Ingénierie Logicielle

Conception et Architectures Logicielles

Prof. Ousmane SALL, Maître de Conférences CAMES en Informatique

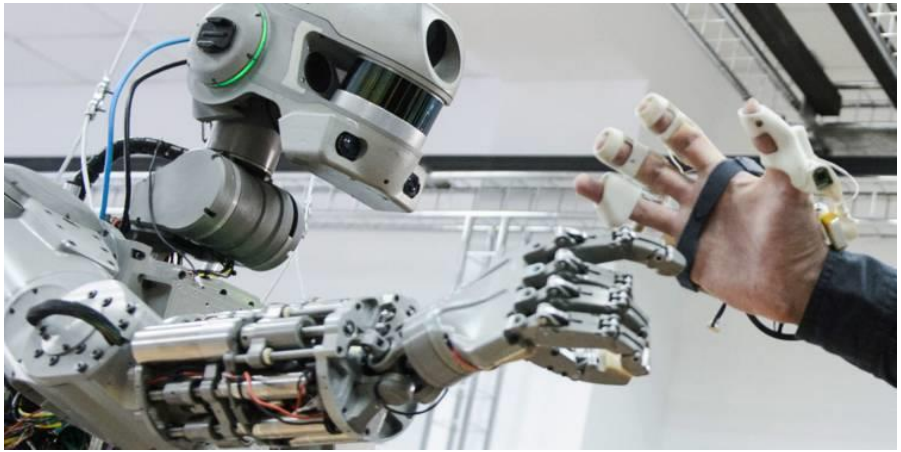
Objectifs de la séquence

- A l'issue de cette séquence vous serez capable de :
  - comprendre les concepts requises des besoins de l'utilisateur et du système et pourquoi ces besoins doivent être rédigées de différentes manières;
  - comprendre les différences entre les exigences logicielles fonctionnelles et non fonctionnelles;
  - comprendre comment les besoins peuvent être organisées dans un document des besoins de logiciel;
  - comprendre les principales activités d'ingénierie des besoins en matière de spécification, d'analyse et de validation, ainsi que les relations entre ces activités;
  - comprendre pourquoi la gestion des besoins est nécessaire et comment elle prend en charge d'autres activités d'ingénierie des besoins.

Remarques

- Ce cours utilise comme support le livre de Ian Sommerville *«Software Engineering»* mais aussi le cours *« IFT2255 - Génie logiciel »* de Bruno DUFOUR de l'Université de Montréal
  http://www.iro.umontreal.ca/~dufour/cours/ift2255/

Les logiciels et la société



Génie logiciel

Le **génie logiciel** est un domaine des sciences de l'ingénieur dont l'objet d'étude est la conception, la fabrication, et la maintenance des systèmes informatiques complexes. 3 févr. 2011

Génie Logiciel Avancé Cours 1 — Introduction - Stefano Zacchiroli
https://upsilon.cc/~zack/teaching/1011/gla/cours-01.pdf

Le génie logiciel touche au cycle de vie des logiciels. Toutes les phases de la création d'un logiciel informatique y sont enseignées : l'analyse du besoin, l'élaboration des spécifications, la **conceptualisation du mécanisme interne au logiciel** ainsi que les techniques de programmation, le développement, la phase de test et finalement la maintenance.
https://fr.wikipedia.org/wiki/Génie_logiciel

Conception et Architectures Logicielles:
Conception

L'activité de conception

- Étape cruciale du développement logiciel
  - analyse → conception → implémentation
  - décomposition du produit en unités (ex: modules, classes) plus simples
- Difficile: exige de la créativité
- Un bonne conception contribue à la qualité du logiciel
  - fiabilité
  - correction
  - facilité d'évolution et maintenance

Tâches de la phase de conception

- Concevoir l'**Architecture**
  - « Qui » fera « quoi » et « où » ?
- Concevoir les interfaces utilisateurs du logiciel
- Concevoir les bases de données
- Concevoir les contrôles du logiciel
  - Mise en œuvre de tous les aspects liés au contrôle, à la correction, à la sécurité, à la tolérance aux fautes, à la protection des données, etc.
- Concevoir le réseau
  - Communication entre les processus distribués

Selon David Garlan, l'architecture logicielle est le pont qui relie les exigences et l'implémentation.

Types de conception

- **Conception architecturale (haut niveau)**
  - Définit la structure et l'organisation générale du logiciel demandé
  - Décrit les principaux modules, les relations entre eux, les contraintes à respecter
- Conception détaillée (bas niveau)
  - Réalise chacun des cas d'utilisation
  - Respecte le plan de la conception architecturale
  - Décrit le fonctionnement interne de chacun des modules
  - Permet l'implémentation dans un language de programmation

Objectifs d'une bonne conception

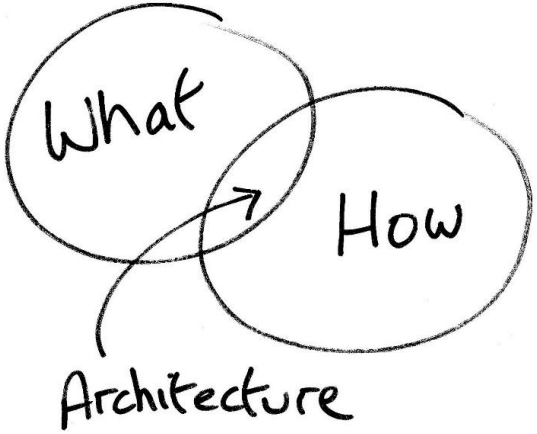
- **Forte cohésion** : les éléments ne sont pas réunis dans un même module (ou une même classe) par hasard, ils forment un tout pour réaliser une tâche
- **Faible couplage** : les modules (ou classes) sont relativement indépendants, ils ne dépendent le moins possible des éléments d'autres modules
- **Abstraction** : une décomposition intuitive et qui permet de se concentrer sur un module (ou classe) à la fois
- **Encapsulation et masquage d'information** : les détails d'implémentation sujets au changements sont cachés derrière une interface stable.

Conception et maintenance

- La conception doit tenir de compte
  - des besoins existants
  - des besoins à venir
- Objectif : obtenir une conception qui facilite l'adaptation du logiciel aux changements
  - capacité d'évolution
  - anticipation du changement

Types de changements

- Changement d'algorithme / de fonctionnalité
- Changement de la représentation des données
- Changement au niveau de la machine abstraite sous-jacente
- Changement au niveau des périphériques
- Changement de l'environnement
- Changements dus au processus de développement
- ...



