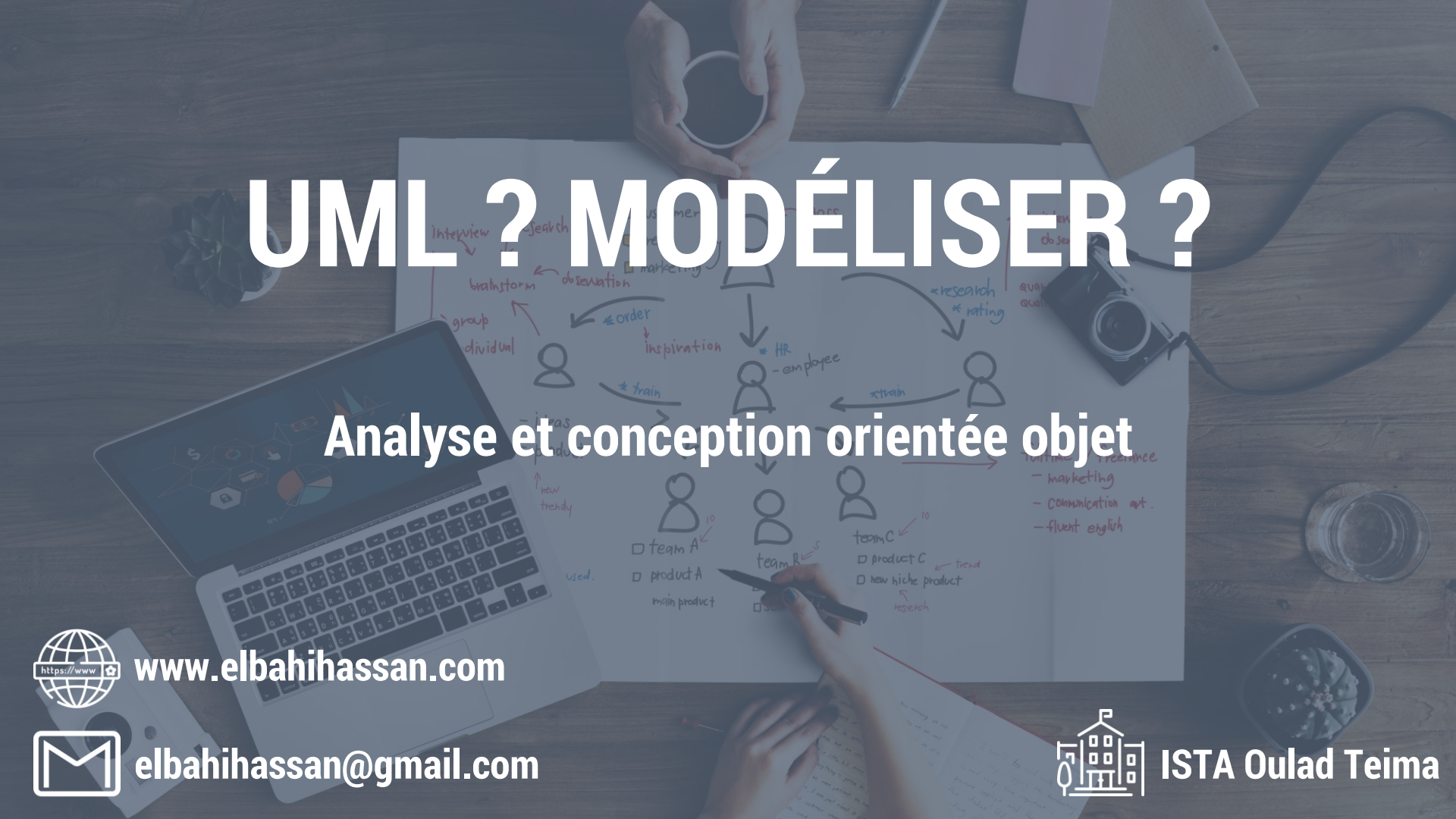


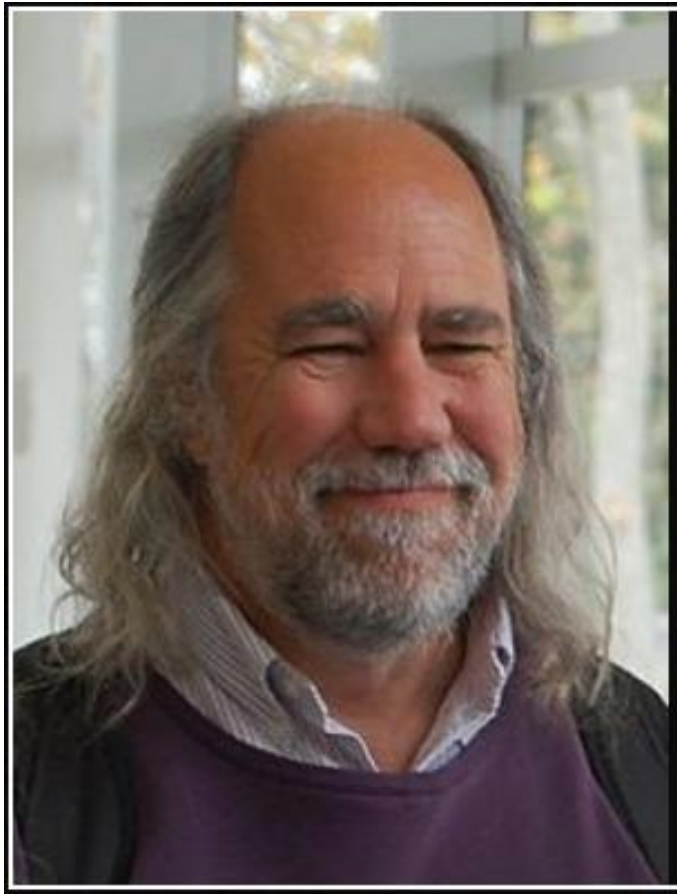
UML ? MODÉLISER ?

Analyse et conception orientée objet

www.elbahihassan.com

elbahihassan@gmail.com

ISTA Oulad Teima



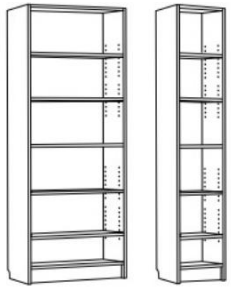
UML is not dessert topping and floor wax.

— Grady Booch —

AZ QUOTES

Pourquoi modéliser ?

- Modéliser, c'est décrire de manière visuelle et graphique les besoins et, les solutions fonctionnelles et techniques de votre projet logiciel. Mais modéliser pour quoi faire ?
- Avez-vous déjà eu à constituer un meuble en kit? Cette tâche sera très difficile constituons page de texte.

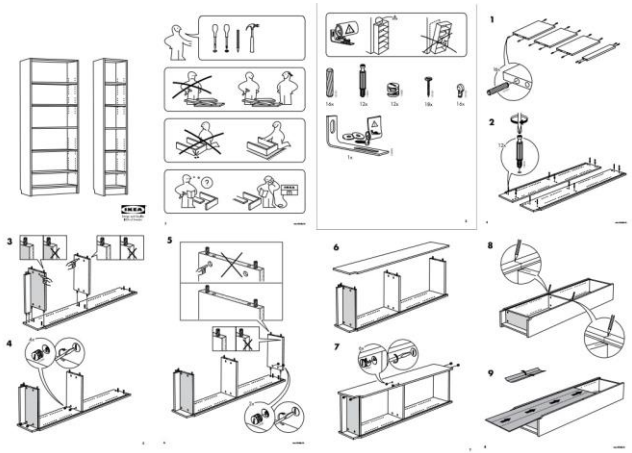


The owner of a contract repair shop for small aircraft engines is faced with the problem of scheduling the various repair operations required on each engine. The type of repairs required varies from engine to engine. For various technical reasons, the repairs on each engine must be made in a specified sequence of operations, which are performed at different "stations" in the shop, each station being manned by one mechanic. Thus, for example, one engine requiring 4 operations might follow the sequence "station 1, station 2, station 4, station 3", while another might follow the sequence "station 2, station 3, station 1, station 4."

A given engine does not necessarily pass through every station, and some engines may be routed through a given station more than once. As a result of equipment limitations there is only 1 station of each type, and the nature of the equipment is such that only 1 engine may be worked on at a station at any given time.

