Partie 4 : Représentations de l'architecture logicielle

- Les principales normes et représentations :
  - UML,
  - Kruchten 4+1 Architectural View,
  - Archimate,
  - BPMN
- Les bonnes pratiques de représentation
- Quelques outils

Partie 4 : lien vers la présentation

https://drive.google.com/file/d/1xhbFUR2k7TPsw7ecQrUWNB-V7QgVbnqe/view?usp=sharing

UML et MDA

- UML : Unified Modeling Language

- Premières versions (0.9 et 0.9.1) en 1996, version 2.0 en 2005
  Version actuelle 2.5.1 en 2017

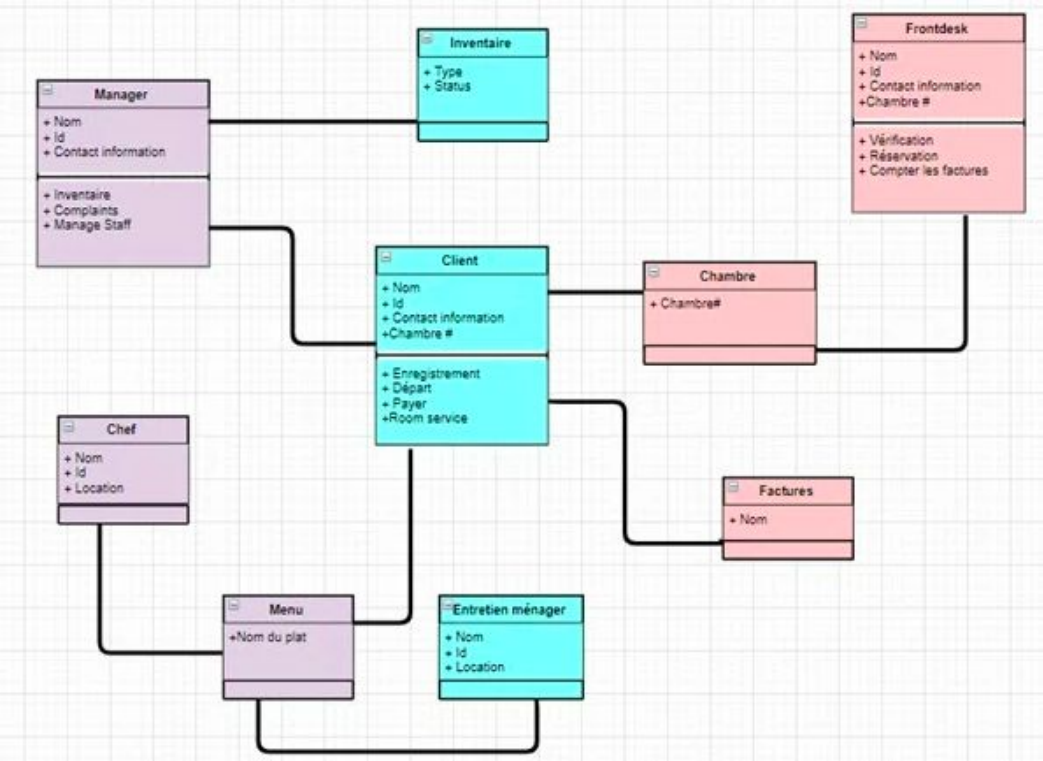
- MDA : Model Driven Architecture
  Principe : élaboration de différents modèles, en partant d'un modèle métier indépendant de l'informatisation (computation independent model, CIM), la transformation de celui-ci en modèle indépendant de la plate-forme (platform independent model, PIM) et enfin la transformation de ce dernier en modèle spécifique à la plate-forme cible (platform specific model, PSM) pour l'implémentation concrète du système.

- Les spécifications UML et MDA sont définies par l'OMG (Object Management Group)

UML : 14 diagrammes

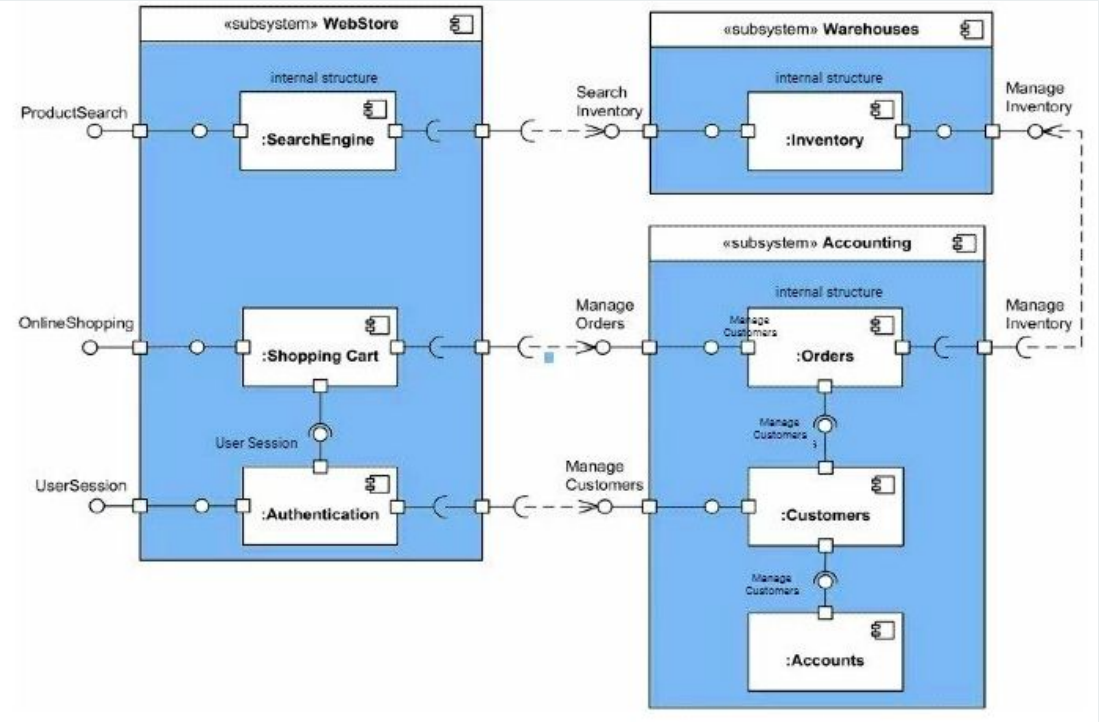
Diagrammes de structures	Diagrammes comportementaux
Diagramme de classes	Diagramme de cas d'utilisation
Diagramme de composants	Diagramme d'activités
Diagramme de structure composite	Diagramme d'état
Diagramme de déploiement	Diagramme de séquence
Diagramme d'objets	Diagramme de communication (Diagramme de collaboration)
Diagramme de paquetage	Diagramme de présentation des interactions
Diagramme de profil	Diagramme de temps

UML : Diagramme de classes



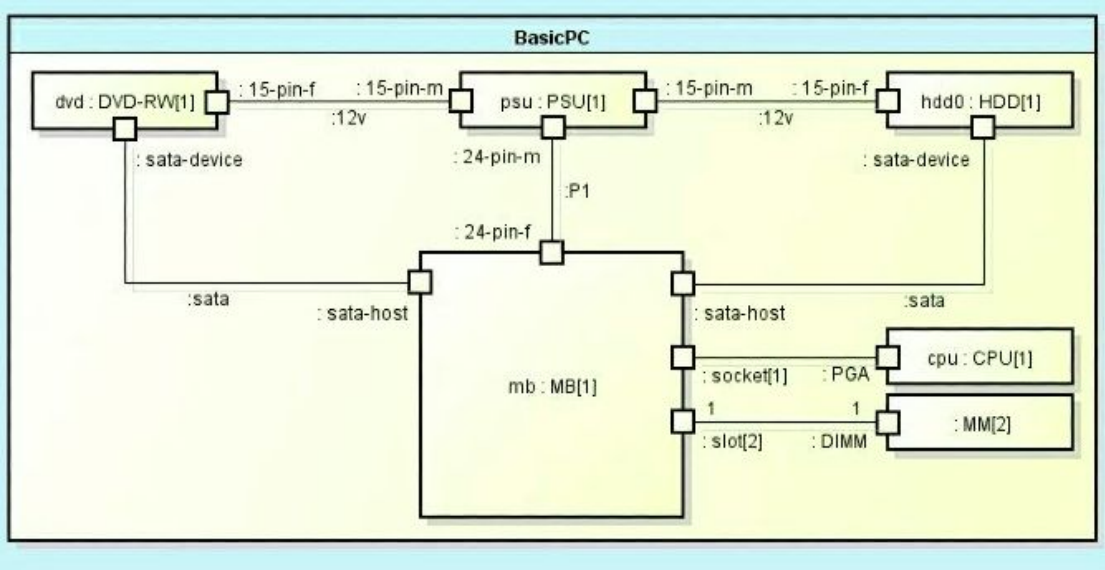
- fondement principal de toute solution orientée objet.
- utilisé pour la modélisation conceptuelle de la structure réelle du système et la modélisation détaillée.
- il est également possible avec les diagrammes de classes de faire de la modélisation des données.
- Le diagramme permet de représenter les classes d'un système, attributs et opérations, et relations entre chaque classe.

UML : Diagramme de composants



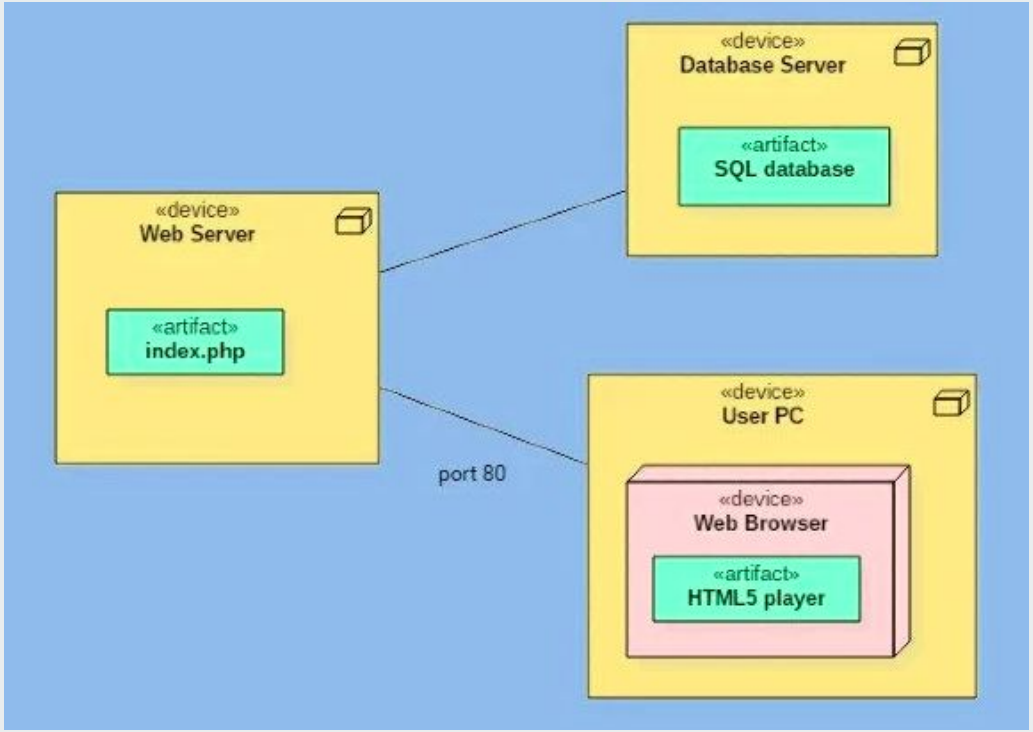
- Représente la relation structurelle entre les éléments d'un système logiciel, le plus souvent utilisé avec des systèmes complexes disposant de multiples composants.
- Les composants communiquent à l'aide d'interfaces.

