Introduction

Pierre-Alain Muller

ENSISA

pa.muller@uha.fr

03.89.33.69.65

Sommaire

- **Problématique**
  - Gestion de la complexité
  - Approches de modélisation
- Unification des méthodes objet
  - Genèse d'UML
- Concepts de base
  - Notation UML

Complexité des logiciels

- Les tendances
  - Programmation sans programmer
  - Micro-architectures (*patterns*)
  - Importance de l'architecture
  - Informatique distribuée
  - Multimédia

Complexité des logiciels

- La problématique du domaine
- Le processus de développement
- L’inhérente flexibilité du logiciel
- La caractérisation du comportement des systèmes discrets

Conséquence de la complexité

- Le risque de déficiences graves est élevé
- La mise au point est lente et chaotique
- La maintenance est disproportionnée
- Le coût est astronomique
- Crise du logiciel (OTAN Garmish 1968)

Classes de logiciels

- Logiciels amateurs
- Logiciels jetables (consommables)
- Logiciels de qualité industrielle
- Logiciel vitaux

Capacités de l’être humain

- Nombre de concepts manipulés (7 éléments plus ou moins 2)
- Vitesse de transfert (5 secondes entre paquets)
- Vitesse de traitement
- Vitesse et limitations des dispositifs d’entrée-sortie

Gestion de la complexité

- A défaut de réduire la complexité il faut la maîtriser
- Donner une illusion de simplicité
- Application de critères de partitionnement (accidentel, fonctionnel, logique...)

Rôle de la décomposition

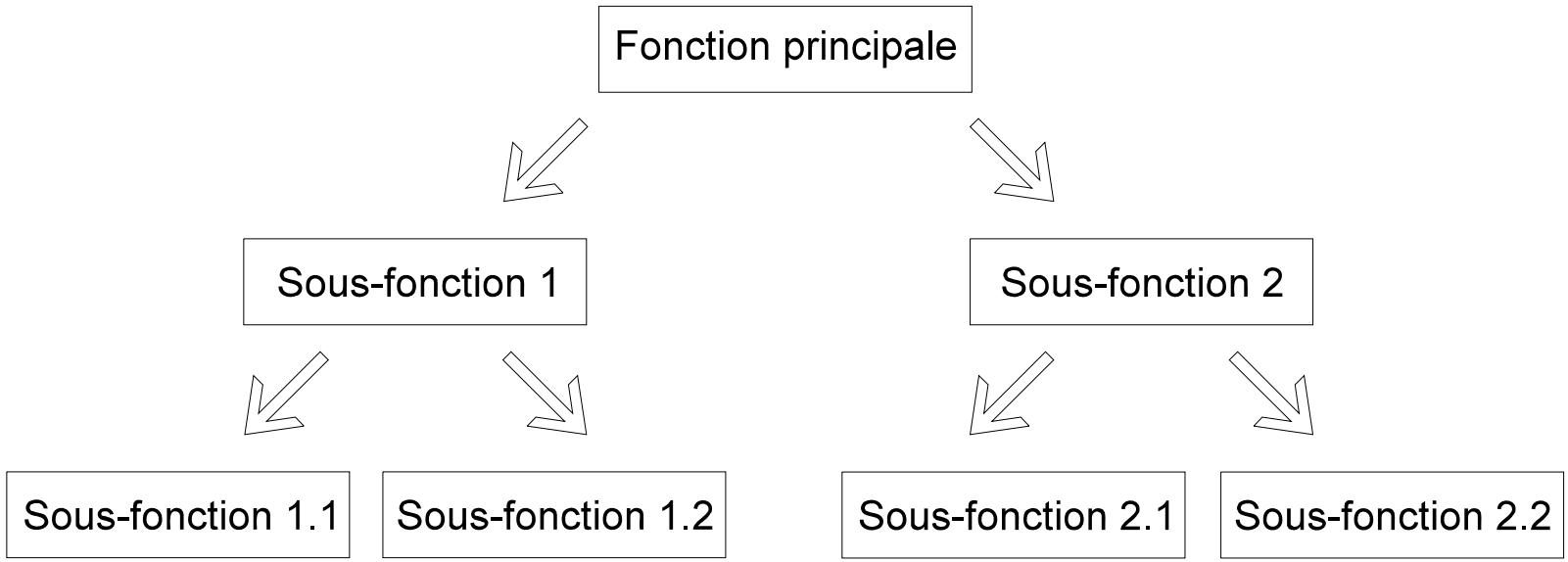
- Diviser pour régner
- Raffinage récursif jusqu'à obtention d'éléments compréhensibles
- Division intelligente de l'espace des états d'un système