Conception et Architectures Logicielles:
Architecture logicielle

Architecture logicielle

- Décrit
  - L'organisation générale du logiciel
  - Les **modules** et leurs **interfaces**
  - L'agencement du logiciel des sous-systèmes, des modules et des composants
    - Leurs propriétés
    - Leur composition («contient »)
    - Leurs collaborations («utilise»)
- Dépend des besoins fonctionnels et non fonctionnels du logiciel

Les besoins fonctionnels répondent aux points précis du cahier des charges, et sont donc requis par le client. Ils ne sont pas négociables en général, c'est le "besoin primaire" du client. Les besoins non-fonctionnels sont soit des besoins optionnels, soit des besoins/contraintes liés à l'implémentation (contraintes de langage ou de plate-forme, par exemple) et à l'interopérabilité générale (ne pas bouffer toutes les ressources de la machine par exemple).
Ils peuvent être fixés par le client (fonctions optionnelles), ou par le développeur (contraintes d'implémentation).

Architecture logicielle

- Comprendre l'architecture logicielle
- Exemples d'architecture logicielle
- Présentations d'architecture
- Points de vue et modèles de vue
- Évaluation de l'architecture?
- Établissement de la structure globale d'un logiciel
- Objectifs
  - Présenter la conception architecturale et discuter de son importance
  - Expliquer pourquoi plusieurs modèles sont nécessaires pour documenter une architecture logicielle
  - Décrire les types de modèles architecturaux pouvant être utilisés

Qu’est-ce que la conception architecturale ?

- La conception architecturale consiste à comprendre comment un logiciel doit être organisé et à en concevoir la structure globale.
- La conception architecturale est le lien essentiel entre la conception et l'ingénierie des besoins, car elle identifie les principaux composants structurels d'un système et leurs relations.
- La sortie du processus de conception architecturale est un modèle architectural qui décrit la manière dont le système est organisé en tant qu'ensemble de composants communicants.

Ian Sommerville, Chapitre 6

Conception architecturale: Pourquoi est-ce nécessaire ?

- Selon Bass et al. (2003) les avantages de la conception et de la documentation d'architecture explicites sont les suivants:
  - **Communication avec les parties prenantes:** L'architecture peut être utilisée comme sujet de discussion par les parties prenantes du système.
  - **L'analyse du système:** Signifie qu'il est possible d'analyser si le système peut satisfaire à ses exigences non fonctionnelles.
  - **Réutilisation à grande échelle**
    - L'architecture peut être réutilisable sur une gamme de systèmes
    - Des architectures de ligne de produits peuvent être développées.

Software Architecture in Practice, 2nd ed. This is a practical discussion of software architectures that does not oversell the benefits of architectural design. It provides a clear business rationale explaining why architectures are important. (L. Bass, P. Clements and R. Kazman, Addison-Wesley, 2003.)

Représentation de l'architecture

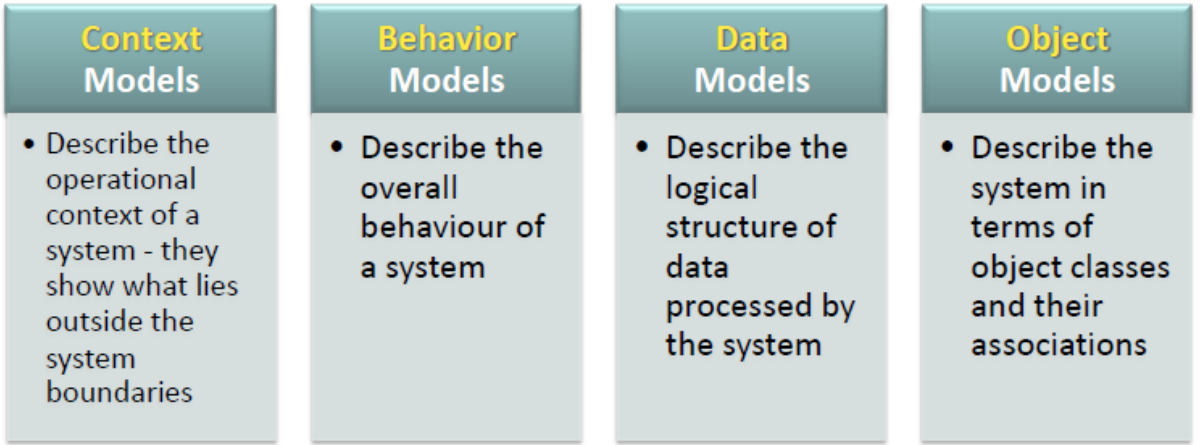
- Les schémas fonctionnels simples et informels illustrant les entités et les relations constituent la méthode la plus fréquemment utilisée pour documenter les architectures logicielles.
  - Dépend de l'utilisation de modèles architecturaux.
  - Les exigences pour la sémantique des modèles dépendent de la manière dont les modèles sont utilisés.
  - Utile pour la communication avec les parties prenantes et pour la planification de projet.

Utilisation de modèles architecturaux

- **Pour faciliter la discussion sur la conception du système**
  - Une vue architecturale de haut niveau d'un système est utile pour la communication avec les parties prenantes du système et la planification de projet, car elle n'est pas encombrée de détails.
  - Les parties prenantes peuvent établir des liens avec elle et comprendre une vue abstraite du système. Ils peuvent ensuite discuter du système dans son ensemble sans se confondre avec les détails.
- Pour documenter la conception d'une architecture
  - L'objectif ici est de produire un modèle de système complet montrant les différents composants d'un système, leurs interfaces et leurs connexions.

Modélisation

- Un modèle est une vue abstraite d'un système
- Nous créons un modèle pour mieux comprendre une entité, par exemple un modèle d'avion est un petit avion.
- Lorsque l'entité est un logiciel, le modèle prend une forme différente
- Types de modèles:
  - Modèles de contexte
  - Modèles de comportement
  - Modèles de données
  - Modèles d'objet



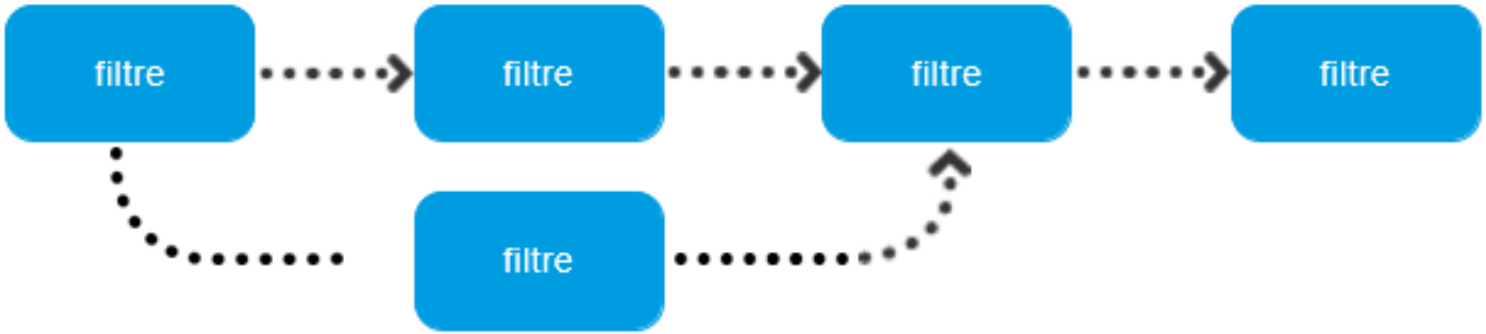
Utilisation de modèles architecturaux

- Un modèle de d'architecture doit être capable de représenter:
  - Les informations que le logiciel transforme,
  - Les fonctions qui permettent la transformation, et
  - Le comportement du système lors de la transformation.

