

# Modélisation

## Premier Pas

### La notation UML

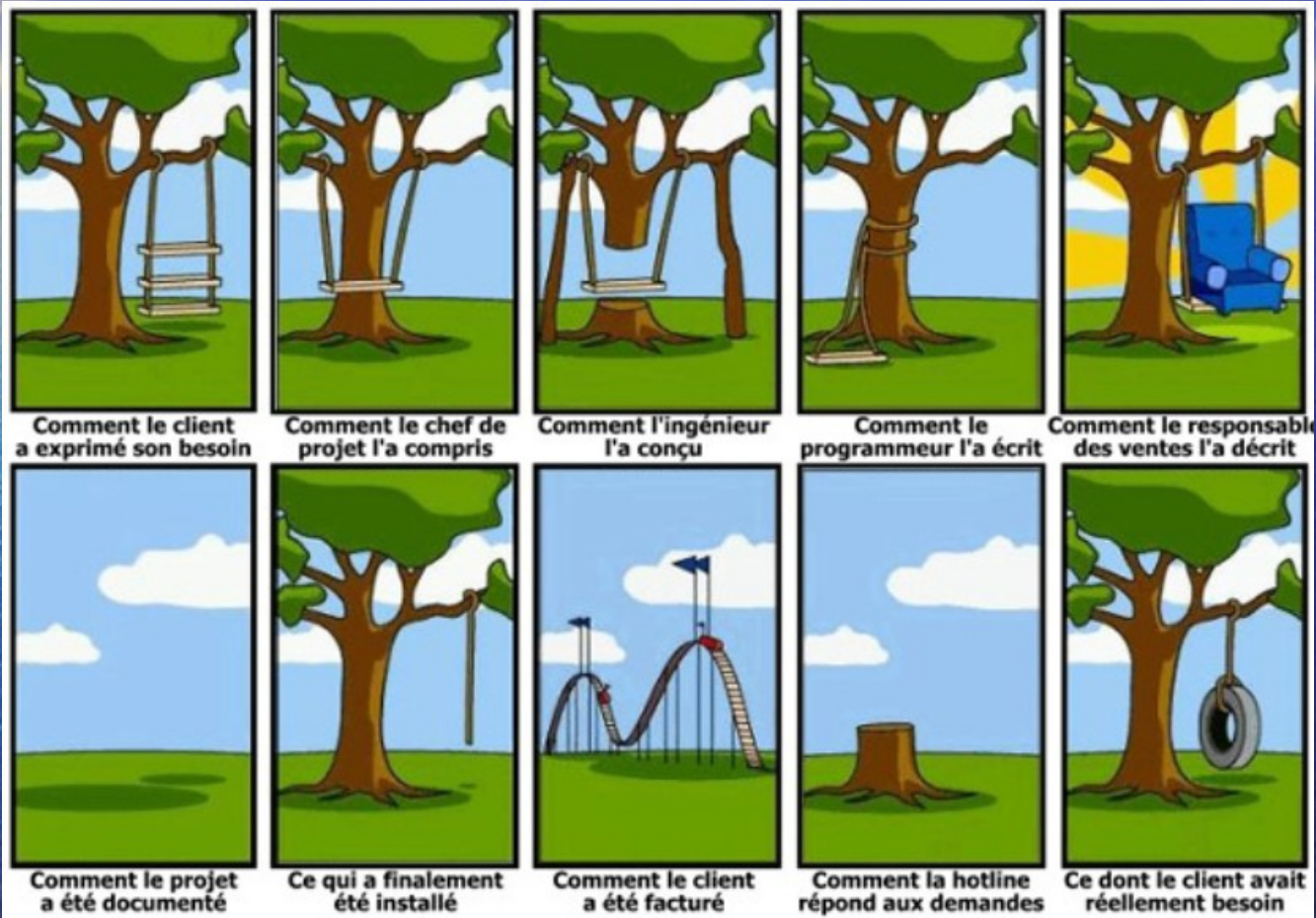
# Qualité d'un logiciel

## Validité

- La validité (correction, justesse, conformité) est la capacité que possède un produit logiciel à remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.
- Adéquation entre :
  - Le besoin effectif de l'utilisateur
  - Les fonctions offertes par le logiciel
- **Même le logiciel le mieux conçu techniquement, s'il ne rend pas les services escomptés, est inutile et son développement aura été du temps perdu.**