Décomposition algorithmique

- Approche traditionnelle
- Chaque module représente une étape du processus global
- Décomposition fonctionnelle depuis le cahier des charges jusqu'au sous-programme

Décomposition algorithmique

- La fonction donne la forme du système

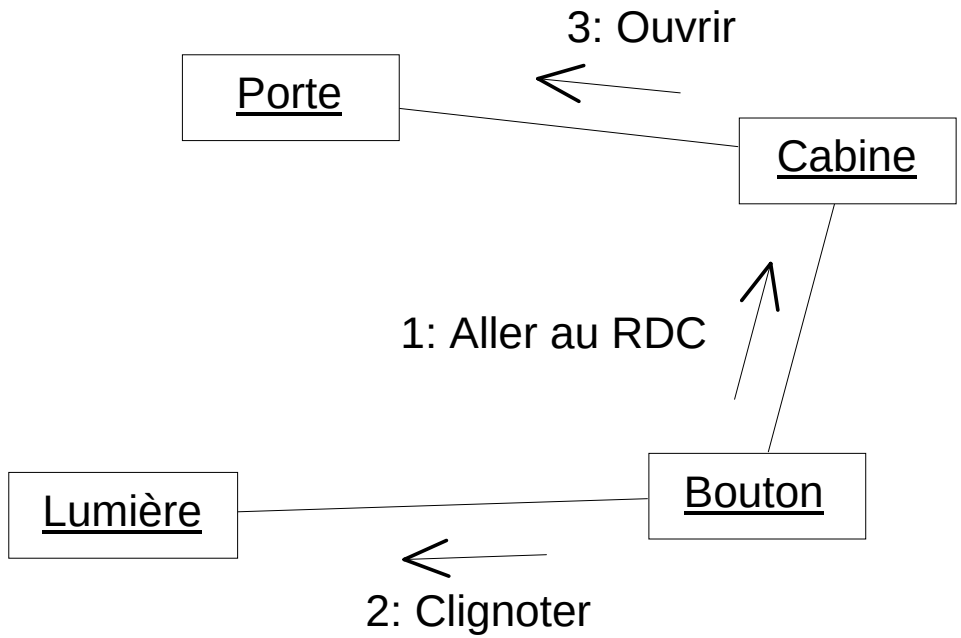


Décomposition objet

- Approche plus récente (en informatique)
- Chaque module représente un objet du domaine de l'application
- Les objets sont des entités autonomes qui collaborent afin de réaliser un projet global