Pourquoi modéliser ?

- Toutefois, vous serez d'accord que c'est plus facile à partir de schéma plutôt qu'une page de texte.



- Cet exemple nous démontre que l'utilisation de schémas et d'illustrations rend quelque chose de complexe plus compréhensible.

Pourquoi modéliser ?

- C'est exactement ce à quoi UML sert dans des projets de réalisation de logiciels.

- Un document de texte décrivant de façon précise ce qui doit être réalisé contiendrait plusieurs dizaines de pages. En général, peu de personnes ont envie de lire ce genre de document. De plus, un long texte de plusieurs pages est source d'interprétations et d'incompréhension.

- UML nous aide à faire cette **description de façon graphique** et devient alors un excellent moyen pour « **visualiser** » le(s) futur(s) logiciel(s).

Pourquoi modéliser ?

- Un logiciel qui a été réalisé sans analyse et sans conception (étapes où l'on modélise le futur logiciel) risque lui aussi de ne pas répondre aux **besoins**, de comporter des anomalies et d'être très difficile à maintenir.

- Tout comme la construction d'une maison nécessite des plans à différents niveaux (vision extérieure, plan des différents étages, plans techniques…), la réalisation d'un logiciel ou d'un ensemble de logiciels a besoin d'un certain nombre de **diagrammes**.

Méthodes de conception

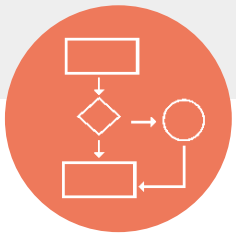
On parle principalement de trois types de méthodes de conception :

MÉTHODES FONCTIONNELLES

- SADT,
- Warnier,
- ...

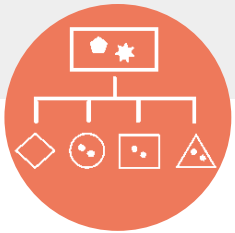
MÉTHODES SYSTÉMIQUES

- MERISE,
- AXIAL,
- ...