Conception et Architectures Logicielles :
Types d'Architectures

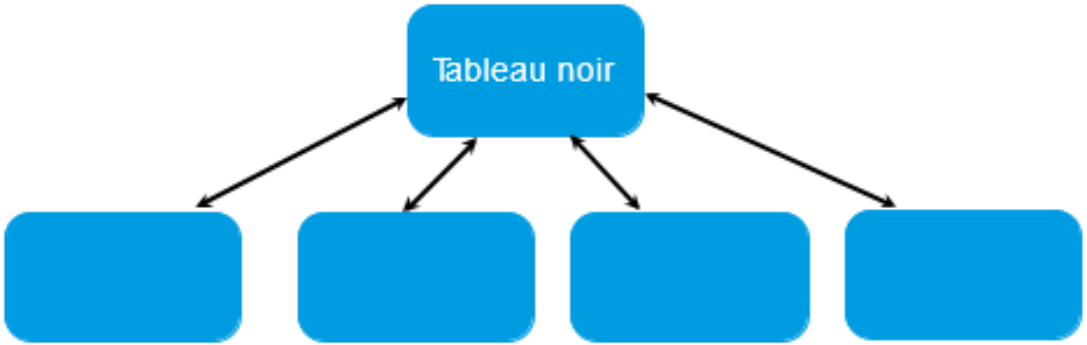
Types d'architectures: Architecture en pipeline

- Un **Pipeline** (ou chaîne de traitement)



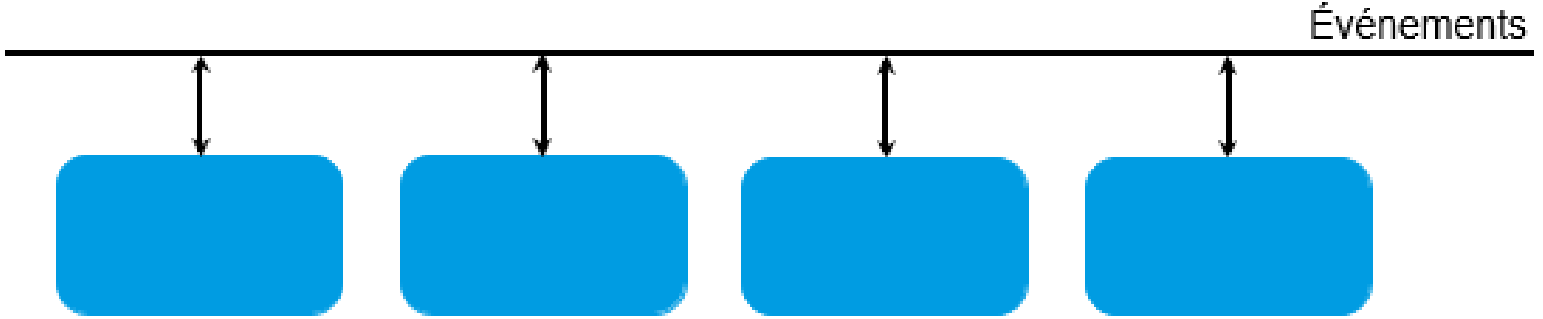
Types d'architectures: Architecture tableau noir

- **Tableau noir** : médium de communication
- Communication étendue à tous les partenaires
- Un des sous-système est désigné comme le tableau noir
- On ne peut recevoir et transmettre des informations que via le tableau
- Utile lorsqu'on ne connaît pas de solution déterministe



Types d'architectures: Architecture par événements

- Les sous-systèmes répondent aux événements
- Appropriée pour les logiciels dont les composants doivent interagir avec l'environnement
- Les sous-systèmes s'abonnent pour recevoir les annonces de certains types d'événements

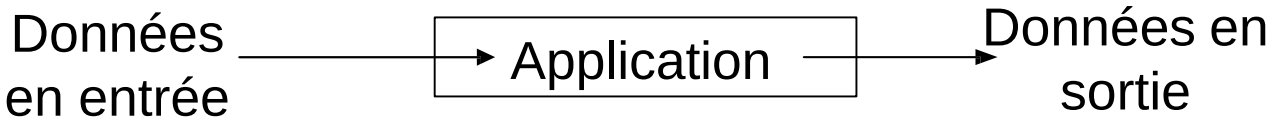


Types d'architectures: Architecture en couches (aussi appelée architecture multi-tiers)

- **Types d'Architectures:**
  - L'architecture en couches (aussi appelée architecture multi-tiers) est une pratique d'architecture logicielle qui propose de concevoir le système comme une superposition de strates, chaque strate étant définit par une responsabilité spécifique.
  - Cette pratique a commencé avec l'architecture client-serveur qui décrivait de quelle façon plusieurs applications ou plusieurs instances de la même application pouvaient partager la même base de données. (Voir Wikipédia client-serveur)
  - L'étape suivante fut de centraliser aussi les règles d'affaires dans le but de résoudre les problèmes inhérents aux applications distribuées. C'est ce qu'on a appelé l'architecture 3-tiers. (Voir Wikipédia Architecture trois tiers)
  - Le concept a évolué pour identifier et distinguer un nombre de couches de plus en plus nombreuses et de plus en plus fines. (Voir L'architecture à 5 couches)

Architecture Multi-tiers

- Traditionnellement une application informatique est un programme exécutable sur une machine qui représente la logique de traitement des données manipulées par l'application.
- Ces données peuvent être saisies interactivement via l'interface ou lues depuis un disque. L'application émet un résultat sous forme de données qui sont, soit affichées, soit enregistrées sur un disque.
- Dans ce schéma, les traitements, les données d'entrées, les données de sortie sont sur une seule machine.