Décomposition objet

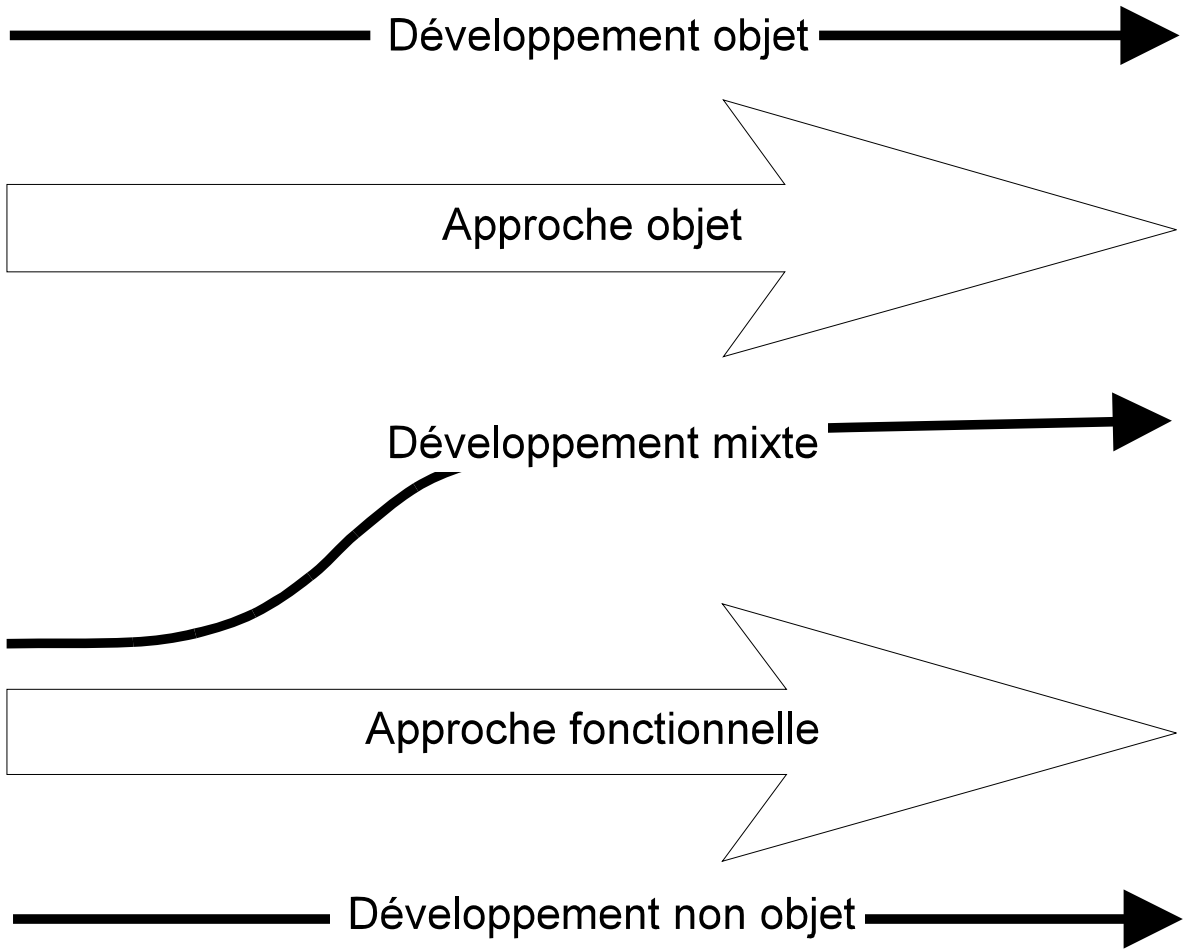
- La fonction est réalisée par des objets collaborants



Décomposition / Composition

- Le terme décomposition objet est réducteur
- L’approche objet n’est pas seulement descendante
- Approche descendante, ascendante, récursive, itérative, incrémentale...

Evolution des méthodes



L’âge de raison

- L’approche orientée-objets à 30 ans
  - 1966 Une idée à Oslo
  - 1969 La recherche au PARC (Xerox)
  - 1980 Version industrielle de Smalltalk
  - 1986 1200 personnes pour OOPSLA’86
  - 1996 Omniprésence des solutions objets
  - 2006 Stabilisation de la transition vers l’objet

Comparaison fonctions / objets

- Les deux techniques de décomposition sont intéressantes
- Elles sont très différentes
- Il faut en choisir une pour commencer à décomposer
- Puis se servir du résultat comme cadre pour exprimer l’autre point de vue

Approche fonctionnelle

- Semble intuitive
- Mise en avant du «FAIRE»
- Bien adaptée lorsque tout est connu
- MAIS
  - Architecture rigide
  - Extension difficile
  - Peu adaptée à la découverte

Approche objet

- Mise en avant de l'«ETRE»
- Simple (petit nombre de concepts de base)
- Raisonnement par abstraction (objets du domaine)
- Adapté pour l'exploration et l'évolution
- MAIS
  - Déroutant pour les habitués de la démarche fonctionnelle

Avantages de l’objet

- Conduit à des modèles plus stables
  - Basés sur le monde réel
- Structure indépendante des fonctions
  - Evolutivité
- Encapsule la complexité
  - Facilite la réutilisation

En résumé

- Le logiciel est complexe par nature
- Il faut gérer cette complexité
- Les systèmes peuvent être décomposés selon ce qu'ils font ou selon ce qu'ils sont
- L'approche objet gère plus efficacement la complexité (que l'approche fonctionnelle)
  - Réutilisabilité, Evolutivité, Stabilité

Modélisation objet

- *Problématique*
  - *Gestion de la complexité*
  - *Approches de modélisation*
- **Unification des méthodes objet**
  - Genèse d'UML
- Concepts de base
  - Notation UML

Unification des méthodes objet

- Appel aux propositions de l’OMG
- Démarche d’unification
- UML (*Unified Modeling Language*)

De quoi a-t-on besoin ?