UML : Diagramme de structure composite



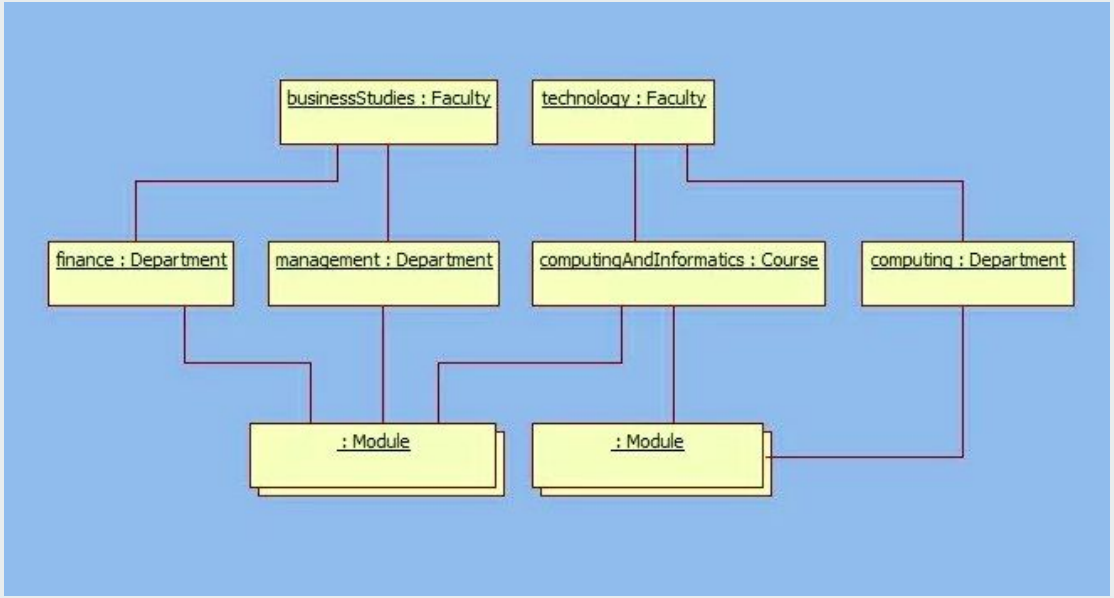
- utilisé pour présenter la structure interne d'une classe.

UML : Diagramme de déploiement



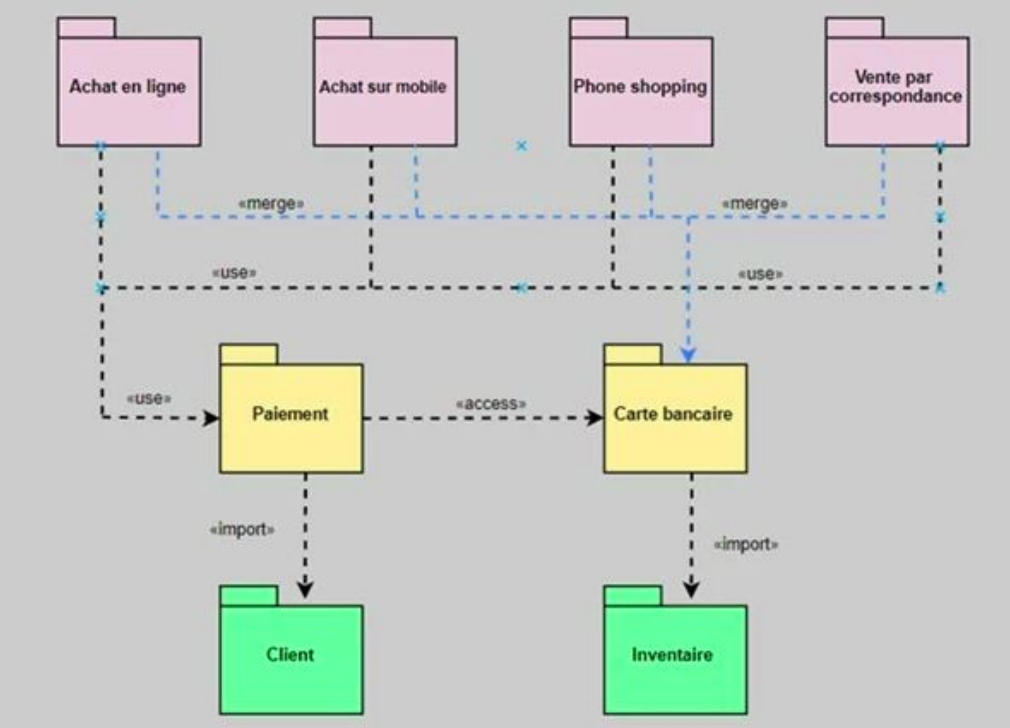
- Illustre les infrastructures physique et logicielle d'un système.
- Ils sont utiles lorsqu'une solution logicielle est déployée sur de nombreuses machines avec des configurations uniques.

UML : Diagramme d'objets



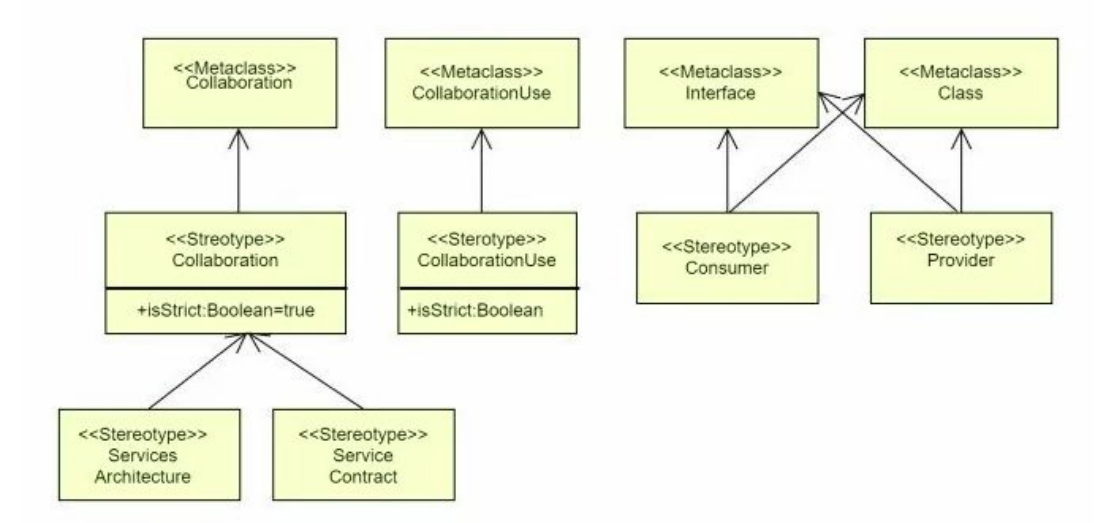
- Montre les relations entre des objets à travers des exemples tirés du monde réel et permet de voir l'apparence d'un système à n'importe quel instant donné.
- Les données sont disponibles à l'intérieur des objets, elles peuvent donc être utilisées pour clarifier les relations entre des objets.

UML : Diagramme de paquetage (package)



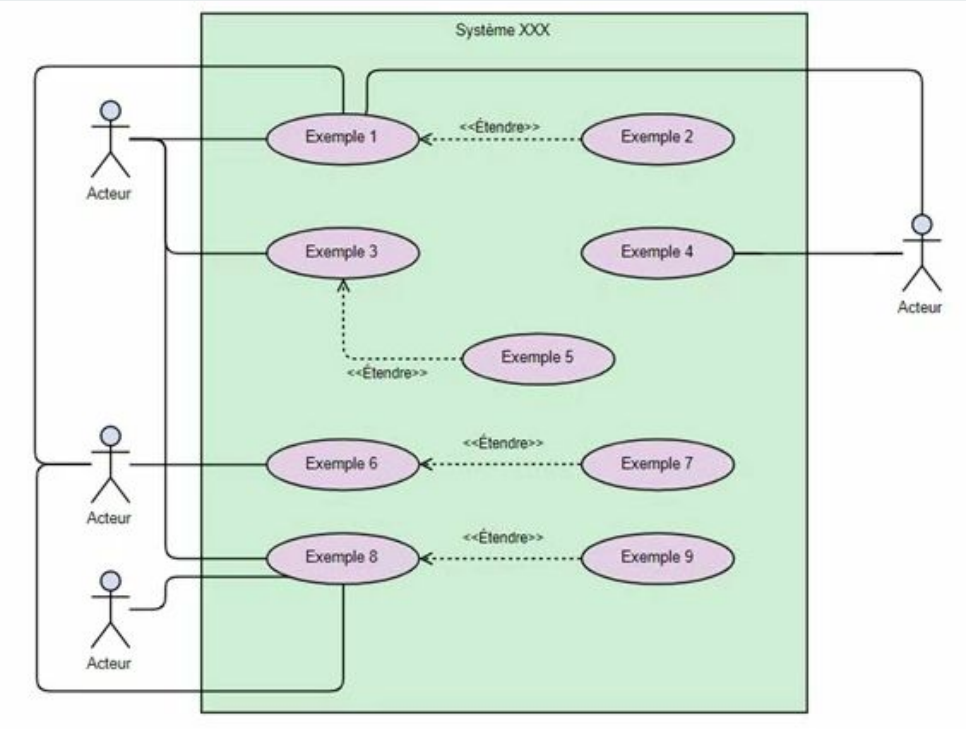
- Affiche la disposition des éléments du modèle dans un projet de moyenne à grande échelle.
- Ce diagramme est principalement utilisé dans les systèmes à grande échelle pour prévoir les dépendances entre les éléments clés du système.