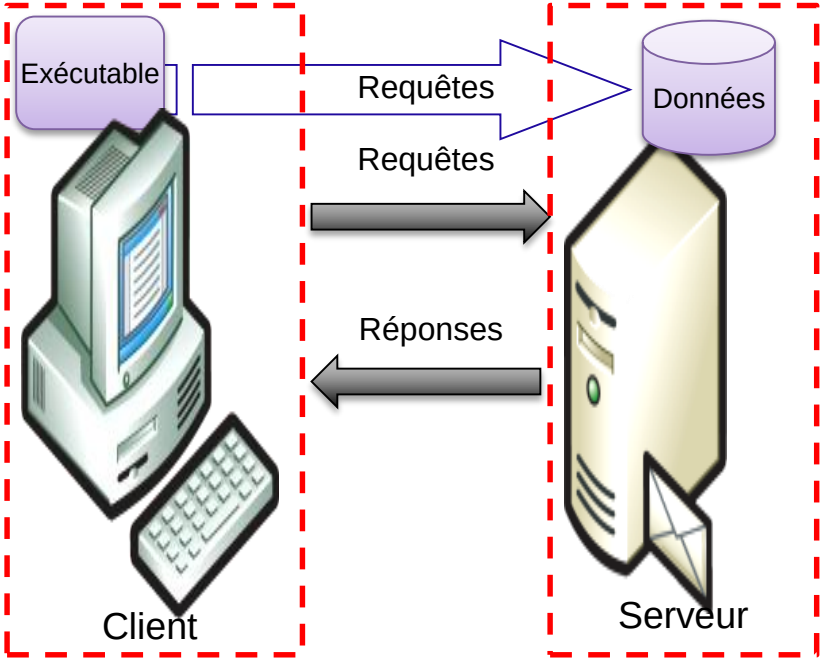


Architecture Multi-tiers

- On peut alors séparer l'application en différentes parties :
  - La couche interface homme machine
  - La couche de traitement
  - La couche de gestion des données.
- Et toute application possède ces trois parties.
- On parle de couches, de niveaux ou de tiers (de l'anglais tiers : étage).
- Jusqu'au années 90, ces trois couches étaient la plupart du temps sur la même machine.
- L'application utilisait les disques de la machine pour lire les données à traiter et stocker les données résultantes du traitement (sauf sur les gros systèmes)

Architecture Multi-tiers

- A partir des années 90, vu l'explosion de la volumétrie des données, on a séparé les applications des données. Ces dernières étaient alors stockées dans des bases de données sur d'autres machines.
- Les applications accédaient aux données à travers les réseaux sur des machines serveurs de données disposant d'un SGBD.
- C'est une **architecture 2-tiers**, ou encore **architecture client-serveur**.



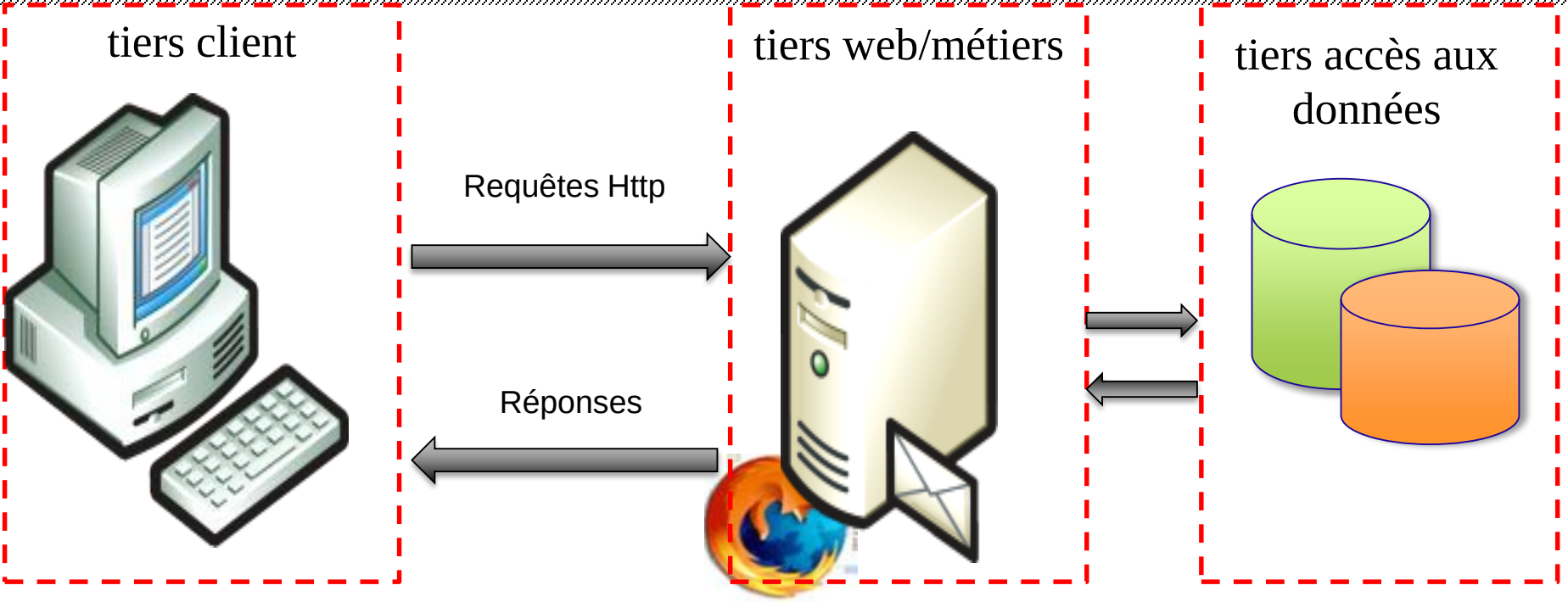
Architecture Client /Serveur classique : 2 tiers

Architecture 2-tiers: les limites

- L'architecture client-serveur de première génération présente les inconvénients suivants :
  - on ne peut pas soulager la charge du client qui supporte déjà **tous les traitements applicatifs** ;
  - le poste client est fortement sollicité, il devient de plus en plus complexe et nécessite des mises à jour régulière, ce qui est contraignant ;
  - le client et le serveur sont assez bruyants, ce qui s'adapte mal à des bandes passantes étroites ;
  - ce qui cantonne ce type d'application à des réseaux locaux ;
  - il est difficile de modifier l'architecture initiale, les applications supportent donc mal les fortes montées en charge ;
  - la relation forte et étroite entre le programme du client et l'organisation de la partie serveur complique les évolutions de cette dernière ;
  - ce type d'architecture est grandement rigidifié par les coûts et la complexité de maintenance.

Architecture 3-tiers

Avec l'apparition des technologies Web, il est possible de séparer la couche présentation de la couche applicative (aussi appelée couche métiers)



Architecture Client /Serveur Web : 3 tiers

Architecture 3-tiers : Couche Présentation

- Elle correspond à la partie de l'application visible et interactive avec les utilisateurs.
- On parle d'IHM. Elle peut être réalisée par une application graphique ou textuelle, en HTML, en WML (on parle alors de clients légers).
- Cette interface peut prendre de multiples facettes sans changer la finalité de l'application.

Définition. Un développeur **front-end** est un professionnel de l'informatique et/ou du web design capable de produire des sites web en utilisant ses connaissances en HTML, CSS et javascript, éventuellement couplées à des compétences dans le domaines d'un ou de plusieurs gestionnaires de contenus (CMS).