- Un langage de modélisation
  - Notation claire
  - Sémantique précise
- Une démarche de génie logiciel

Méthode = Langage + Démarche

Langage de modélisation

- Générique
- Expressif
- Flexible (configurable, extensible)
- Syntaxe et sémantique
- Unification par convergence aujourd’hui

Démarche

- Générique
- Impossible à standardiser
  - Personnes, applications, cultures...
- Cadre configurable
- Unification par convergence dans le futur

La prolifération des méthodes objet

- Une cinquantaine de méthodes objet dans les cinq dernières années
  - Confusion, attentisme
- Consensus autour d'idées communes
  - Objets, classes, associations, sous-systèmes, cas d'utilisation

Rapprochement de Booch et OMT

- Booch’93 et OMT-2 sont plus ressemblantes que différentes
  - Booch’93 adopte les associations, les diagrammes d'Harel, les traces d’événements
  - OMT-2 introduit les flots de messages et retire les diagrammes de flot de données
- Booch-93 construction
- OMT-2 analyse et abstraction

L’unification des méthodes

- La guerre des méthodes ne fait plus avancer la technologie des objets
- Recherche d’un langage commun unique
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation

Différentes sortes de systèmes

- Logiciels
  - Ingénierie des logiciels
- Logiciels et matériels
  - Ingénierie des systèmes
- Personnes
  - Ingénierie des affaires

Unification sur plusieurs domaines d'applications

La notation unifiée UML

- Basée sur les méthodes de BOOCH, OMT et OOSE
- Influencée par les bonnes idées des autres méthodes
- Mûrie par le travail en commun

Principales influences

- Souvent une histoire imbriquée

Booch	Catégories et sous-systèmes
Embley	Classes singletons et objets composites
Fusion	Description des opérations, numérotation des messages
Gamma, et al.	*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-Brock	Responsabilités (CRC)

Portée d'UML

- Standardiser les artefacts du développement
  - Modèles, notation et diagrammes
- Ne pas standardiser le processus
  - Dirigé par les cas d'utilisation
  - Centré sur l'architecture
  - Itératif et incrémental

Les objectifs

- Représenter des systèmes entiers
- Etablir un couplage explicite entre les concepts et les artefacts exécutables
- Prendre en compte les facteurs d’échelle
- Créer un langage de modélisation utilisable à la fois par les humains et les machines

Approche retenue

- Identifier la sémantique des concepts de base
- Classer les concepts
- Construire un métamodèle
- Choisir une notation graphique
- Regrouper par niveau d’abstraction, complexité et domaine

Métamodèle

- Identification des concepts fondamentaux
  - Définition de la sémantique de ces concepts
  - Choix d'une représentation graphique
- Métamodélisation d'UML avec UML
  - Description formelle des éléments de modélisation
- Austère, pas pédagogique
  - Méthodologistes
  - Constructeurs d'outils