MÉTHODES ORIENTÉES OBJETS

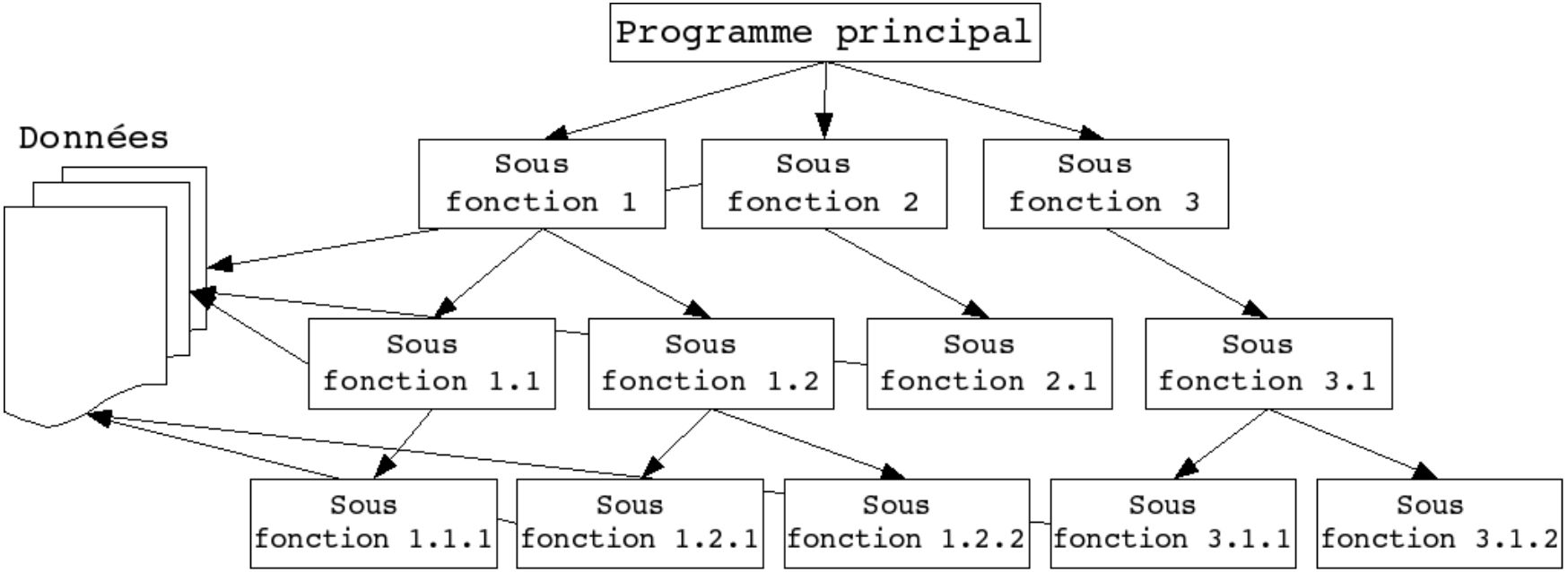
- UML



Méthodes fonctionnelles

- Les méthodes fonctionnelles (également qualifiées de méthodes structurées) trouvent leur origine dans les langages procéduraux. Elles mettent en évidence les fonctions à assurer et proposent une approche hiérarchique descendante et modulaire.
- Les méthodes fonctionnelles ou cartésiennes consistent à décomposer hiérarchiquement une application en un ensemble de sous applications. Les fonctions de chacune de ces dernières sont affinées successivement en sous fonctions simples à coder dans un langage de programmation donné.

Méthodes fonctionnelles



Représentation graphique d'une approche fonctionnelle.

Méthodes fonctionnelles

Les points forts des méthodes fonctionnelles sont les suivants :

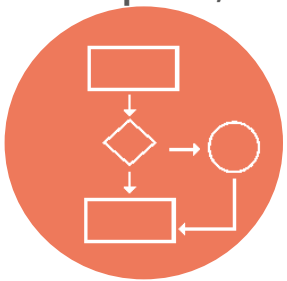
- **Simplicité** du processus de conception préconisé,
- Capacité à répondre **rapidement** aux besoins ponctuels de leurs utilisateurs.

Les points faibles des méthodes fonctionnelles sont les suivants :

- Une simple mise à jour du logiciel à un point donné, peut impacter en cascade une multitude d'autres fonctions.
- Éventuelle redondance des données

Méthodes systémiques

- Les méthodes systémiques proposent une double démarche de modélisation : la modélisation des **données** et celle des **traitements**. Elle est influencée par les systèmes de gestion de bases de données.
- MERISE est un exemple d'une méthode systémique. Elle permet de faire l'analyse et la conception des systèmes d'information basée sur le principe de la séparation des données et des traitements.
- Elle possède plusieurs modèles qui sont répartis sur 3 niveaux : Le niveau conceptuel, le niveau logique ou organisationnel, le niveau physique.



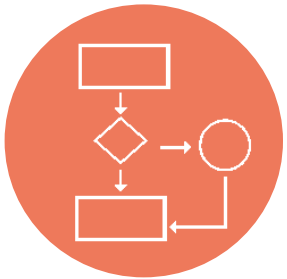
Méthodes systémiques

Les points forts des méthodes systémiques sont les suivants :

- Approche **globale** qui prend en compte la modélisation des données et des traitements.
- Introduction des **niveaux d'abstraction** dans le processus de conception (niveau conceptuel, niveau logique et niveau physique),

Les points faibles des méthodes systémiques sont les suivants :

- Double démarche de conception : les données et les traitements.
- Pas de fusion possible des deux aspects (données et traitements).

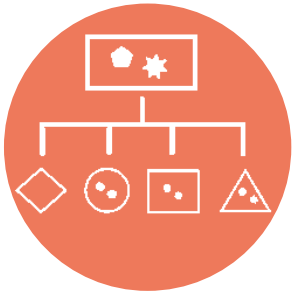


Méthodes fonctionnelles et systémiques

- Les méthodes fonctionnelles et systémiques sont potentiellement de type **descendant**.
- Elles ne favorisent pas les critères de :
  - **Réutilisabilité** : les modules ne sont pas généraux, mais adaptés aux sous problèmes pour lesquels ils ont été conçus.
  - **Extensibilité** : l'architecture du logiciel est fondée sur les traitements. Or ces derniers sont moins stables que les données d'où cette approche est inadaptée à la conception de gros logiciels.