UML : Diagramme de profil



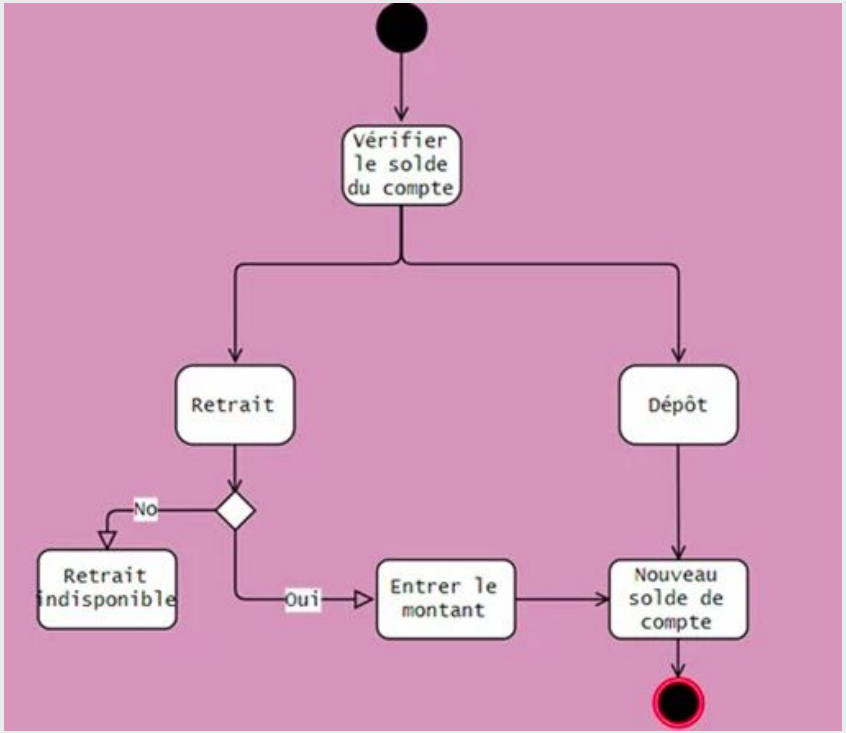
- Les profils servent à étendre l'UML, qui est basé sur davantage de stéréotypes ajoutés et de valeurs balisées qui sont appliquées aux éléments, aux composants et aux connecteurs.

UML : Diagramme de cas d'utilisation



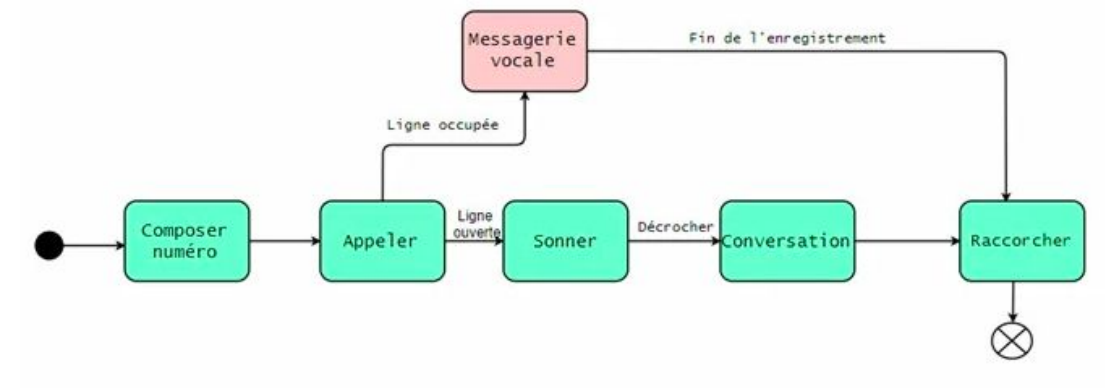
- représente une fonctionnalité spécifique dans un système et est créé pour illustrer comment différentes fonctionnalités sont interconnectées et montrer leurs contrôleurs (ou acteurs) internes et externes.

UML : Diagramme d'activité



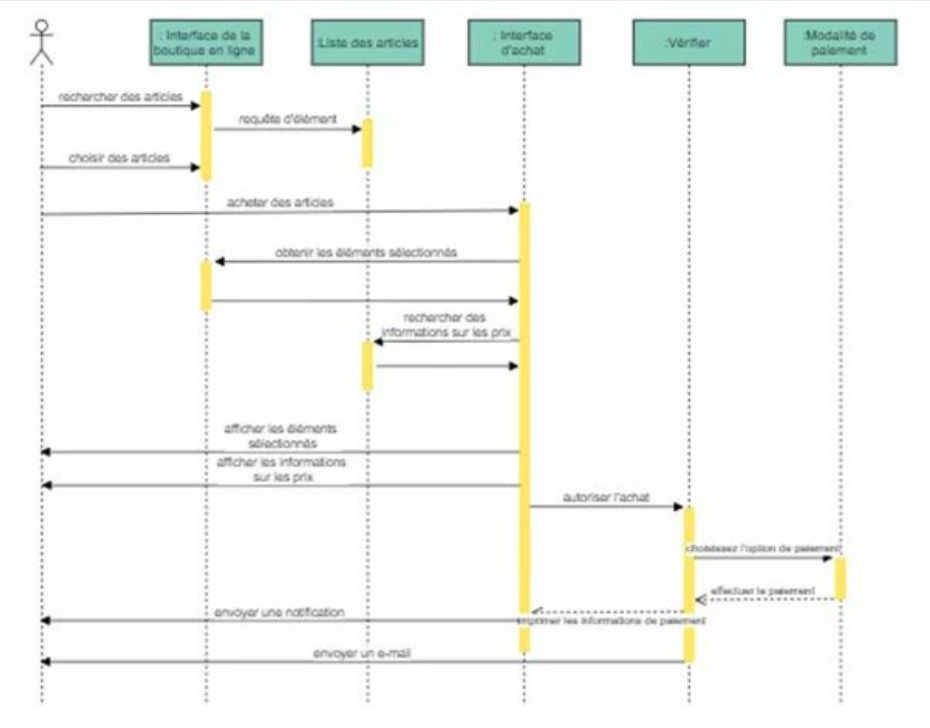
- représente l'activité de chacune des composantes d'un système
- Le diagramme d'activité a généralement un état initial et un état final.

UML : Diagramme d'état (états - transitions)



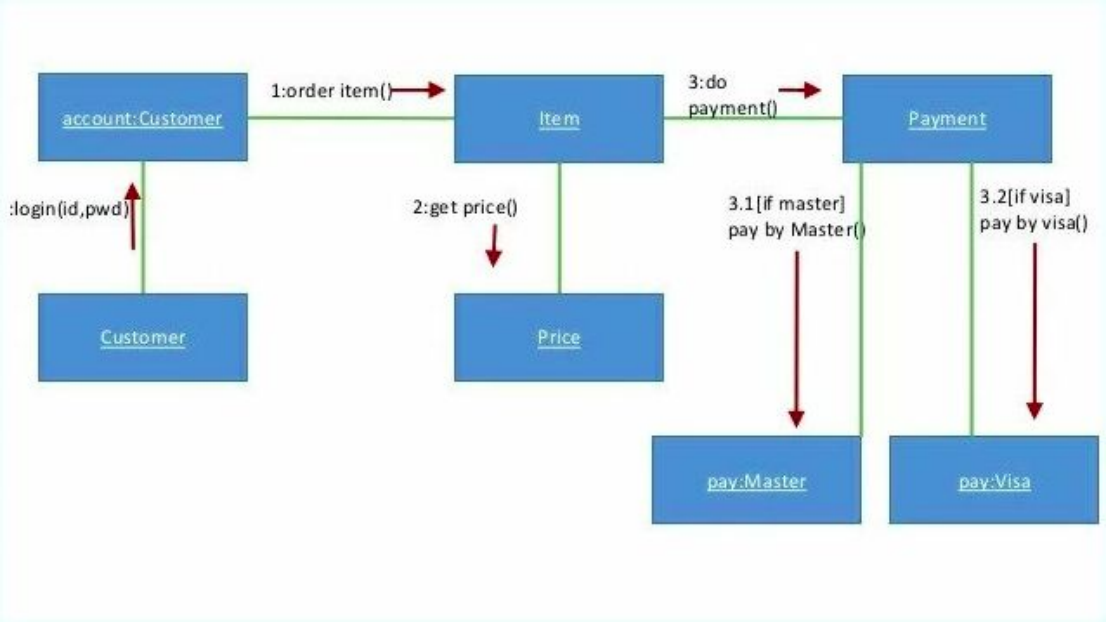
- montre le comportement d'un objet spécifique, indiquant la séquence d'événements que l'objet subit tout au long de sa durée de vie.

UML : Diagramme de séquences



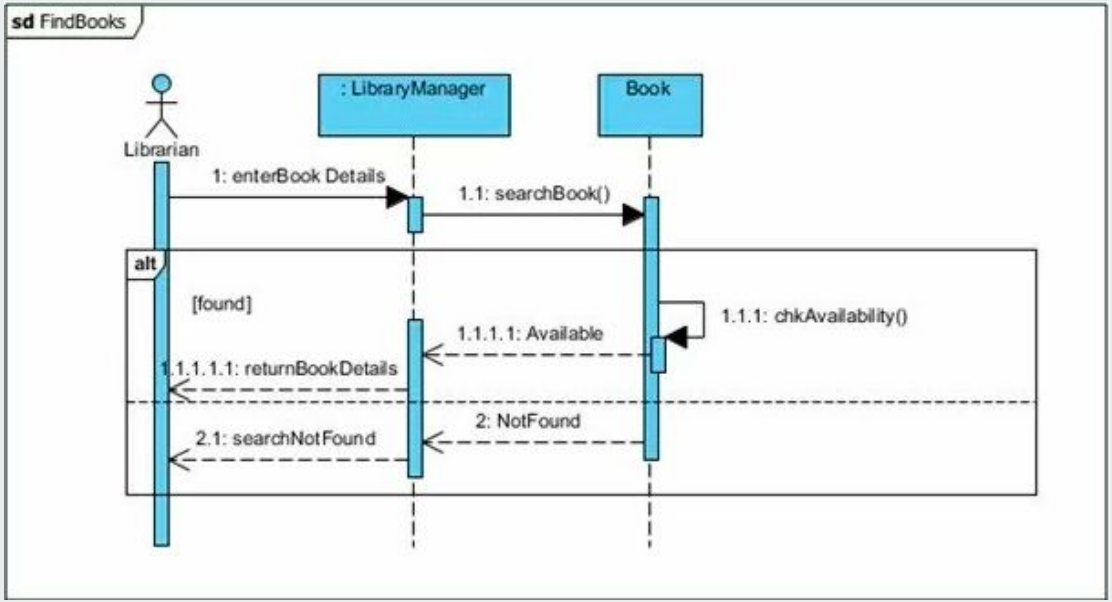
- montre comment les objets interagissent les uns avec les autres et dans quel ordre.
- représente les interactions d'un scénario particulier.