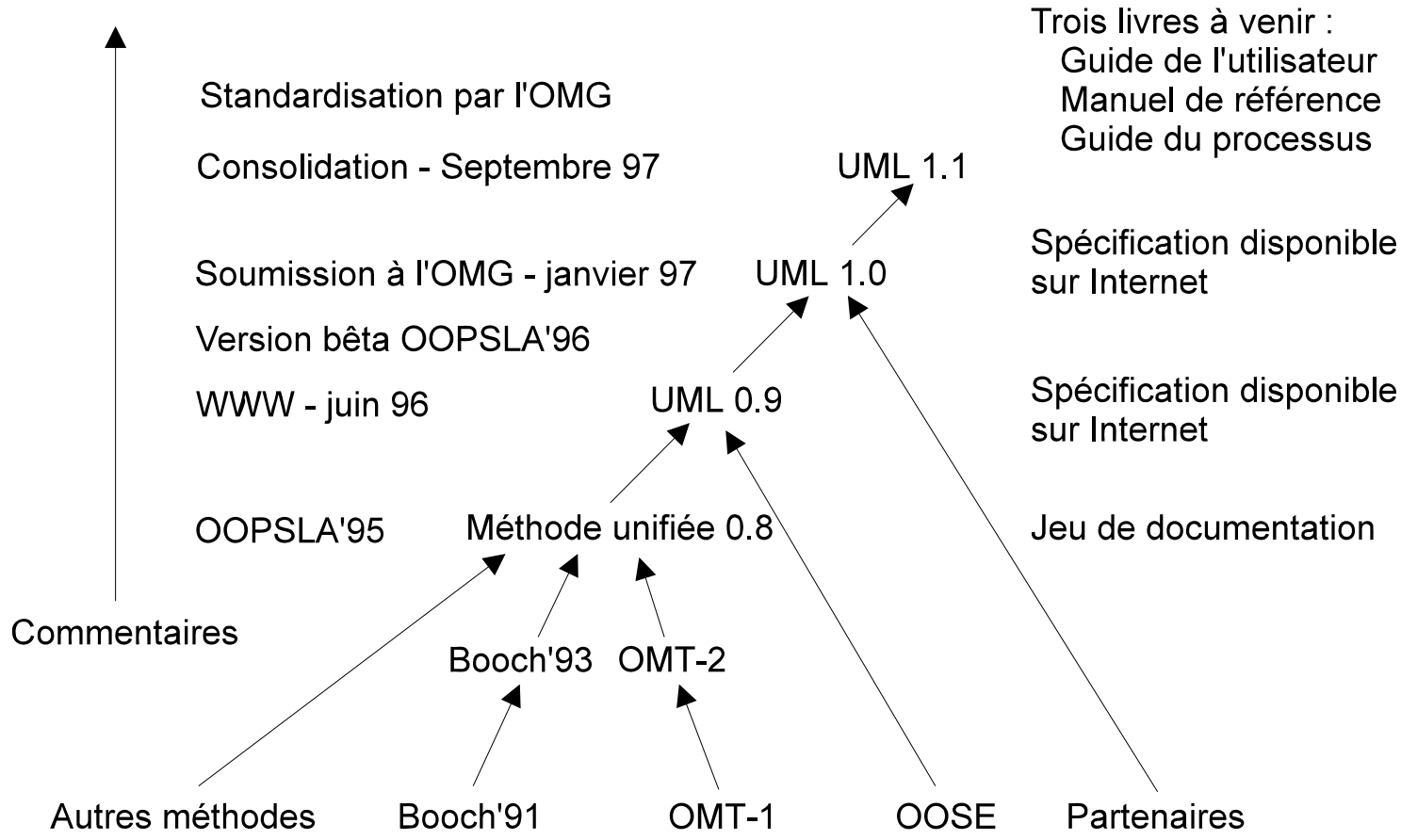
Les modèles et les vues

- Un modèle est un quanta de développement
  - Cohérence interne forte
  - Couplage faible avec les autres modèles
  - Relié à une phase de développement
- Une vue est une projection au travers des éléments de modélisation
  - Graphique
  - Peut englober plusieurs modèles

Les grandes étapes

- Octobre 95
  - *Unified Method* V0.8
- Octobre 96
  - UML V0.91 (*The Unified Modeling Language for Object-Oriented Development*)
- Janvier 97
  - UML 1.0 est soumise à l'OMG
- Décembre 97
  - UML 1.1 est normalisée par l'OMG

Evolution d'UML



Acceptation d'UML

- UML est dans le domaine public
- Successeur naturel des méthodes de Booch, OMT et OOSE
- UML est le fruit de l'expérience et des besoins de la communauté des utilisateurs

Les partenaires

- Courant 96 UML est devenu un enjeu stratégique
- Consortium de partenaires
  - HP, i-Logix, Intellicorp, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI et Unisys
  - IBM, ObjectTime, Platinum, Ptech, Reich Technologies, Softeam, Taskon

En résumé

- UML est une notation, pas une méthode
- UML est un langage de modélisation objet
- UML convient pour toutes les méthodes objet
- UML est dans le domaine public

UML est **la notation** pour documenter les modèles objets

Modélisation objet avec UML

- *Problématique*
  - *Gestion de la complexité*
  - *Approches de modélisation*
- *Unification des méthodes objet*
  - *Genèse d'UML*
- **Concepts de base**
  - Les objets