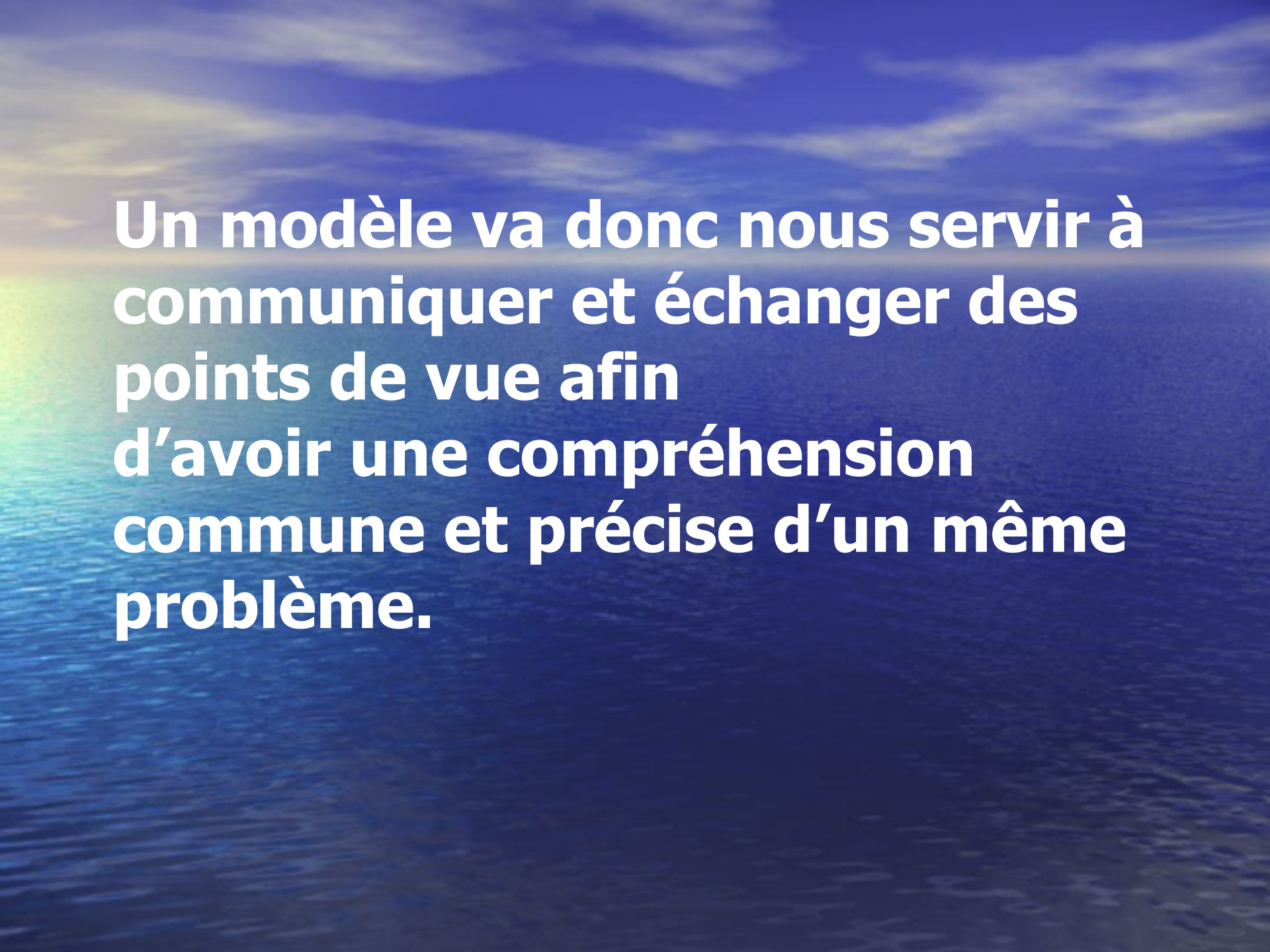
# Problèmes de conception



# Qu'est ce qu'un modèle ?

- Un modèle est avant tout une représentation abstraite du monde réel.
  - A ce titre, ce n'est pas le monde réel et donc ce n'est pas, exactement le monde réel.
  - Un modèle offre donc une vision schématique d'un certains nombre d'éléments que l'on veut décrire ; un dessin quoi !
- On a coutume de dire : « Un dessin vaut mieux qu'un beau discours » ;
- et bien écoutons et tentons de comprendre nos ancêtres, nos parents ou tout ceux qui ont pu nous répéter cette phrase.
- Réaliser un modèle c'est avant tout dessiner ce que l'on a compris d'un problème dans une syntaxe précise (la syntaxe UML ou Merise).





**Un modèle va donc nous servir à  
communiquer et échanger des  
points de vue afin  
d'avoir une compréhension  
commune et précise d'un même  
problème.**

# Les niveaux d'abstraction

- Lorsque l'on a à traiter un problème complexe, il est conceptuellement impossible de l'appréhender d'un seul bloc, dans son ensemble.
  - Notre esprit a besoin de « dégrossir »
  - Une fois le problème découpé en sous-problèmes, l'analyse de chacun de ces sous-problèmes nécessite de commencer à en comprendre les grandes lignes puis d'affiner sa compréhension pour enfin comprendre tous les détails.
- Ce mode de raisonnement est à la base même des niveaux d'abstraction que l'on retrouve dans les méthodes comme Merise (niveau conceptuel, niveau logique, niveau physique) ou RUP1 (niveau fonctionnel, niveau analyse, niveau conception).