UML : Diagramme de communication (collaboration)



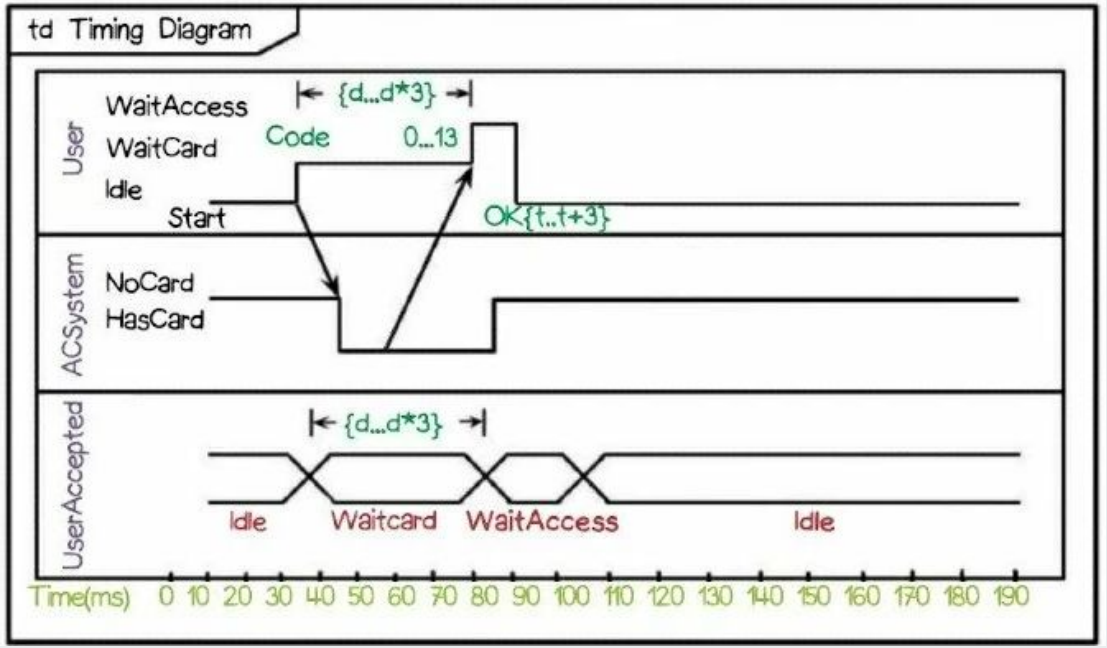
- Semblable à un diagramme de séquences, mais l'accent est mis sur les messages transmis entre les objets.

UML : Diagramme de présentation des interactions



- détaille l'image globale du flux de contrôle de l'interaction spécifique.
- variante du diagramme d'activités

UML : Diagramme de temps

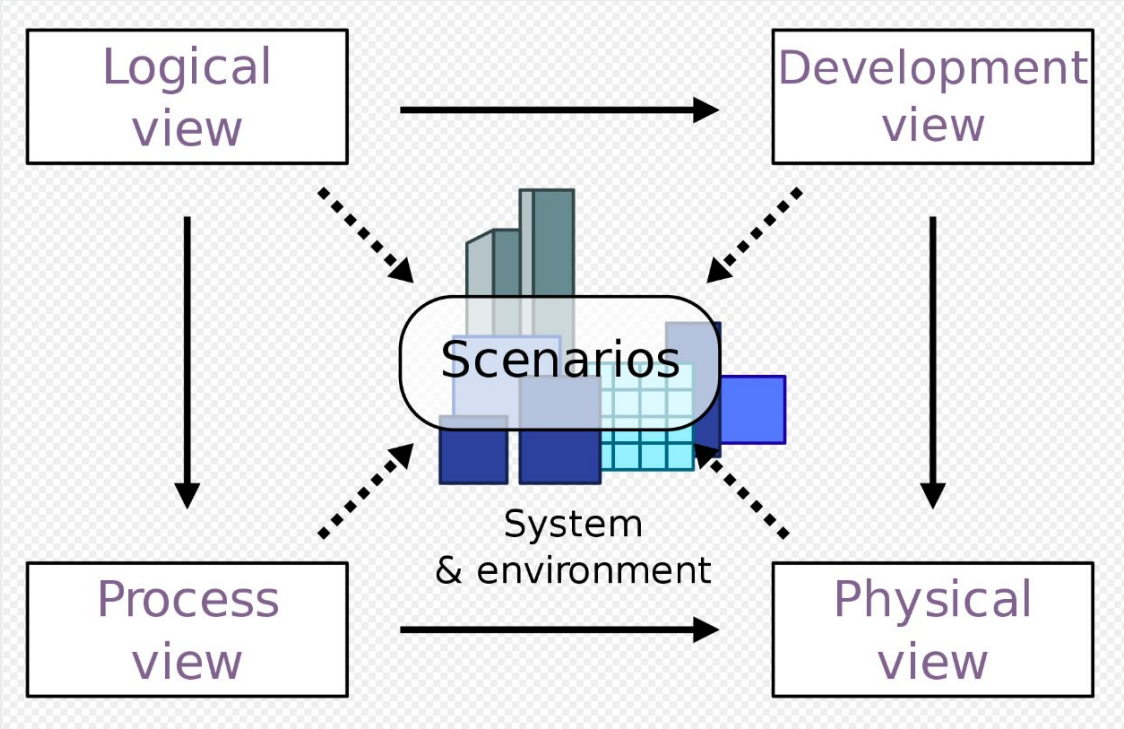


- représente le changement d' état ou de valeur d'un ou plusieurs objets sur un certain temps.
- Ce type de diagramme se compose principalement d'une ligne de vie, d'une chronologie d'état, d'une contrainte de durée, d'une contrainte de temps et d'une occurrence de destruction.

UML : Conclusion

- quelques diagrammes utiles pour schématiser l'architecture logicielle
  - diagramme de déploiement
  - diagramme de séquences
  - diagramme de composants

Kruchten 4+1 Model

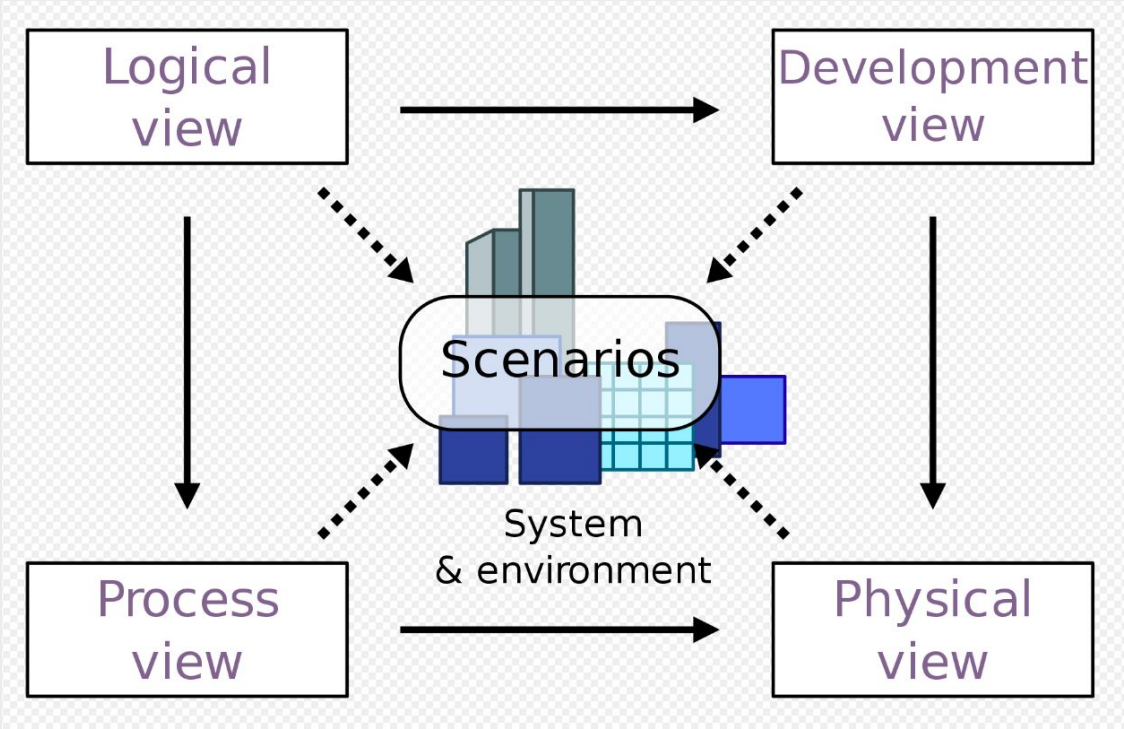


Les vues sont utilisées pour décrire le système du point de vue des différentes parties prenantes, telles que :

- les utilisateurs finaux,
- les développeurs,
- les ingénieurs système,
- les chefs de projet

Les scénarios constituent la 5ème vue.

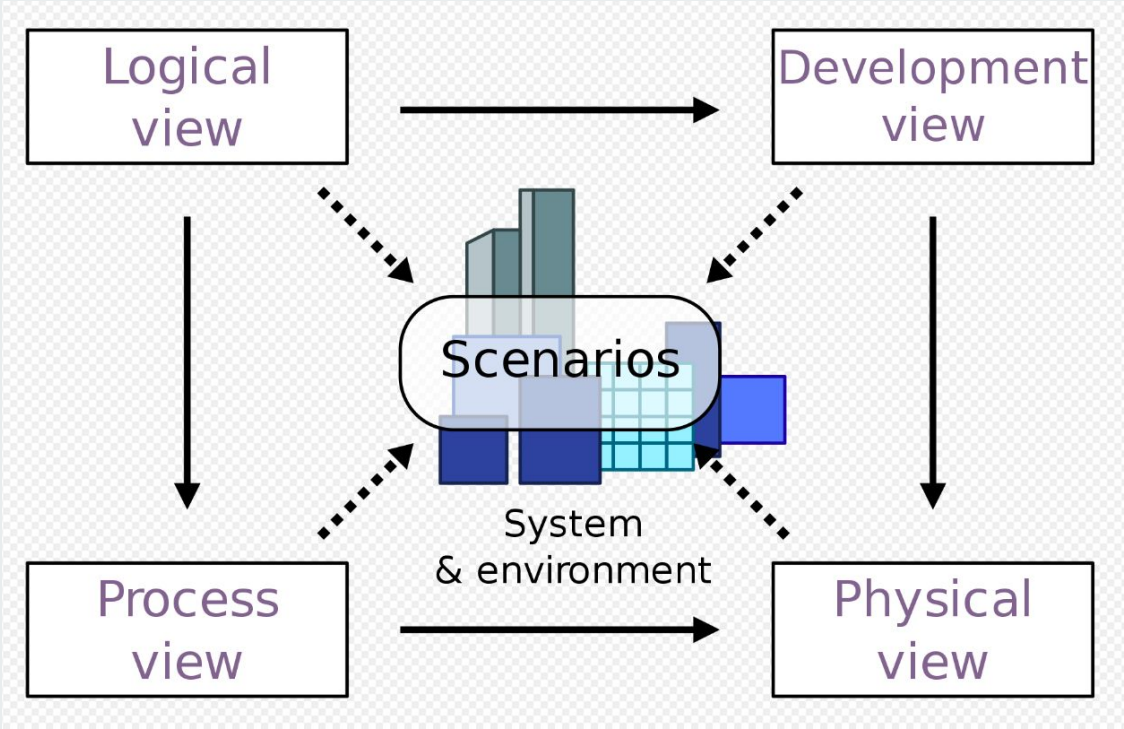
Kruchten 4+1 Model



Logical view :

concerne les fonctionnalités du système du point de vue des utilisateurs finaux.
Les diagrammes de classes et les diagrammes d'états sont des exemples de diagrammes UML utilisés pour cette vue.