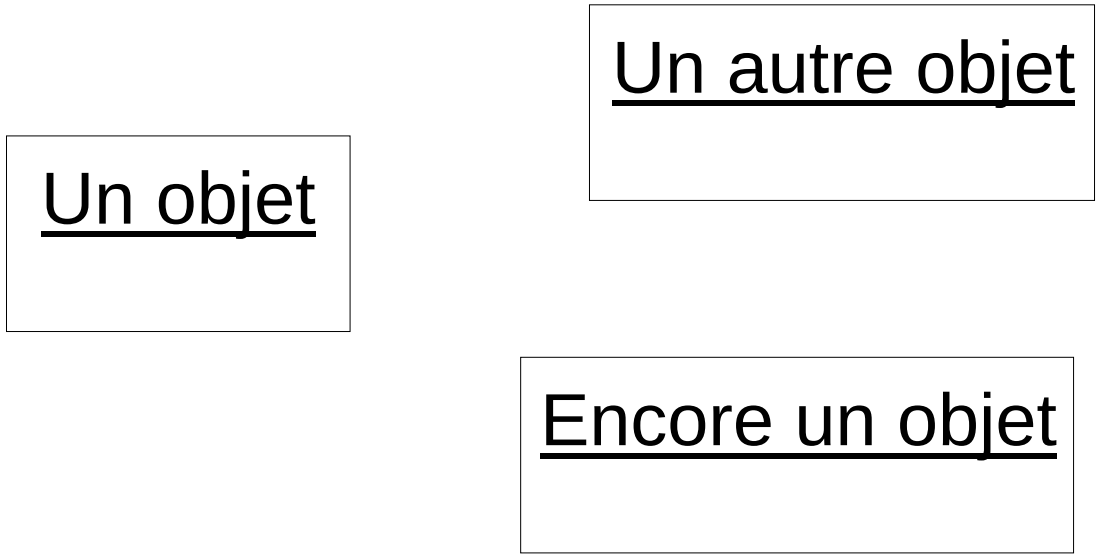
Les concepts de base

- **Les objets**

Les objets

- Les objets du monde réel nous entourent, ils naissent, vivent et meurent
- Les objets informatiques définissent une représentation simplifiée des entités du monde réel
- Les objets représentent des entités concrètes (avec une masse) ou abstraites (concept)

Représentation graphique des objets



Les objets sont des abstractions

- Une abstraction est un résumé, un condensé
- Mise en avant des caractéristiques essentielles
- Dissimulation des détails
- Une abstraction se définit par rapport à un point de vue

Exemples d’abstractions

- Une carte routière
- Un nombre complexe
- Un téléviseur
- Une transaction bancaire
- Une porte logique
- Une pile