Méthodes orientées objets

- L'approche orientée objet considère le logiciel comme une collection d'**objets** dissociés définis par des **propriétés**. Une propriété est soit un **attribut** soit une **opération**. Un objet comprend donc à la fois une structure de données et une collection d'opérations (son comportement).



Méthodes orientées objets

Vendeur de véhicules

Voiture

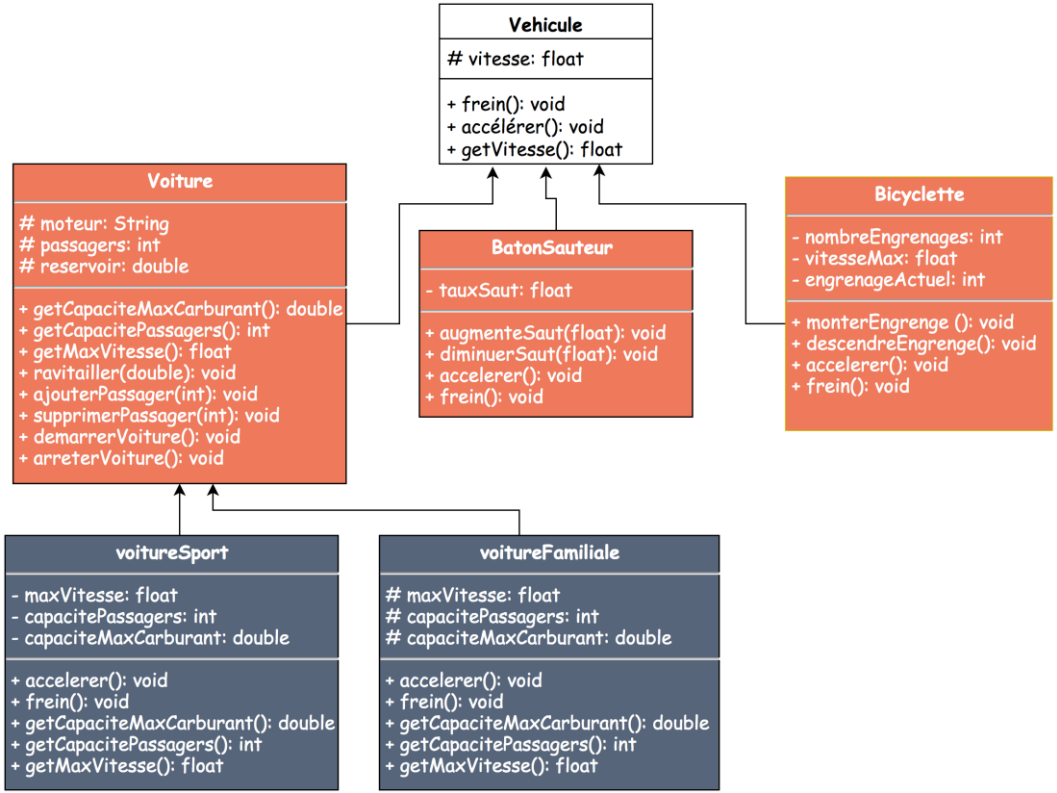
Bâton sauteur

Bicyclette

Voiture de sport

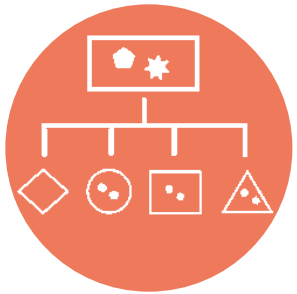
voiture familiale

Méthodes orientées objets

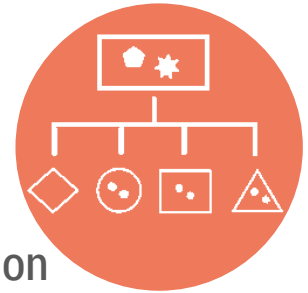


Méthodes orientées objets

- L'approche orientée objet est potentiellement de type **ascendant** : un effort de regroupement basé sur l'abstraction des données est entretenu tout au long du processus de conception.
- En effet, on commence par l'identification des **objets**. Ces objets sont regroupés dans des **classes** selon leurs propriétés. Ensuite ces classes sont à nouveau regroupées en classes plus abstraites appelées **modules** ou **sous-systèmes** jusqu'à la **modélisation du problème posé**.