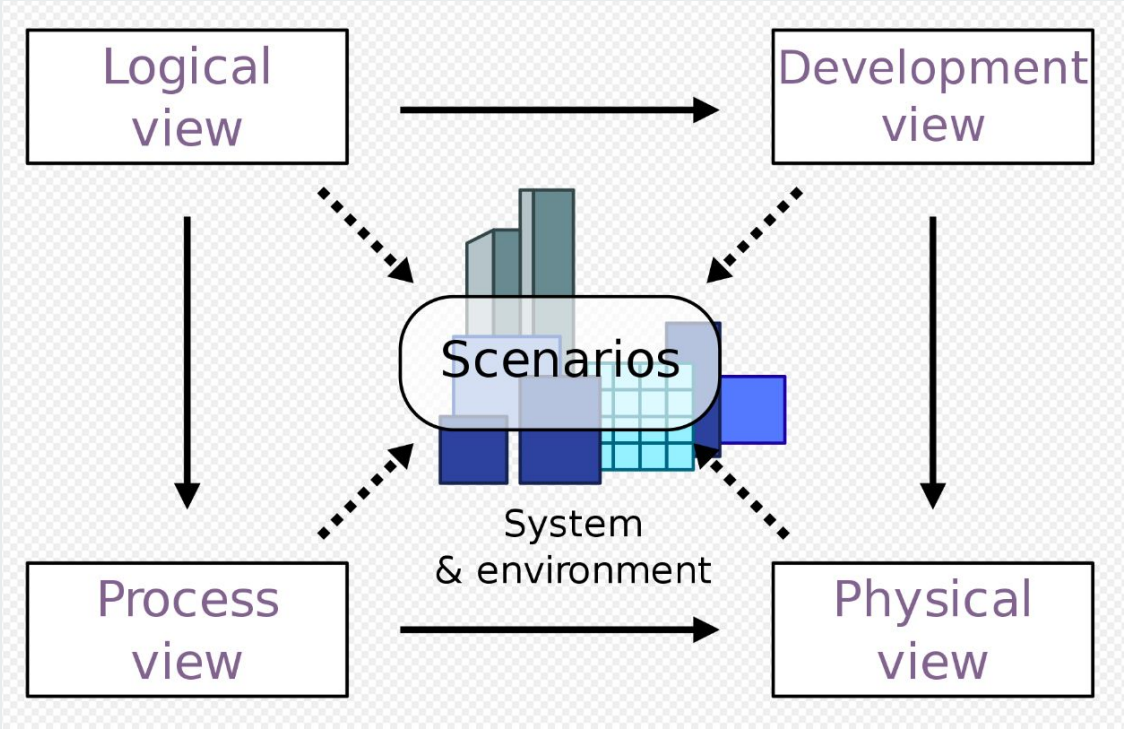
Kruchten 4+1 Model



Process view :

Représente le comportement du système en cours d'exécution et traite des éléments dynamiques du système. Elle explique les processus du système et la façon dont ils communiquent. Les problématique de performances et la scalabilité sont abordées dans la vue processus.
Le diagramme de séquence, le diagramme de communication et le diagramme d'activité peuvent être utilisés pour cette vue.

Kruchten 4+1 Model



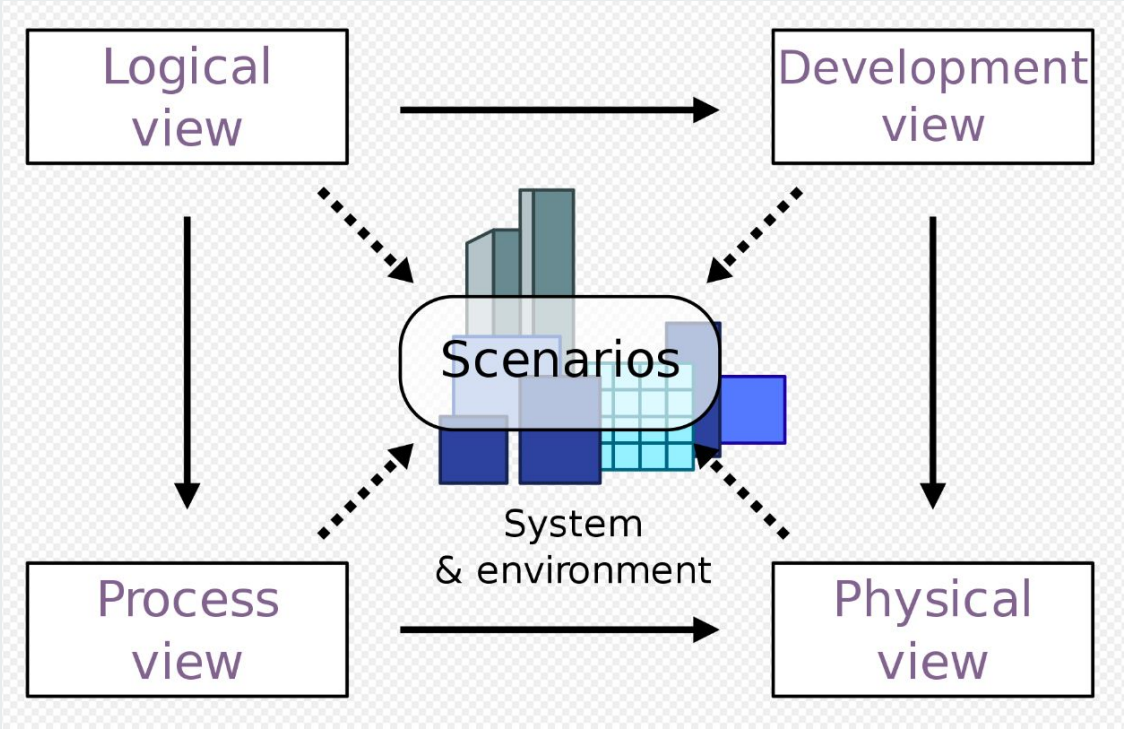
Development view :

Représente le système du point de vue du développeur et s'intéresse à l'administration du logiciel.

Autre nom : Implementation view.

Les diagrammes UML Composants et Package peuvent être utilisés pour représenter cette vue.

Kruchten 4+1 Model



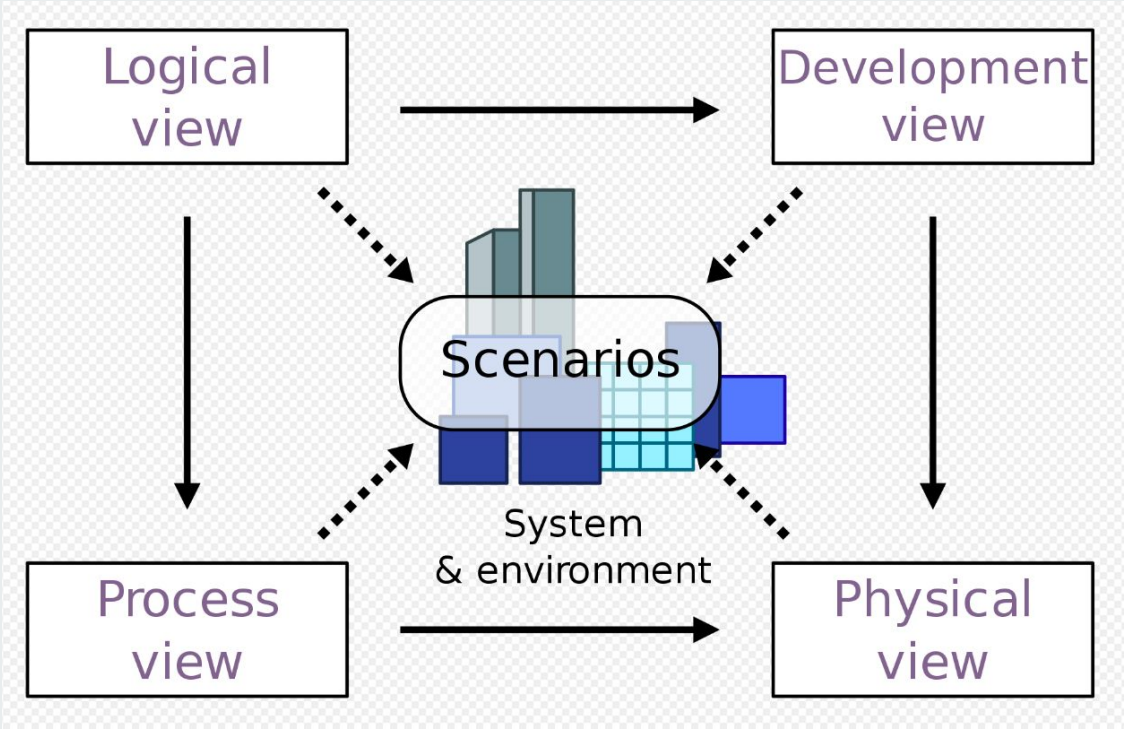
Physical view :

Représente le système du point de vue d'un ingénieur système et s'intéresse à la topologie des composants logiciels ainsi qu'aux connexions physiques entre ces composants.

Autre nom : Deployment view.

Le diagramme de déploiement est l'un des diagrammes UML utilisés pour cette vue.

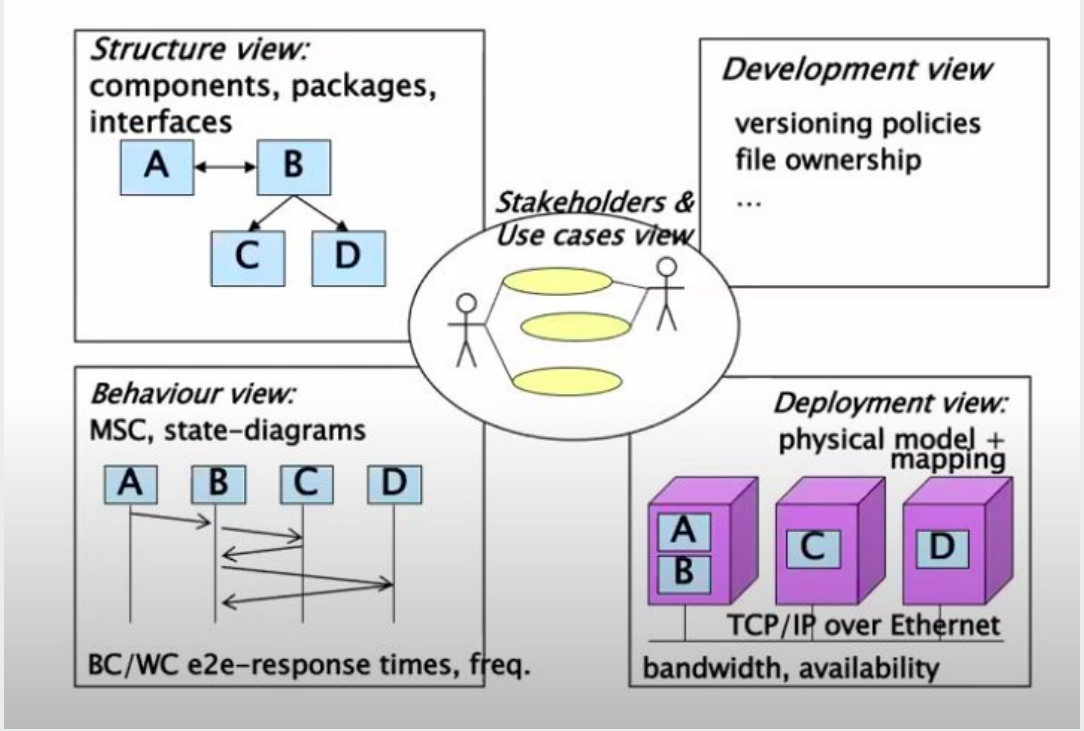
Kruchten 4+1 Model



Scénarios :

Un petit nombre de cas d'utilisation, ou scénarios, qui deviennent la cinquième vue, sont utilisés pour illustrer la description de l'architecture. Les séquences d'interactions entre les objets et les processus sont décrites dans les scénarios.