## **Avec l'expérience on peut se passer de modèles et gagner du temps dans la réalisation !**

- Un modèle, sert à comprendre le monde réel, le problème de votre maîtrise d'ouvrage !
- C'est aussi un moyen de poser ses idées « sur la table », de ne pas les oublier et de les partager avec les autres membres de l'équipe.
- Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc à ce type un vecteur privilégié pour communiquer.
- Imaginez en plus le cas d'équipes géographiquement distribuées et de nationalités différentes...



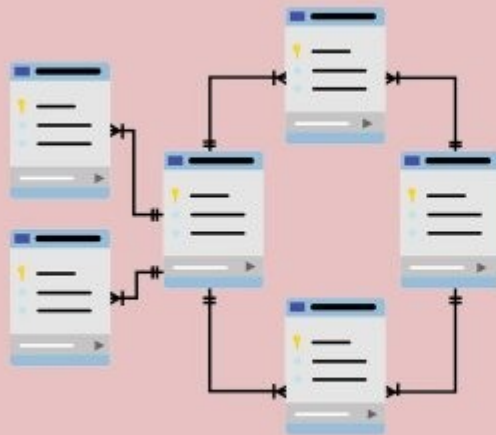
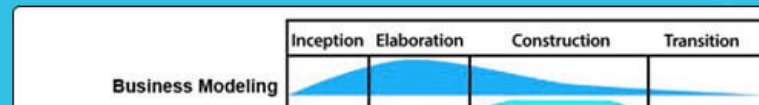
# Les méthodes agiles permettent de réaliser les spécifications en temps réel avec la MOA et le codage à la suite « Y'a pas de perte » !

- La communication ne s'arrête pas aux membres de l'équipe projet.
  - Une fois mise en production, l'application va devoir être maintenue, probablement par une autre équipe et pas nécessairement de la même société que celle ayant créée l'application.
  - ?? Comment fournir une documentation suffisante à cette équipe de maintenance ?
  - ?? Comment assurer l'homogénéité de cette documentation entre plusieurs projets pour faciliter les passages d'un projets à un autre (c'est la capitalisation ça ?!) ?



# Des méthodes et une notation pour modéliser et concevoir

## RUP Methodology



**MERISE**  
pour la Conception



**Unified  
Modeling  
Language**

# Les méthodes objet et la genèse d'UML

- **• Les premières méthodes d'analyse (années 70)**
  - Décomposition (fonctionnelle et hiérarchique d'un système).
- **L'approche systémique (années 80)**
  - Modélisation des données + modélisation des traitements (Merise, Axial).



# Les méthodes objet et la genèse d'UML

- **L'émergence des méthodes objet (1990-1995)**
  - Prise de conscience de l'importance d'une méthode
  - spécifiquement objet :
    - Encapsulation des données (la structure) et des traitements (le comportement)
- Plus de 50 méthodes objet sont apparues durant cette période (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE...)
  - Confusion autour de l'analyse et de la conception