Méthodes orientées objets



Les points forts des méthodes orientées objets sont les suivants :

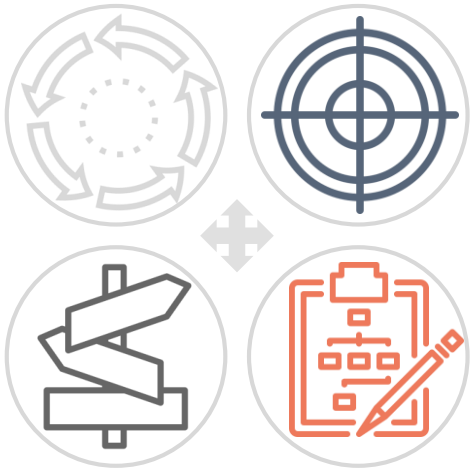
- Intégrer dans l'objet des données et des traitements
- Profiter des avantages des concepts objets : phase d'analyse et de conception
- Prendre en compte une plus large gamme d'applications
- Favoriser la conception et la réutilisation des composants : concevoir dans un but de réutilisation et non pas pour répondre à un besoin ponctuel.
- Améliorer la productivité et la rentabilité en utilisant des bibliothèques de composants réutilisables
- Simplifier le passage conceptuel/physique.
- Faciliter le prototypage.

Principes fondamentaux d'une approche objet.

- L'approche objet est une démarche qui s'organise autour de 4 principes fondamentaux. C'est une démarche :

ITÉRATIVE ET INCRÉMENTALE

CENTRÉE SUR L'ARCHITECTURE DU LOGICIEL



GUIDÉE PAR LES BESOINS DU CLIENT ET DES UTILISATEURS

DÉCRIT LES ACTIONS ET LES INFORMATIONS DANS UNE SEULE ENTITÉ

Itérative et incrémentale

Principes fondamentaux d'une approche objet.

- La modélisation d'un logiciel se fera en plusieurs fois.
- Les diagrammes ne seront pas réalisés de façon complètement satisfaisante dès la première fois. Il nous faudra plusieurs versions d'un même diagramme pour obtenir la version finale.
- Chaque version est le fruit d'une itération (un nouveau passage sur les diagrammes).
- Une itération permet donc d'affiner la compréhension du futur logiciel.

Guidée par les Besoins du client

Principes fondamentaux d'une approche objet.

- Les besoins d'un client, dans un projet logiciel sont les exigences exprimées par le client au niveau des fonctionnalités, du rendu et des actions d'un logiciel.
- Par exemple, lors de la réalisation du progiciel Word, les clients ont probablement souhaité un logiciel qui leur permettrait d'écrire du texte, de l'effacer, d'intégrer des images, enregistrer le contenu, le supprimer, etc.

Guidée par les Besoins du client

Principes fondamentaux d'une approche objet.

Les besoins des utilisateurs servent de fil rouge, tout au long du cycle de développement :

- Lors de la phase **d'analyse** pour la clarification, l'affinage et la validation des besoins des utilisateurs ;
- Lors de la phase **de conception** et **de réalisation** pour la vérification de la prise en compte des besoins des utilisateurs ;
- Lors de la phase de **test** afin de garantir la satisfaction des besoins.

Centrée sur l'architecture du logiciel

Principes fondamentaux d'une approche objet.

- L'architecture logicielle décrit les choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).
- On peut citer l'architecture client serveur.

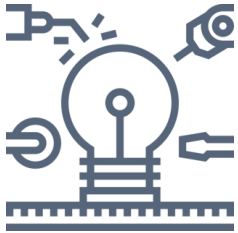
Entité = Informations + actions

Principes fondamentaux d'une approche objet.