Kruchten 4+1 Model : Exemple



ArchiMate

- ArchiMate est un langage de modélisation pour décrire les architectures d'entreprise publié par The Open Group (qui publie également le TOGAF).
- ArchiMate est composé d'éléments, classés selon deux axes :
  - Les couches
  - Les aspects
- ArchiMate présente un ensemble clair de concepts au sein des domaines d'architecture et de relations entre eux, et offre une structure simple et uniforme pour décrire le contenu de ces domaines.
- ArchiMate se distingue d'autres langages tels que le langage de modélisation unifié (UML) et la notation de modélisation des processus d'entreprise (BPMN) par son méta modèle bien défini et sa portée plus large de modélisation d'entreprise.

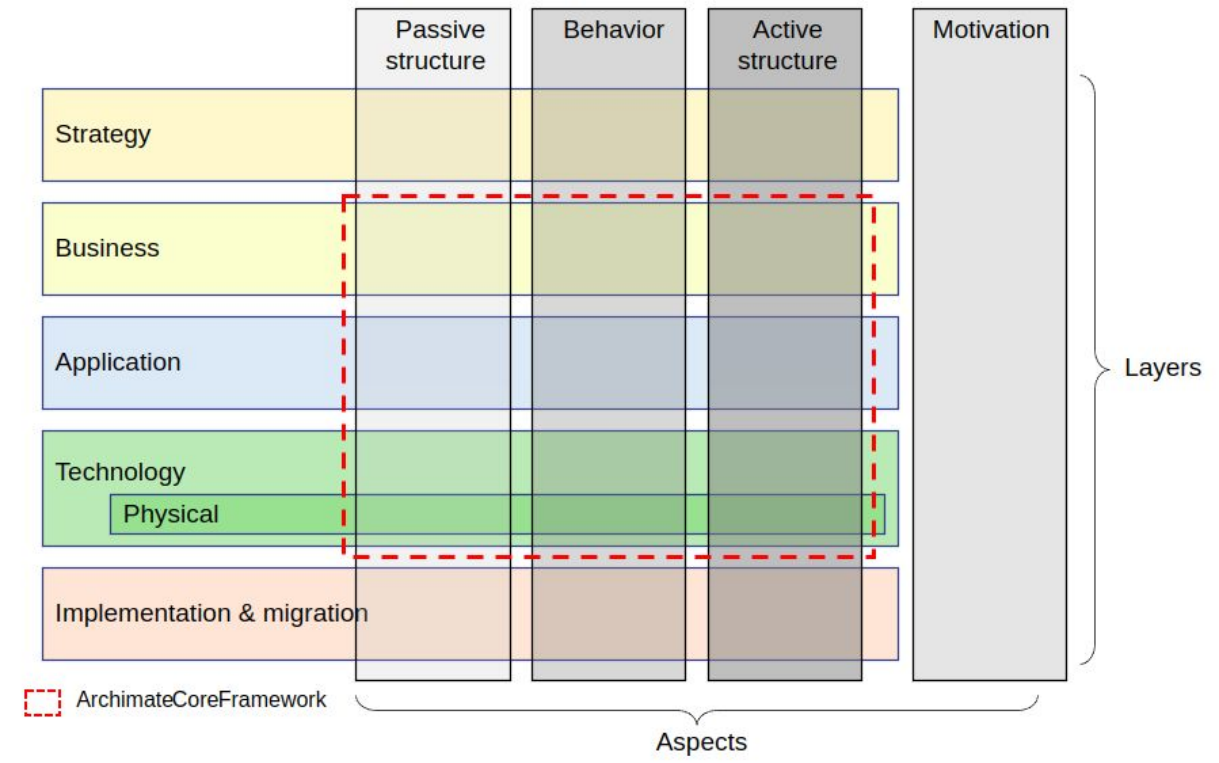
ArchiMate : les couches

- La couche “Business” décrit les services délivrés aux clients, mais également les processus métiers, les acteurs de ces processus, …
- La couche “Application” décrit les applications, ainsi que les données supportant les processus métiers.
- La couche “Technology” regroupe tous les services nécessaires à l’hébergement des applications (réseau, stockage, …). La sous-couche physique décrit le matériel réalisant ces services (switches/routeurs, serveurs, …)
- Les éléments de la couche “Strategy” permettent la modélisation de la stratégie ou des choix de l’entreprise, ayant potentiellement un impact sur l’architecture.
- La couche “Implementation & Migration” décrit, comme son nom l’indique, la réalisation de l’architecture définie. On peut y retrouver les différentes étapes du passage d’une ancienne, à une nouvelle architecture.

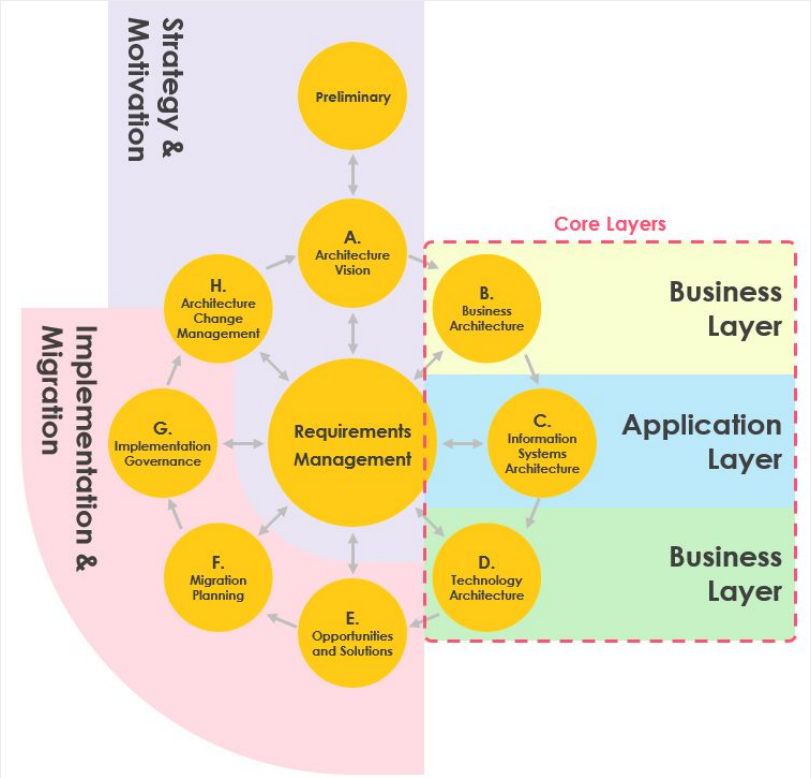
ArchiMate : les aspects

- L'aspect "Active Structure" représente les éléments structurels (les intervenants, les applications, le matériel, ...),
- L'aspect "Behavior" décrit les rôles et les comportements des éléments de l'aspect "Active Structure". On parle de processus, de fonctions, ...
- L'aspect "Passive Structure" inclut tous les éléments sur lesquels s'appliquent les "comportements". En fonction de la couche, on peut y trouver les produits, les données, ...
- L'aspect "Motivation" est un peu à part. Il regroupe tous les éléments qui ont conduit aux décisions prises. On y trouve les motivations, les principes d'architecture, la stratégie de l'entreprise, ...

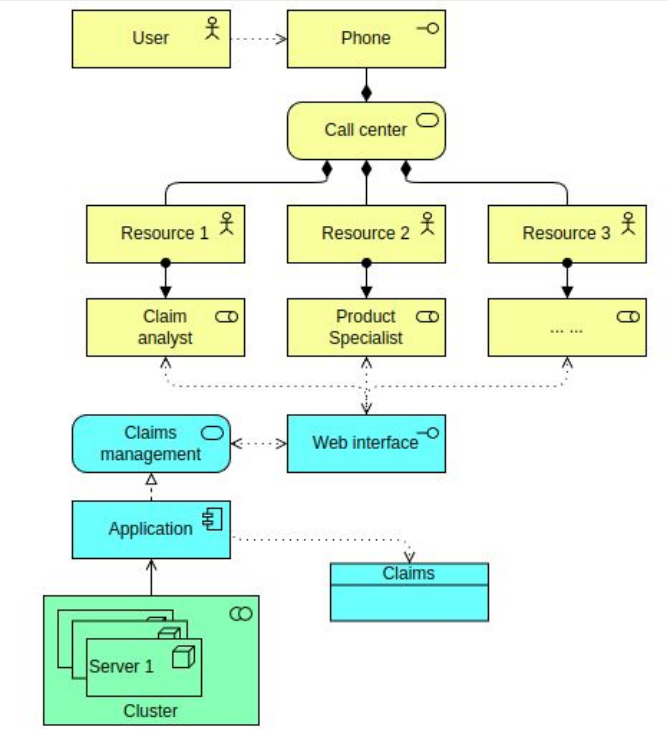
ArchiMate : la structure



ArchiMate : le périmètre



ArchiMate : exemple de schéma



ArchiMate : plus en savoir plus

- http://www.hosiaisluoma.fi/ArchiMate-Cookbook.pdf : plus de 60 pages d'exemples
- https://archimatetool.gitbook.io/quick_guide/

BPMN

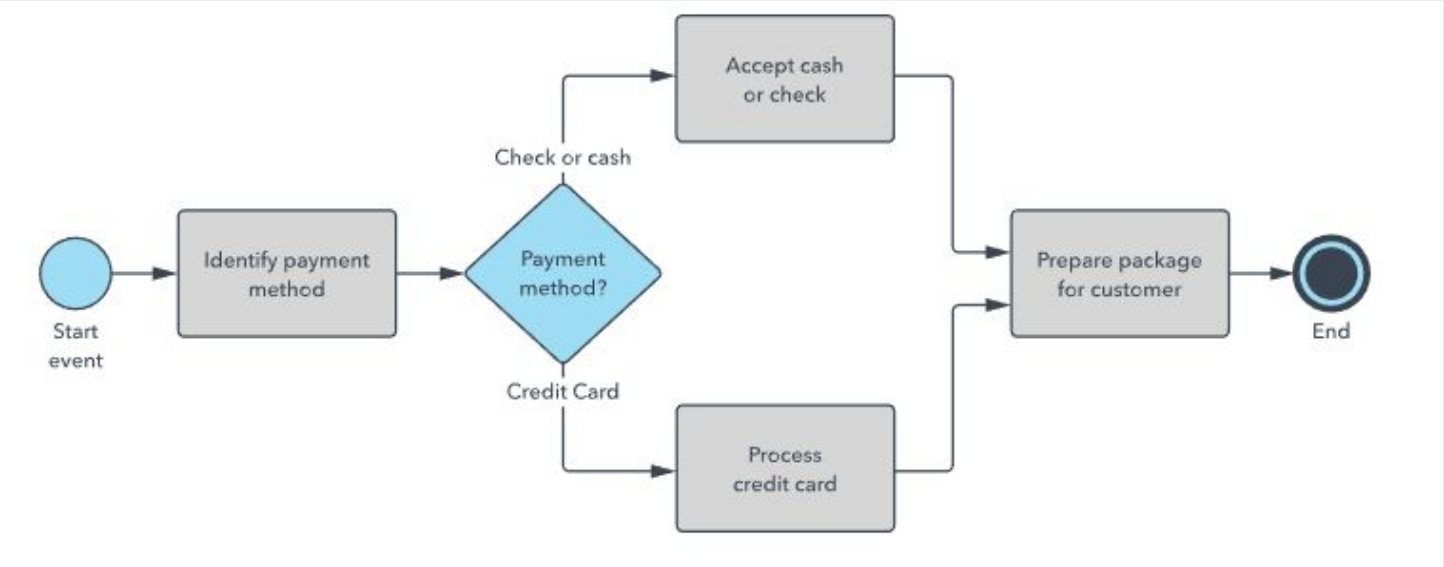
- BPMN : Business Process Model and Notation, créé en 2004.
  maintenu par l'OMG (Object Management Group) depuis 2005
  version actuelle : BPMN 2.0 depuis 2011
  norme ISO/CEI 19510:2013 depuis 2013

- modélise de A à Z les étapes d'un processus métier planifié

- la notation BPMN est utile à toutes les parties prenantes d'un processus métier, car elle permet de mieux le comprendre grâce à une représentation visuelle accessible de toutes ses étapes.