Développeur front-end (ou front office) - 1min30 - Agence web 1min30 ...
https://www.1min30.com/dictionnaire-du-web/developpeur-front-end-ou-front-office

Architecture 3-tiers : Couche Présentation

- Dans le cas d'un système de distributeurs de billets, l'automate peut être différent d'une banque à l'autre, mais les fonctionnalités offertes sont similaires et les services identiques (fournir des billets, donner un extrait de compte, etc.).
- Une même fonctionnalité métiers (par exemple, la commande d'un nouveau chéquier) pourra prendre différentes formes de présentation selon qu'elle se déroule sur Internet, sur un distributeur automatique de billets ou sur l'écran d'un chargé de clientèle en agence.

Architecture 3-tiers : Couche Présentation

- La couche présentation relaie les requêtes de l'utilisateur à destination de la couche métiers, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.
- NB: 3-tiers est une application du modèle plus général qu'est le multi-tiers.

Architecture 3-tiers : Couche applicative

- Egalement **couche métiers** ou **business** ou encore **serveur d'applications**:
- Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.
- Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

Architecture 3-tiers : Couche applicative

- La couche métiers offre des services applicatifs et métiers à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche inférieure.
- En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

Backend. ... En informatique, un **back-end** (parfois aussi appelé un arrière-plan) est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-**end** (aussi appelé un frontal) qui lui est la partie visible de l'iceberg.

Backend — Wikipédia
https://fr.wikipedia.org/wiki/Backend

Architecture 3-tiers : Couche données

- **Données gérées par un autre système:**
  - Les données peuvent aussi être gérées de manière externe.
  - Elles ne sont pas stockées par le système considéré, il s'appuie sur la capacité d'un autre système à fournir ces informations.
  - Par exemple, une application de pilotage de l'entreprise peut ne pas sauvegarder des données comptables de haut niveau dont elle a besoin, mais les demander à une application de comptabilité.
- Celle-ci est indépendante et pré-existante, et on ne se préoccupe pas de savoir comment elle les obtient ou si elle les sauvegarde, on utilise simplement sa capacité à fournir des données à jour.

Architecture 3-tiers : Couche données

- **Données propres au système** :
  - Ces données sont pérennes, car destinées à durer dans le temps, de manière plus ou moins longue, voire définitive.
  - Les données peuvent être stockées indifféremment dans de simples fichiers texte, fichiers XML, ou encore dans un SGBD
  - Quel que soit le support de stockage choisi, l'accès aux données doit être le même. Cette abstraction améliore la maintenance du système.
  - Le DATA ACCESS OBJET (DAO ou design pattern) consiste à représenter les données du système sous la forme d'un modèle objet. Par exemple un objet pourrait représenter un contact ou un rendez-vous.
  - La représentation du modèle de données objet en base de données (appelée persistance) peut être effectuée à l'aide d'outils tel que HIBERNATE.

Architecture 3-tiers : Couche données

- **Données propres au système** :
  - Ces données sont pérennes, car destinées à durer dans le temps, de manière plus ou moins longue, voire définitive.
  - Les données peuvent être stockées indifféremment dans de simples fichiers texte, fichiers XML, ou encore dans un SGBD
  - Quel que soit le support de stockage choisi, l'accès aux données doit être le même. Cette abstraction améliore la maintenance du système.
  - Le DATA ACCESS OBJET (DAO ou design pattern) consiste à représenter les données du système sous la forme d'un modèle objet. Par exemple un objet pourrait représenter un contact ou un rendez-vous.
  - La représentation du modèle de données objet en base de données (appelée persistance) peut être effectuée à l'aide d'outils tel que HIBERNATE.

Architecture Multi-tiers

- L'architecture multi-tiers est aussi appelée architecture distribuée ou architecture n-tiers.
  - L'architecture multi-tiers qualifie la distribution d'applications entre de multiples services et non la multiplication des niveaux de service : les trois niveaux d'abstraction d'une application sont toujours pris en compte.
  - Cette distribution est facilitée par l'utilisation de **composants métiers**, spécialisés et indépendants, introduits par les concepts orientés objets.
  - Elle permet de tirer pleinement partie de la notion de **composants métiers** réutilisables et modulables.
  - Un **composant** est une «boîte noire», créé par un développeur, qu'il va réutiliser et que d'autres développeurs vont utiliser. Le développeur utilisateur connaît seulement les points d'entrée et le type des informations retournées. Ex: **Les composants JavaEE sont les EJB.**

Architecture Multi-tiers

Ces composants rendent un service, si possible, générique et clairement identifié. Ils sont capables de communiquer entre eux et peuvent donc coopérer en étant implantés sur des machines distinctes et hétérogènes.

- Tiers Applicatif :
  - CGI
  - ASP
  - Java Servlets
  - JSP
  - PHP, Python
  - JavaScripts
  - …

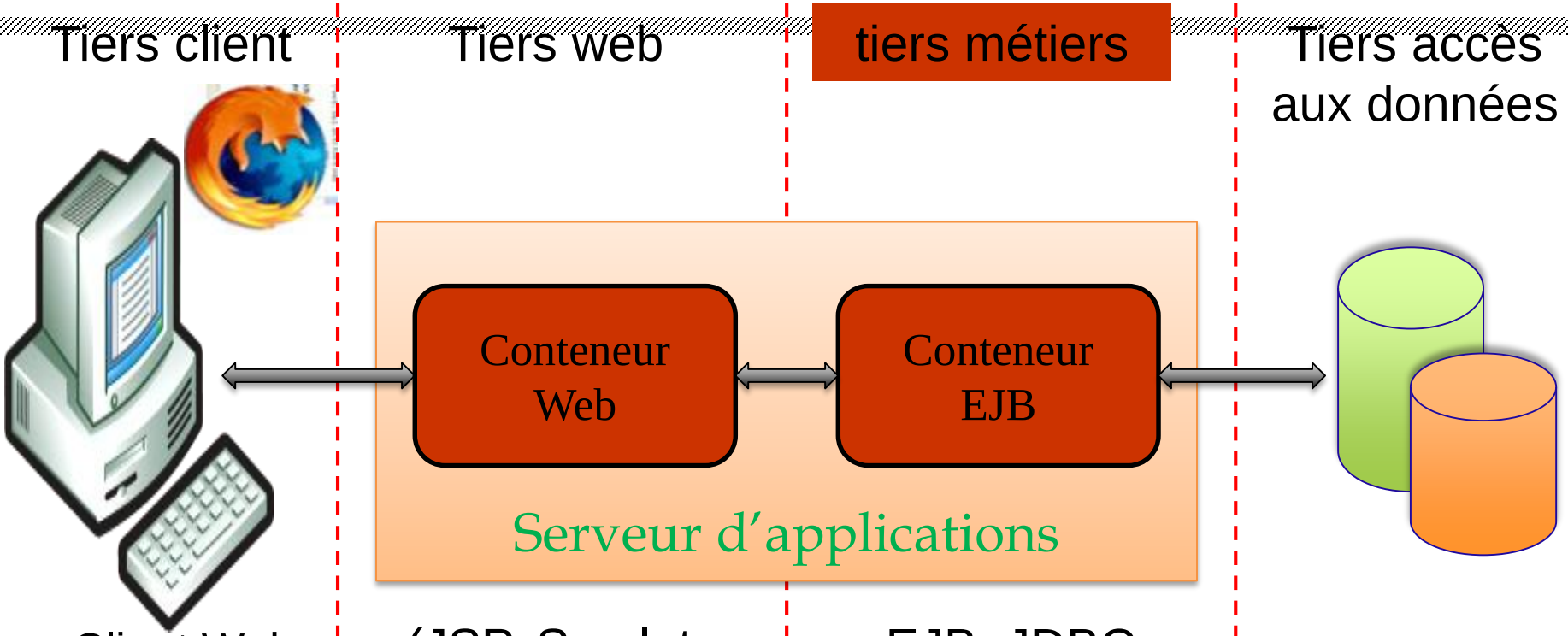
- Tiers client:
  - Un navigateur Web
  - Un PDA ou SmartPhone