- Les différents **diagrammes** utilisés en UML donnent tous une vision particulière du logiciel à développer.
- Parmi ces diagrammes, il en est un qui représente particulièrement bien ce qui est à développer si on opte pour un développement objet. Il s'agit du diagramme de **classes**.
- le diagramme de classes donnera une **vision** assez claire des informations qui seront utilisées par le logiciel, mais également des **fonctions** (ou opérations) qui devront s'appuyer sur ces informations.
- L'approche objet **mélange** donc habillement l'**analyse** des informations et des **actions** au lieu de les analyser séparément.

Projet informatique

- Un projet informatique nécessite une phase d'analyse, suivi d'une étape de conception.



ANALYSE **CONCEPTION**

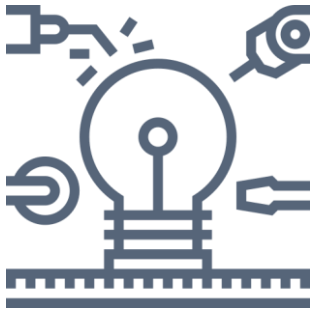
La phase d'analyse

- Dans cette phase, on cherche d'abord à bien comprendre et à décrire de façon précise les besoins des utilisateurs ou des clients.
    - Que souhaitent-ils faire avec le logiciel ?
    - Quelles fonctionnalités veulent-ils ?
    - Pour quel usage ?
    - Comment l'action devrait-elle fonctionner ?

- C'est ce qu'on appelle « **l'analyse des besoins** ». Après validation de notre compréhension du besoin, nous imaginons la solution. C'est la partie **analyse de la solution**.

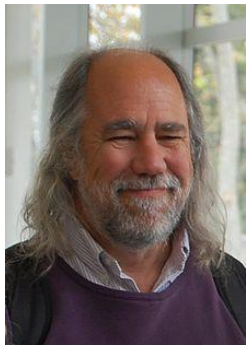
La phase de conception

- Dans cette phase, on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques, tels que l'installation des différentes parties logicielles à installer sur du matériel.
- Pour réaliser ces deux phases dans un projet informatique, nous utilisons des méthodes, des conventions et des notations. UML fait partie des notations les plus utilisées aujourd'hui.



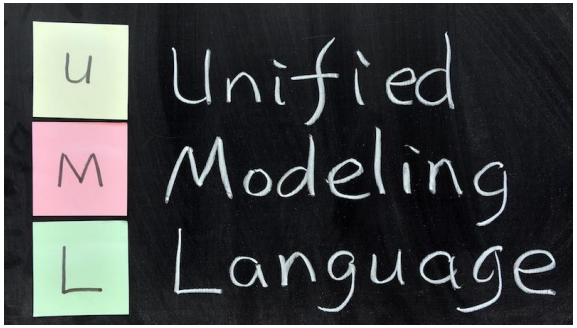
Langage UML

- UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ».
- UML est né de la fusion des trois méthodes qui ont influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE.
- Il est proposé par : Grady Booch, James Rumbaugh et Ivar Jacobson.



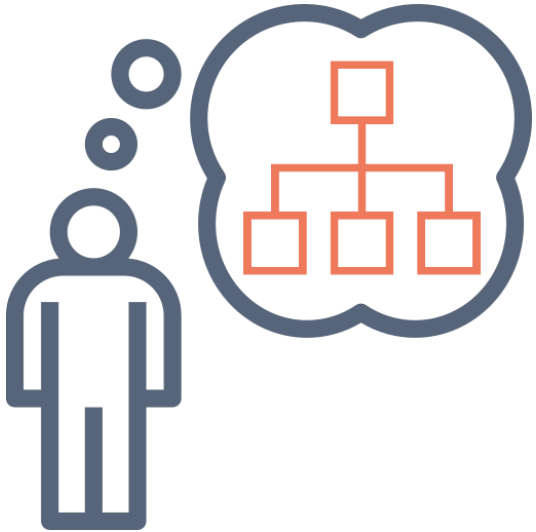
Langage UML

- UML est surtout utilisé lorsqu'on prévoit de développer des applications avec une démarche objet (développement en Java, en C++, etc.).
- La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter.
- UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc..