Caractéristiques fondamentales des objets

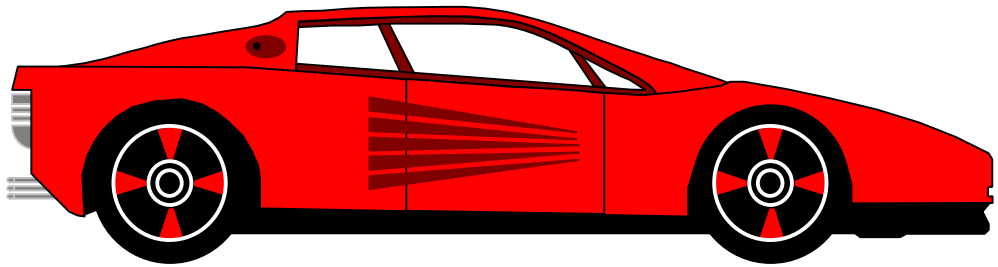
- L’état
- Le comportement
- L’identité

L’état

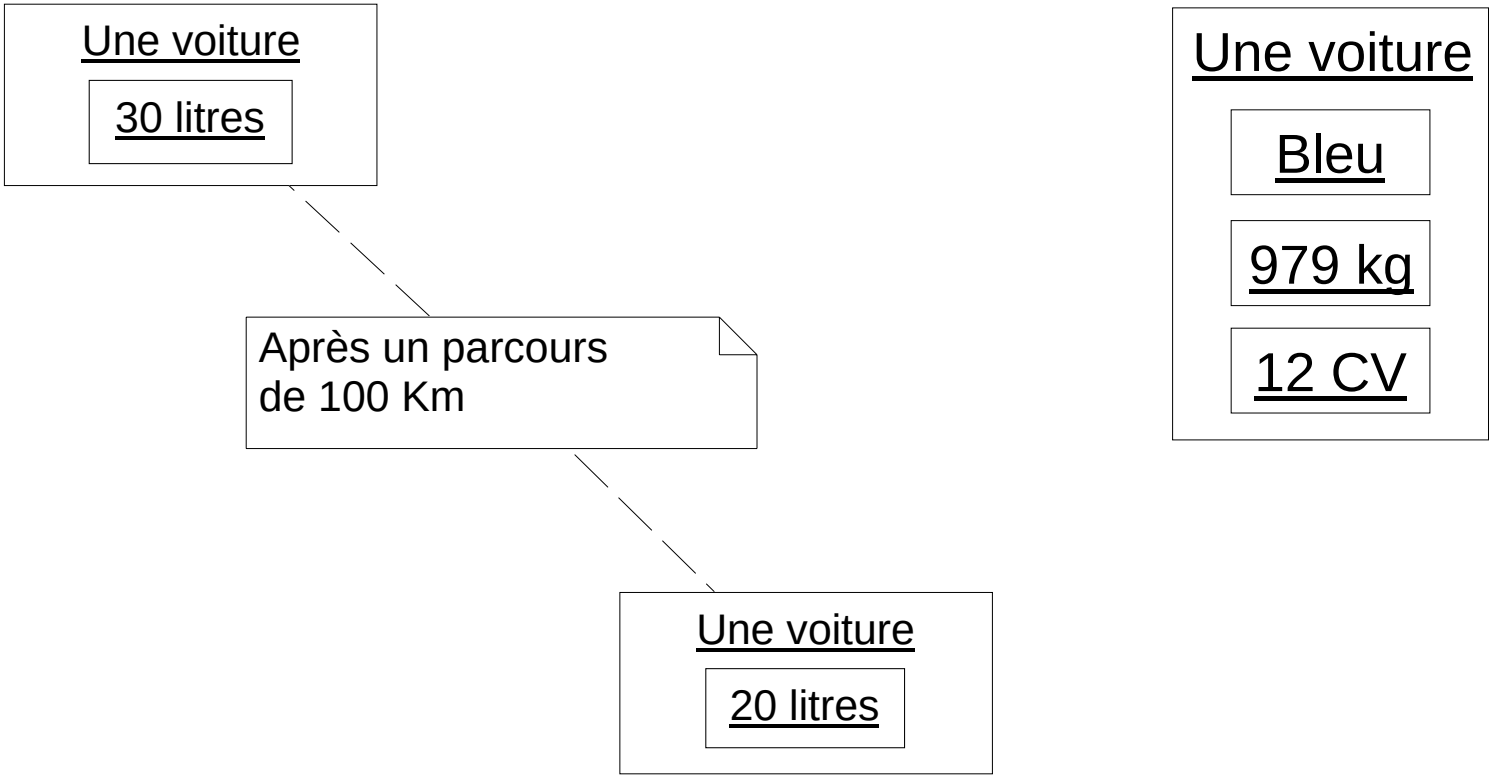
- L’état regroupe les valeurs instantanées de tous les attributs d’un objet
- L’état évolue au cours du temps
- L’état d’un objet à un instant donné est la conséquence de ses comportements passés

Exemples

- Pour un signal électrique : l'amplitude, la pulsation, la phase, ...
- Pour une voiture : la marque, la puissance, la couleur, le nombre de places assises, ...



Exemple

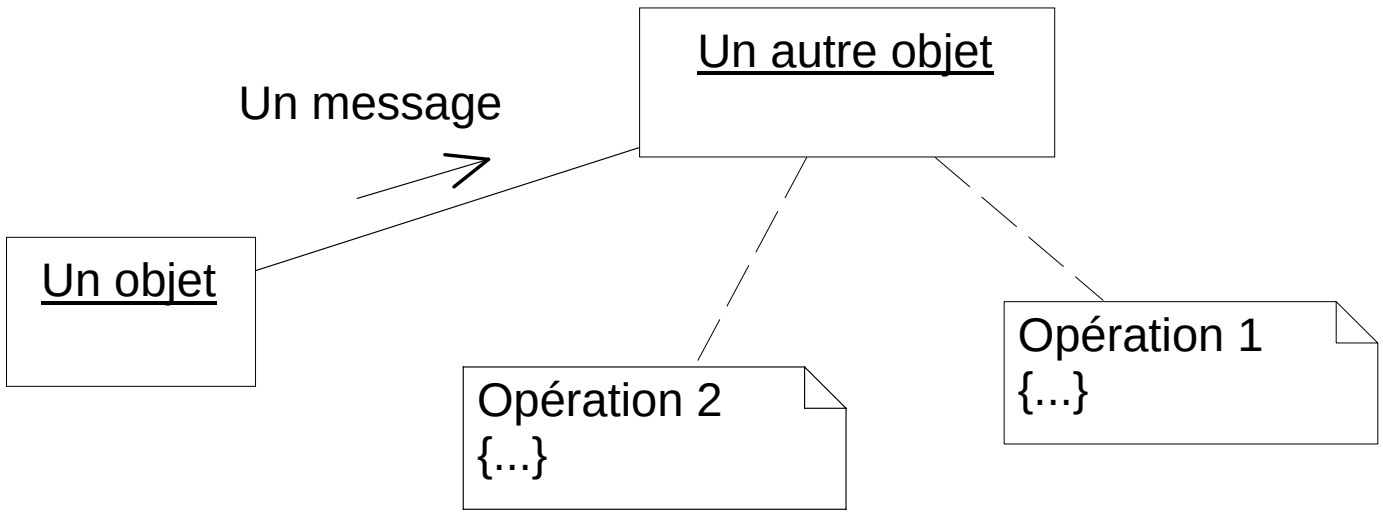

Une voiture
30 litres
Après un parcours
de 100 Km
Une voiture
20 litres
Une voiture
Bleu
979 kg
12 CV