- Tiers données:
  - Base de données via SQL, JDBC, .NET, JavaEE
  - ERP (Enterprise Resource Planning),
  - EAI (Enterprise Application Integration)

Architecture Multi-tiers: Point de vue matériel

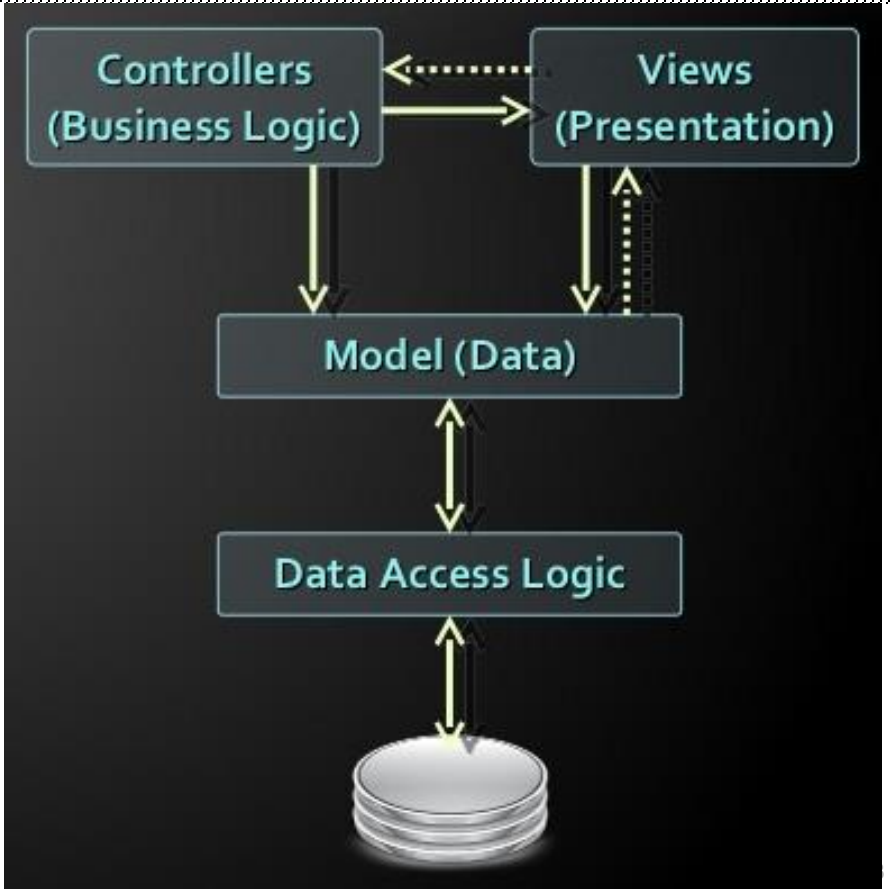
- L'architecture présentée en purement logicielle.
- En conséquence, chaque étage peut être implémenté sur n'importe quelle machine.
- Ainsi, Les trois étages peuvent être réunies sur une seule ou sur plusieurs machines.
- L'intérêt de répartir les différents étages sur plusieurs machines permet un accroissement de la sécurité.

Architecture Multi-tiers: Point de vue JavaEE



Architecture Multi-tiers et Modèle MVC

- Peuvent être utilisées ensemble pour la séparation des couches métiers, données et présentation



Finalement… Développer un modèle architectural

- Commencer par faire une esquisse de l'architecture
  - En se basant sur les principaux requis des cas d'utilisation ; décomposition en sous-systèmes
  - Déterminer les principaux composants requis
  - Sélectionner un style architectural
- Raffiner l'architecture
  - Identifier les principales interactions entre les composants et les interfaces requises
  - Décider comment chaque donnée et chaque fonctionnalité sera distribuée parmi les différents composants
  - Déterminer si on peut réutiliser un framework existant (réutilisation) ou si on peut en construire un (réutilisabilité).

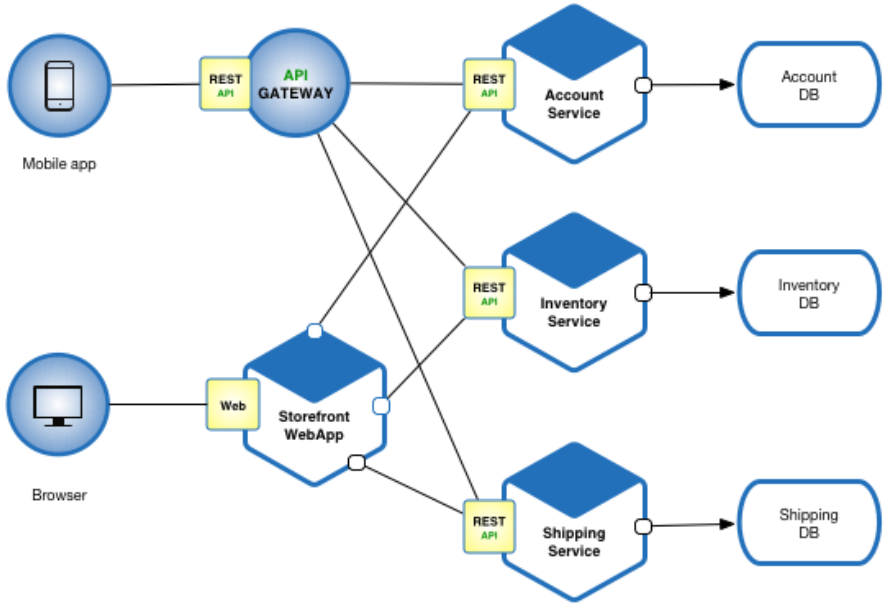
Pour terminer la tendance: Architecture Microservices

