes.

CONCLUSION

- L'architecture logicielle est au centre du cycle de vie.
- L'architecture logicielle est décrite à l'aide de plusieurs modèles (Ex. Diagrammes UML).
- Le style architectural est un patron issu d'architectures similaires.
- Le choix du style architectural est une décision prise lors de l'élaboration de l'architecture logicielle.
- Une architecture logicielle combine plusieurs styles architecturaux.
- Il existe plusieurs méthodes pour évaluer une architecture.