- permet de créer des documents XML (Extensible Markup Language) nécessaires à l'exécution de divers processus.
  L'une des principales normes XML est appelée Business Process Execution Language for Web Services ou, en version abrégée, BPEL/BEPEL4WS.

BPMN : exemple (diagramme de processus)



Ressemble à un diagramme UML d'activités

BPMN

4 types de diagrammes :

- diagramme de processus (voir exemple précédent)
- diagramme de chorégraphie : indique les interactions entre deux ou plusieurs participants. Il peut également être étendu à des sous-chorégraphies.
- diagramme de collaboration : indique les interactions entre deux processus ou plus.
- diagramme de conversation : en général, il s'agit d'une version simplifiée d'un diagramme de collaboration.
  Il illustre un groupe de messages liés échangés au sein d'un processus métier.

Représenter l'architecture logicielle

- UML n'offre pas toujours le bon niveau de détails ni les bons éléments de représentation.
  Kruchten 4+1 et Archimate sont plutôt orientés architecture d'entreprise
  BPMN est plutôt orienté architecture métier

- Il est donc utile d'avoir des schémas spécifiques orientés architecture logicielle

Représenter l'architecture logicielle : bonnes pratiques (1 / 2)

- Utiliser un nombre réduit de types de diagrammes.
  - diagramme d'architecture d'application : pour représenter les couches applicatives, les composants et les liens entre eux
  - diagramme d'architecture d'intégration : pour mettre en évidence les protocoles utilisés pour la communication entre les composants
  - diagramme d'architecture de déploiement : pour représenter la répartition physique du matériel et des logiciels dans le système.
    Le but est de visualiser comment le système sera déployé sur les équipements.
  - diagramme d'architecture DevOps : représente les flux opérationnels de déploiement d'applications, ressemble à un diagramme de processus.
  - diagramme d'architecture de données : illustre la manière et l'endroit où les données circulent, sont traitées et sont utilisées.

Représenter l'architecture logicielle : bonnes pratiques (2 / 2)

- Conserver une cohérence structurelle et sémantique entre les diagrammes
  - Chaque diagramme doit être cohérent avec les autres en termes de formes, de bordures, de lignes, de couleurs, etc.
  - L'idéal est de s'en tenir à un outil de création de diagrammes commun et de le réutiliser pour tous les projets.
- Maintenir la traçabilité des diagrammes
- Ajouter des légendes à côté des diagrammes architecturaux