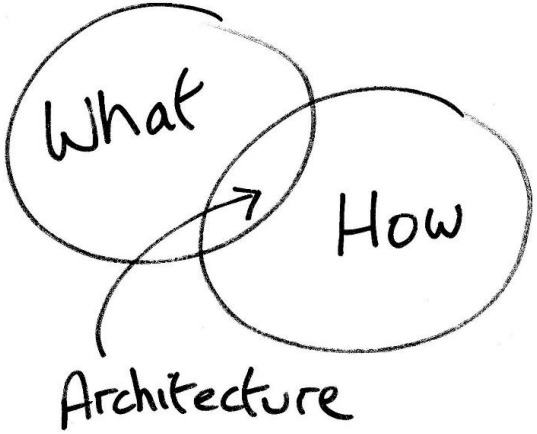
- https://fr.wikipedia.org/wiki/Microservices
  - En informatique, les microservices sont un style d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche. Les processus indépendants communiquent les uns avec les autres en utilisant des API langage-agnostiques.
  - Des API REST sont souvent employées pour relier chaque microservice aux autres. Un avantage avancé est que lors d'un besoin critique en une ressource, seul le microservice lié sera augmenté, contrairement à la totalité de l'application dans une architecture classique, par exemple une architecture trois tiers. Cependant, le coût de mise en place, en raison des compétences requises, est parfois plus élevé.

Pour terminer la tendance: Architecture Microservices

Imaginons que vous construisiez une application de commerce électronique qui prend les commandes des clients, vérifie l'inventaire et le crédit disponible, et les expédie. L'application se compose de plusieurs composants, y compris le StoreFrontUI, qui implémente l'interface utilisateur, ainsi que certains services de backend pour vérifier le crédit, maintenir l'inventaire et les commandes d'expédition. L'application consiste en un ensemble de services.

Amazon, Bluemix, Cloud Foundry, Google, *The Guardian* Hailo Taxi,HP Helion Development Platform, Jelastic Devslack, Microsoft Azure, Netflix[5] (Netflix reçoit 2 milliards de requêtes chaque jour, conduisant à environ 20 milliards d'appels à des API internes), Riot Games SoundCloud, Uber et d'autres sites Web à grande échelle et les applications ont tous évolué de l'architecture monolithique à microservices.



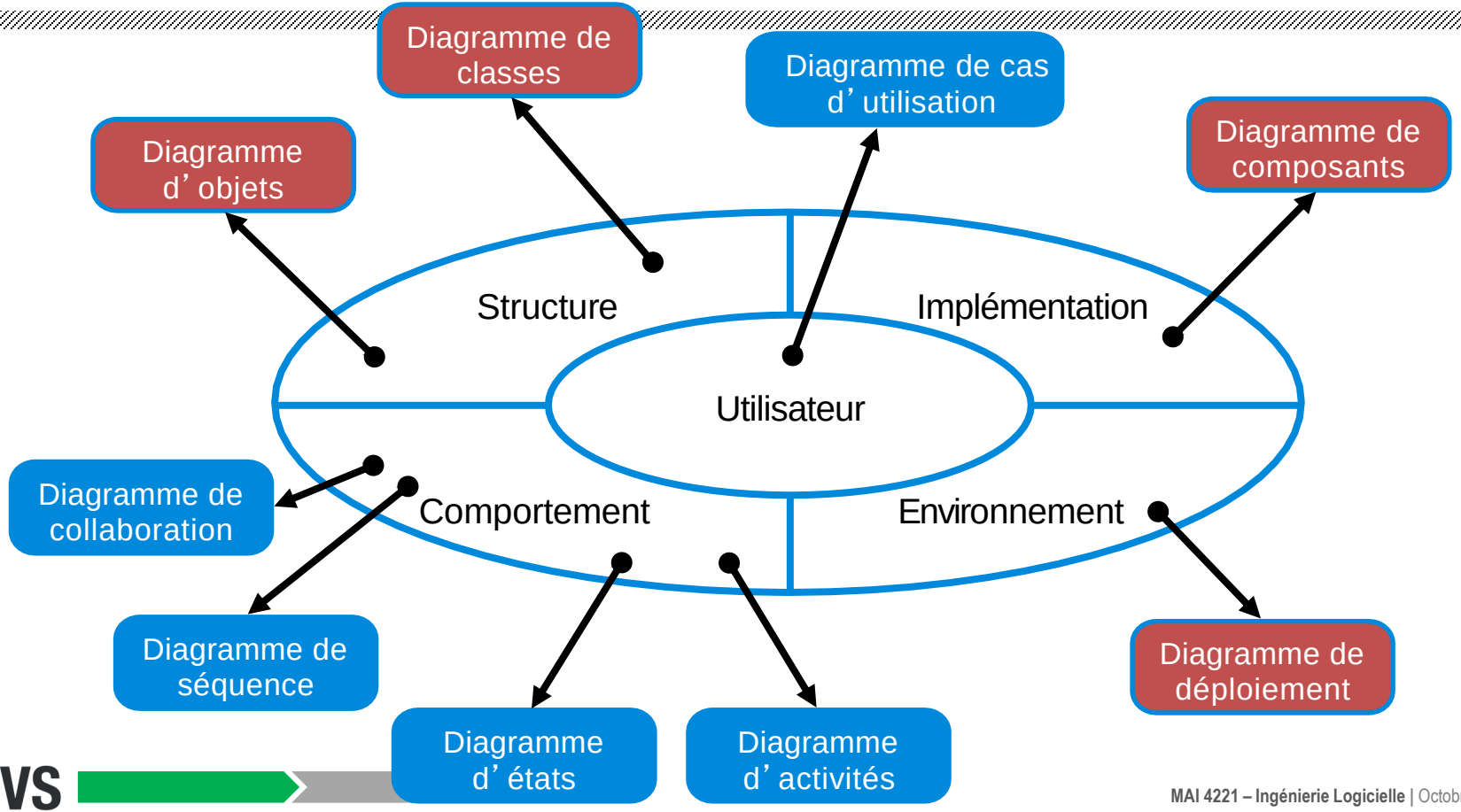


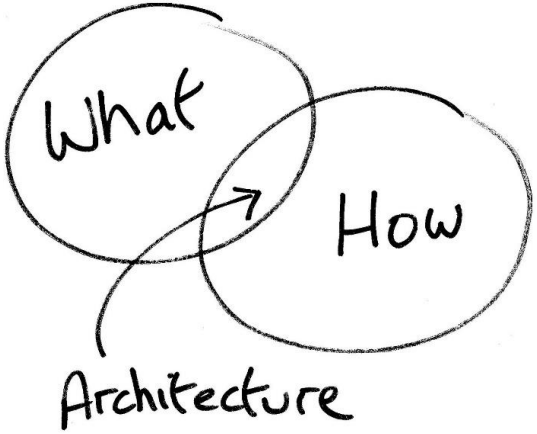
Conception et Architectures Logicielles:
Modèle architectural avec UML