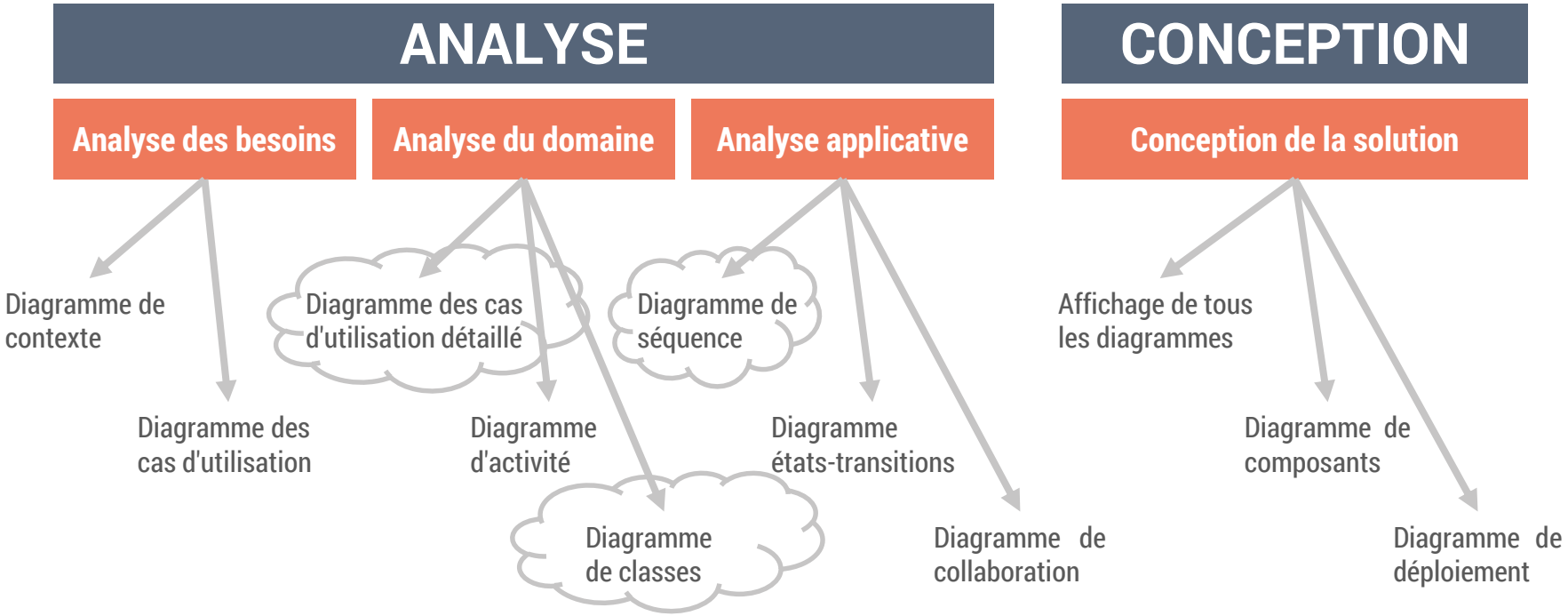


Langage UML

- Réaliser ces diagrammes revient donc à modéliser les besoins du logiciel à développer.
- Le langage UML ne demande aucune démarche, ce n'est donc pas une méthode.
- Chacun est libre d'utiliser les types de diagramme qu'il souhaite, dans l'ordre qu'il veut.
- Il suffit que les diagrammes réalisés soient cohérents entre eux, avant de passer à la réalisation du logiciel.



Positionnement de ce cours



Résumé

- UML c'est un langage de **modélisation** qui permet de représenter graphiquement les **besoins** des utilisateurs. On utilise pour cela des **diagrammes**.
- UML est une démarche qui se base sur une approche **objet**. Cette approche s'appuie sur 4 principes fondamentaux :
    - L'approche objet nécessite une démarche itérative et incrémentale.
    - L'approche objet est guidée par les besoins du client.
    - La démarche est également centrée sur l'architecture du logiciel.
    - L'approche objet décrit les informations et les actions dans une seul entité.


MEDELLIN CARTEL
GUSTAVO
GACHA
PABLO ESCOBAR
OCHOAS
LEHDER
HERMILDA TATA
GACHA'S PARTNER
ARETE
POISON
GALEANO
KIKO MONGADA
COCKROACH
LION
SICARIOS