Exemples de diagramme

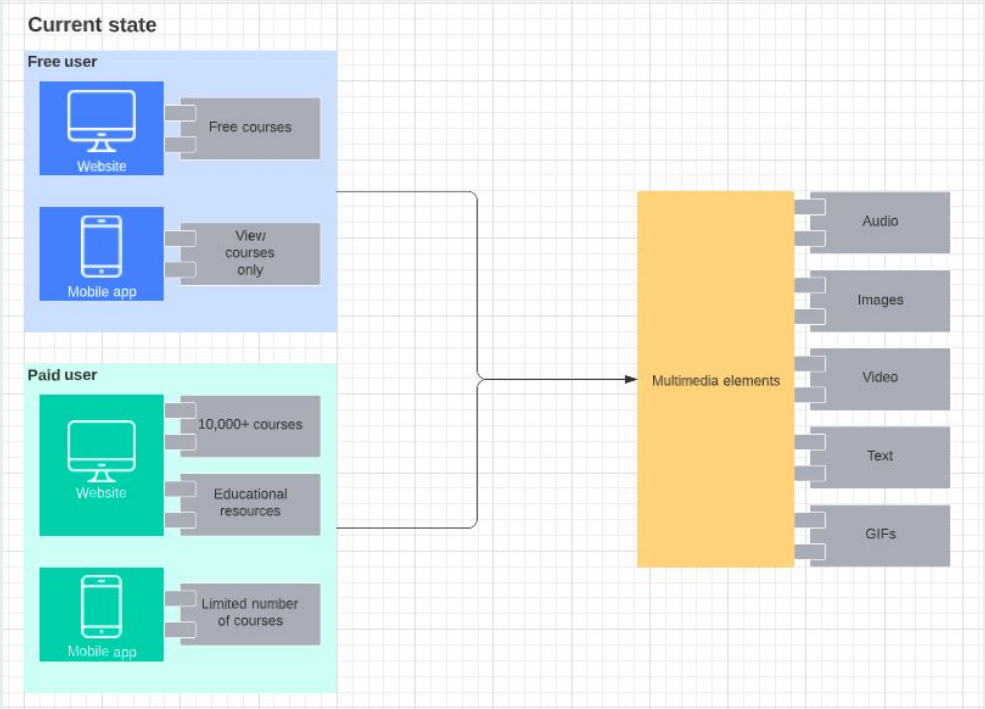


diagramme d'architecture d'application - modèle Lucidchart

Exemples de diagramme

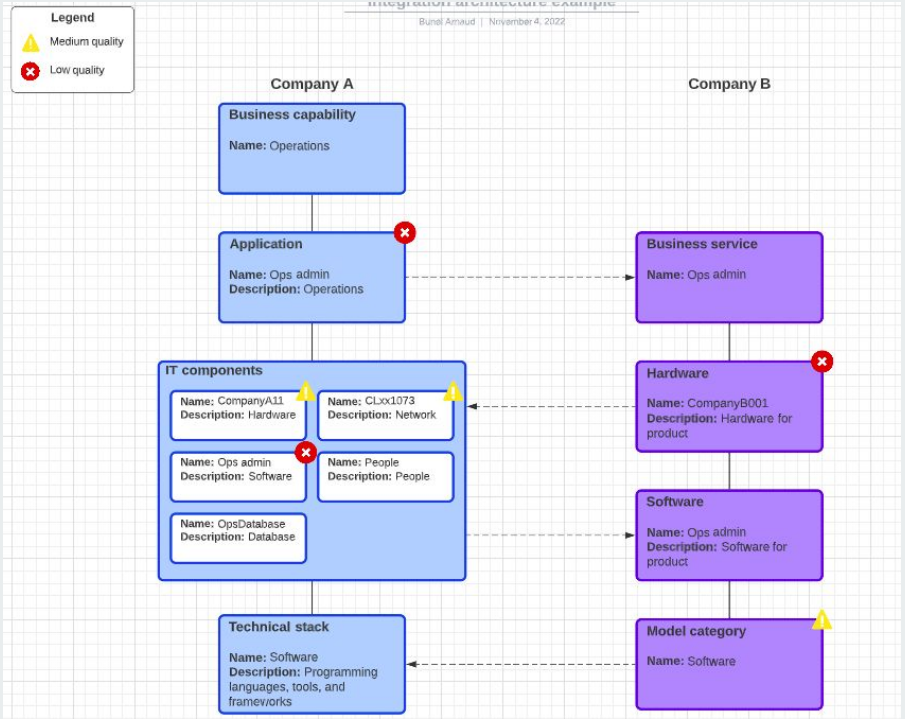


diagramme d'architecture d'intégration - modèle Lucidchart

Exemples de diagramme

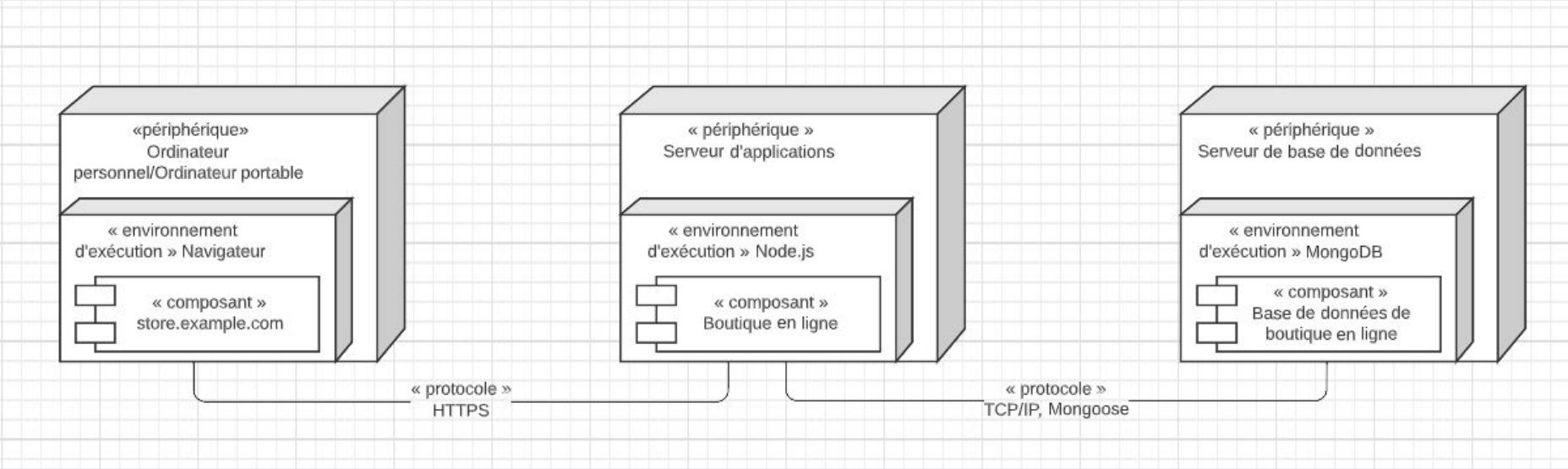
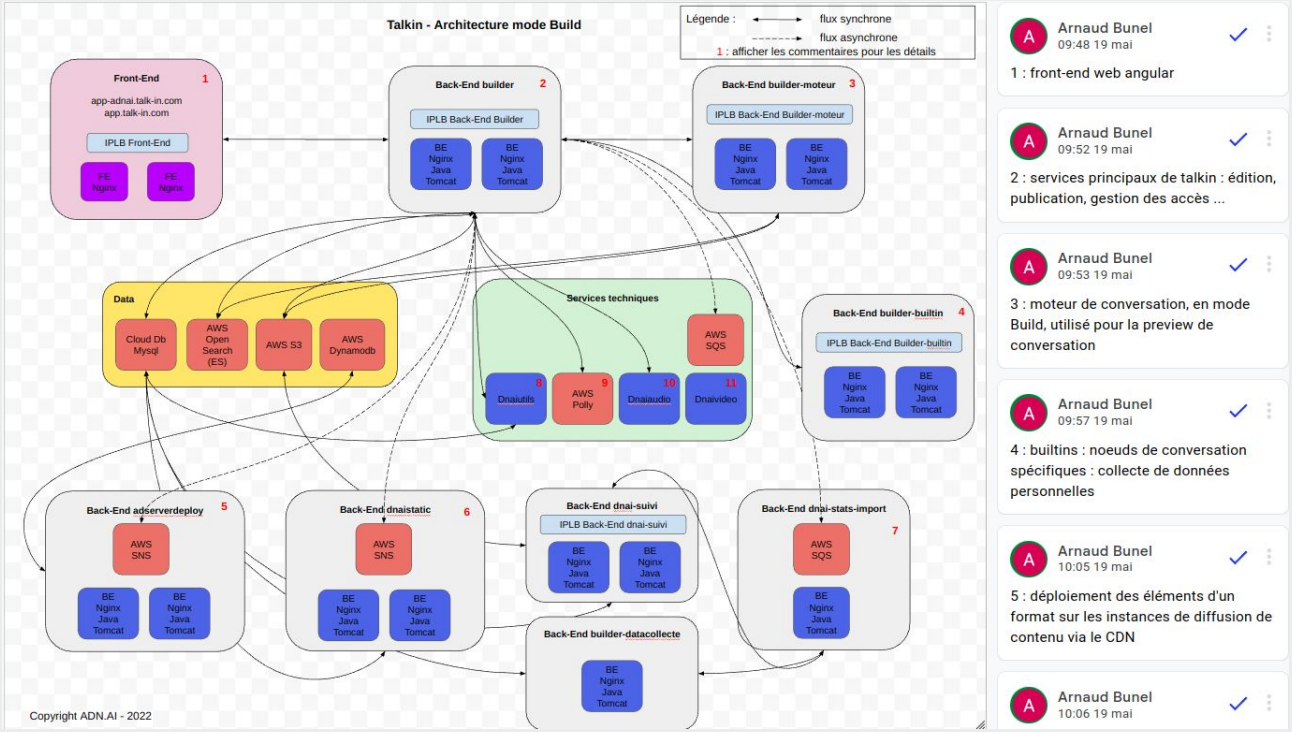


diagramme d'architecture de déploiement - modèle Lucidchart

Exemples de diagramme



Arnaud Bunel
09:48 19 mai
1 : front-end web angular

Arnaud Bunel
09:52 19 mai
2 : services principaux de talkin : édition, publication, gestion des accès ...

Arnaud Bunel
09:53 19 mai
3 : moteur de conversation, en mode Build, utilisé pour la preview de conversation

Arnaud Bunel
09:57 19 mai
4 : builtins : noeuds de conversation spécifiques : collecte de données personnelles

Arnaud Bunel
10:05 19 mai
5 : déploiement des éléments d'un format sur les instances de diffusion de contenu via le CDN

Arnaud Bunel
10:06 19 mai

diagramme d'architecture d'intégration avec légende et commentaires adn.ai

Exemples de diagramme

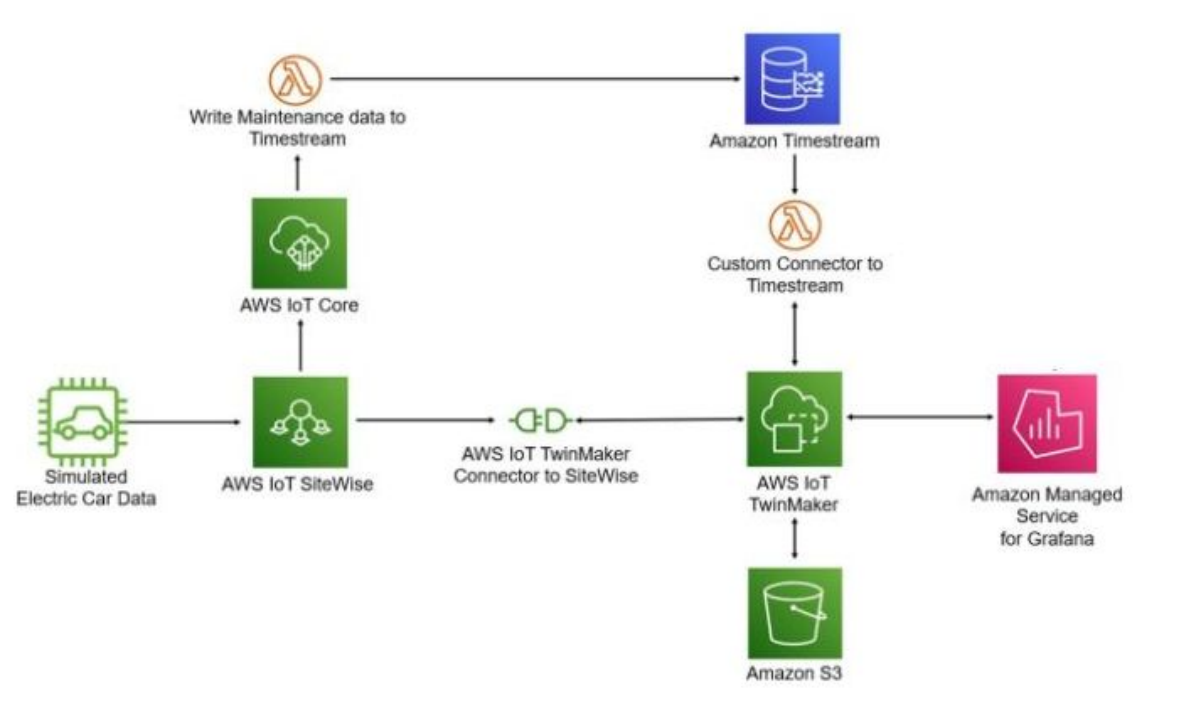


diagramme d'architecture d'application AWS.

Source : https://aws.amazon.com/fr/what-is/architecture-diagramming/

Exemples de diagramme

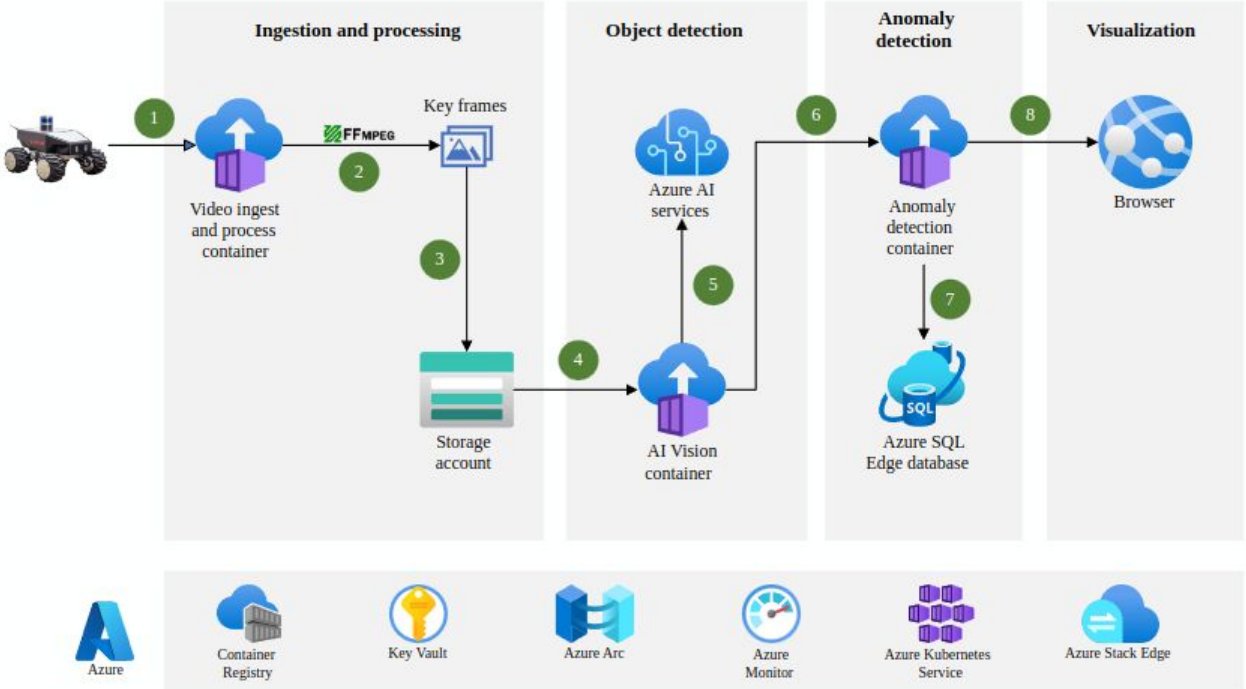


diagramme d'architecture Azure.

Source : https://learn.microsoft.com/en-us/azure/architecture/ai-ml/idea/video-ingestion-object-detection-edge-cloud

Représenter l'architecture logicielle : quelques outils

- Il existe des dizaines d'outils gratuits, payants, en ligne ou à installer

- gratuits (ou freemium) et en ligne :
  - Google Drawings (Google Drive)
  - Draw.io (permet de choisir le stockage) : https://app.diagrams.net/
  - Lucidchart
  - Creatly

- les outils payants d' éditeurs
  - IBM Rational Software Architect Modeler (IBM Rational)
  - Mega (Mega Software)
  - PowerAMC (Sybase)

Partie 4 : Références

- UML
  - https://www.lucidchart.com/pages/fr/langage-uml
  - https://gitmind.com/fr/types-diagrammes-uml.html
  - https://www.omg.org/spec/UML/2.5.1/About-UML
- Kruchten
  - https://medium.com/javarevisited/4-1-architectural-view-model-in-software-ec407bf27258
- Archimate
  - http://www.hosiaisluoma.fi/ArchiMate-Cookbook.pdf
  - https://archimatetool.gitbook.io/archimate_learn_by_example/
  - https://emmanuelgeorjon.com/architecture/archimate-presentation/
- BPMN
  - https://www.lucidchart.com/pages/fr/bpmn
  - https://fr.wikipedia.org/wiki/Business_process_model_and_notation
  - https://www.bpmn.org/
- Représentations
  - https://www.lucidchart.com/blog/fr/les-diagrammes-d-architecture-logicielle
  - https://www.infoq.com/fr/articles/crafting-architectural-diagrams/
- Les outils
  - https://geekflare.com/fr/create-application-architecture-diagram/