Développer un modèle architectural avec UML

- Décrire l'architecture avec UML
  - Tous les diagrammes UML peuvent être utiles pour décrire les différents aspects du modèle architectural
  - Trois des diagrammes UML sont particulièrement utile pour décrire une architecture logicielle
    - Diagramme de packages
    - Diagramme de composants
    - Diagramme de déploiement
  - Diagramme de classes : décrit les "données" du logiciel (structure, traitements, relations, contraintes, rôles)
  - Diagramme d'objets : diagramme de classes instanciées utilisé pour illustrer un exemple particulier
  - Diagramme de composants : montre l'architecture physique du logiciel et l'affectation des objets aux différents composants de cette architecture
  - Diagramme de déploiement : montre la configuration des différents composants à l'exécution et leur distribution

Modélisation de l'architecture avec UML





Conception et Architectures Logicielles:
Architecte Logiciel, Evaluation d'une architecture logicielle

Rôle d'un Architecte Logiciel

- Le rôle de l'**Architecte logiciel** est:
  - de définir une architecture logicielle qui satisfasse les contraintes du système
  - de la communiquer et la promouvoir
  - de défendre son intégrité conceptuelle
  - de la critiquer
  - de la raffiner
- Fiche métier:

http://referentiels-metiers.opiiec.fr/fiche-metier/97-architecte-logiciel

Profil d'un Architecte Logiciel

- Crédibilité
- Pensée intégrative
- Créativité
- Empathie
- Autodidacte
- Leader/Mentor
- Communicateur(Négociateur)
- Excellent analyste
- Designer hors pair
- Bon programmeur

7 Périls d'un Architecte Logiciel

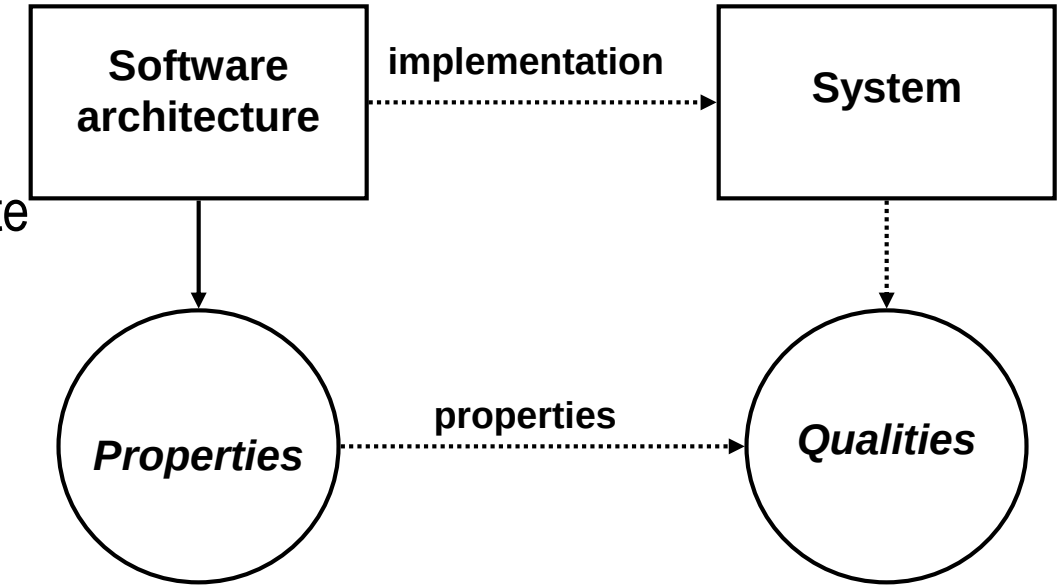
- Un projet où la direction de l'organisation ne croie pas à l'architecture logicielle
- Un projet dont les usagers ne veulent pas
- Embarquer sur un projet sans avoir de crédibilité auprès d'une équipe rebelle
- Un projet avec un niveau d'incertitude élevé soumis à un développement en cascade
- Un projet dont des choix technologiques clés imposés sont inappropriés
- Un projet où les analystes d'affaires produisent le schéma de BD
- Prendre la relève d'un projet en détresse

Évaluation d'architecture

- Évaluation / analyse d'architecture
- Évaluer si l'architecture répond à certains objectifs de qualité, tels que ceux de maintenabilité, modificabilité, fiabilité, performance
- NB: l'architecture est évaluée, alors que nous espérons que les résultats seront valables pour un système à construire

Conditions préalables pour une évaluation réussie

- Objectifs clairs et exigences pour l'architecture
- Portée contrôlée
- Rentabilité
- Disponibilité du personnel clé
- Équipe d'évaluation compétente
- Attentes gérées



Conception et Architectures Logicielles:
A Retenir...

A retenir…

- Une architecture logicielle est une description de la manière dont un système logiciel est organisé. Les propriétés d'un système, telles que les performances, la sécurité et la disponibilité, sont influencées par l'architecture utilisée.
- Les décisions de conception architecturale comprennent les décisions relatives au type d'application, à la distribution du système, aux styles architecturaux à utiliser et aux moyens de documenter et d'évaluer l'architecture.

A retenir…

- Les architectures peuvent être documentées à partir de plusieurs perspectives ou vues. Les vues possibles incluent une vue conceptuelle, une vue logique, une vue processus, une vue développement et une vue physique.
- Les modèles d'architecture permettent de réutiliser les connaissances sur les architectures de système génériques. Ils décrivent l'architecture, expliquent quand elle peut être utilisée et discutent de ses avantages et inconvénients.
- Les modèles d'architecture couramment utilisés incluent Modèle-Vue-Contrôleur, Architecture en couches, Référentiel, Client-serveur et Pipe and Filter.

A retenir…

- Les modèles génériques d'architectures de systèmes d'application nous aident à comprendre le fonctionnement des applications, à comparer des applications du même type, à valider les conceptions de système d'application et à évaluer les composants à grande échelle pour les réutiliser.


thank you
danke
謝謝
рахмат
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
dziękuję
bedankt
mauruuru
hvala
go raibh maith agat
sukriya
kop khun krap
terima kasih
grazie
arigatô
dakujem
merci
спасибо
Баярлалаа
faafetai lava
kiitos
dankie
dhanyavad
nandri
nanni
köszönöm
enkosi
bayarlalaa
grâçie
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
хвала
asante
manana
tenki
obrigada
murakoze
chokrane
mochchakkeram
djiere dieuf
tau
дякую
mamnun
akun
dankon
aciù
chnorakaloutioun
gralias ago
gràcies
sulpáy
taiku
takk
trugarez
sobodi
dekuji
sagolun
mèsi
didi madloba
kam sah hamnida
najis tuke
rahmat
ありがとう
tanemirt
rahmet
dhanyavadagalu
diolch
shukriya
мерси
merce
xièxie
감사합니다
তোমাকে ধন্যবাদ
ευχαριστώ
vinaka
спасибі
blagodaram
mersi
kia ora
barka
welalin
tack
spas