# Les méthodes objet et la genèse d'UML

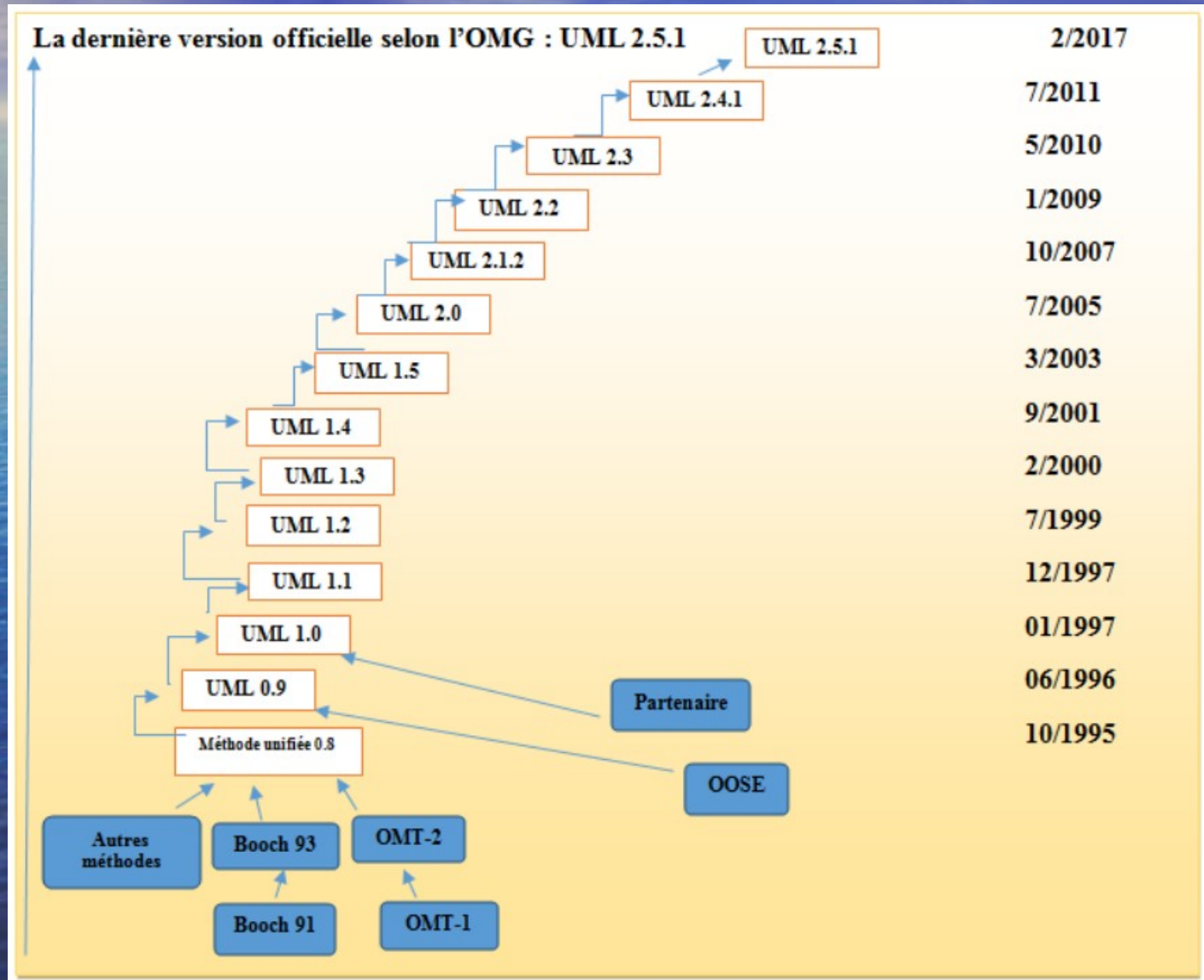
- Booch - catégories et sous-systèmes
- Embley - classes singletons et objets composites
- Fusion - description des opérations, numérotation des messages
- Gamma - frameworks, patterns et notes
- Harel - automates (statecharts)
- Jacobson - cas d'utilisation (use cases)
- Meyer - pré et post conditions
- Odell - classification dynamique, éclairage sur les événements
- OMT - associations
- Shlaer-Mellor - cycle de vie des objets
- Wirfs-Brook - responsabilités (CRC card)



# Les méthodes objet et la genèse d'UML

- L'expérience a permis de faire le tri parmi les méthodes existantes avec la constatation que les différences entre les méthodes s'amenuisent
- 1994: Rumbaugh et Booch décident d'unifier leurs travaux au sein d'une méthode unique: la méthode unifiée.
- 1995: Jacobson les rejoint
- Octobre 1995: Première version de la méthode unifiée (V0.8)
- Juin 1996: Deuxième version (V0.9)
- Octobre 1996: Version décisive avec une nette évolution: la *méthode unifiée* devient le *langage de modélisation unifié*.
- 1996: Consortium de partenaires
- 17 Janvier 1997: Version 1.0 soumise à standardisation

# Les méthodes objet et la genèse d'UML





# L'unification des méthodes: L'OMG

- Object Management Group
- Objectif de l'OMG: « maximiser la portabilité et l'interopérabilité des logiciels par la définition de standards industriels pour le développement d'applications commerciales orientées objets »
- le plus grand consortium d'industriels au monde (plus de 570 membres (1995))

# L'unification des méthodes : Objectifs d'UML

- Proposer un langage visuel de modélisation
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- Proposer des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base
- Être indépendant des langages particuliers de programmation
- Proposer une base formelle pour comprendre le langage de modélisation
- Encourager l'application des outils OO



# UML: Principales influences

- Booch - catégories et sous-systèmes
- Embley - classes singletons et objets composites
- Fusion - description des opérations, numérotation des messages
- Gamma - frameworks, patterns et notes
- Harel - automates (statecharts)
- Jacobson - cas d'utilisation (use cases)
- Meyer - pré et post conditions
- Odell - classification dynamique, éclairage sur les événements
- OMT - associations
- Shlear-Mellor - cycle de vie des objets
- Wirfs-Brock - responsabilités (CRC)

A wide-angle photograph of a calm ocean under a vast blue sky. A soft rainbow is visible on the horizon, with its colors blending into the sea. The water is a deep blue with gentle ripples. The sky is a lighter blue with wispy white clouds. The overall mood is peaceful and expansive.

Merci de votre attention