Le comportement

- Le comportement décrit les actions et les réactions d’un objet
- Le comportement regroupe toutes les compétences d’un objet
- Le comportement se représente sous la forme d’opérations

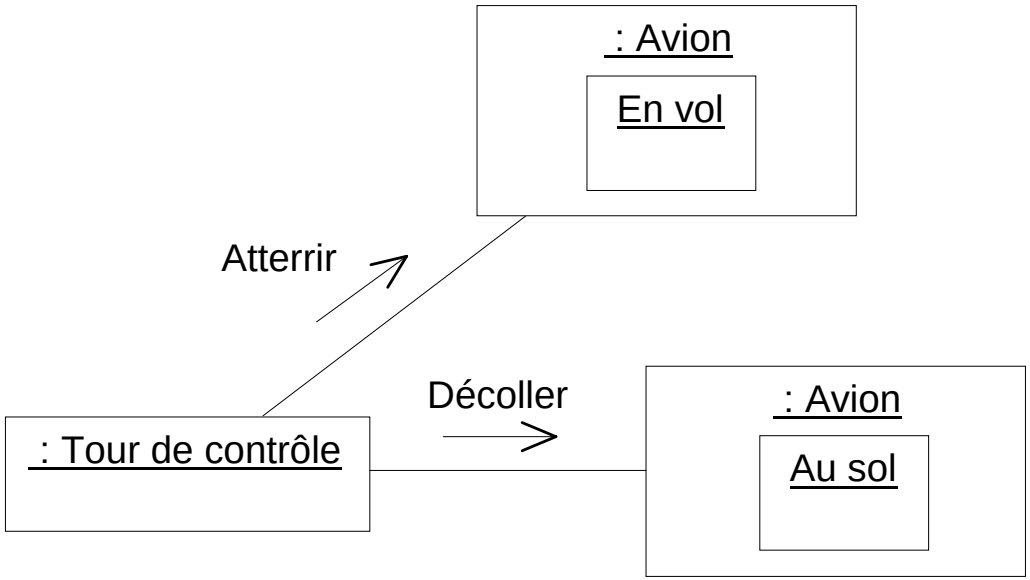
Comportement (suite)

- Un objet peut faire appel aux compétences d'un autre objet



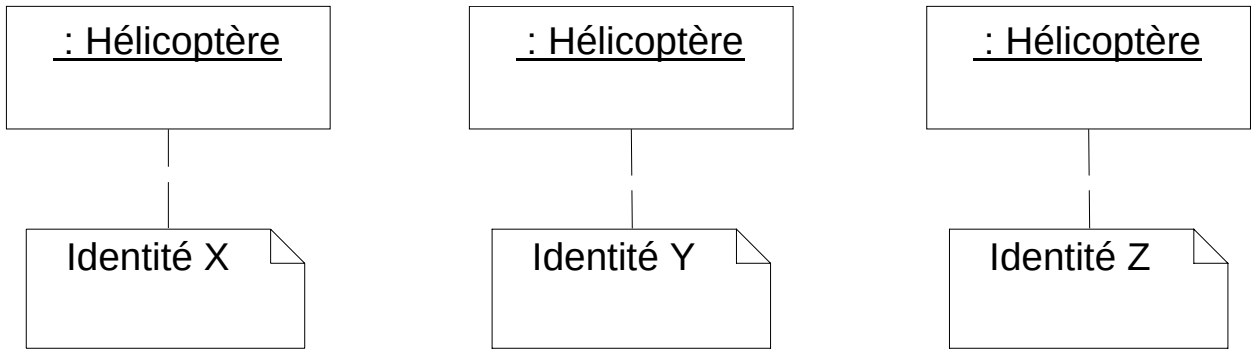
Comportement (suite)

- L’état et le comportement sont liés
  - Le comportement dépend de l’état
  - L’état est modifié par le comportement



L’identité

- Tout objet possède une identité qui lui est propre et qui le caractérise
- L’identité permet de distinguer tout objet de façon non ambigüe, indépendamment de l’état

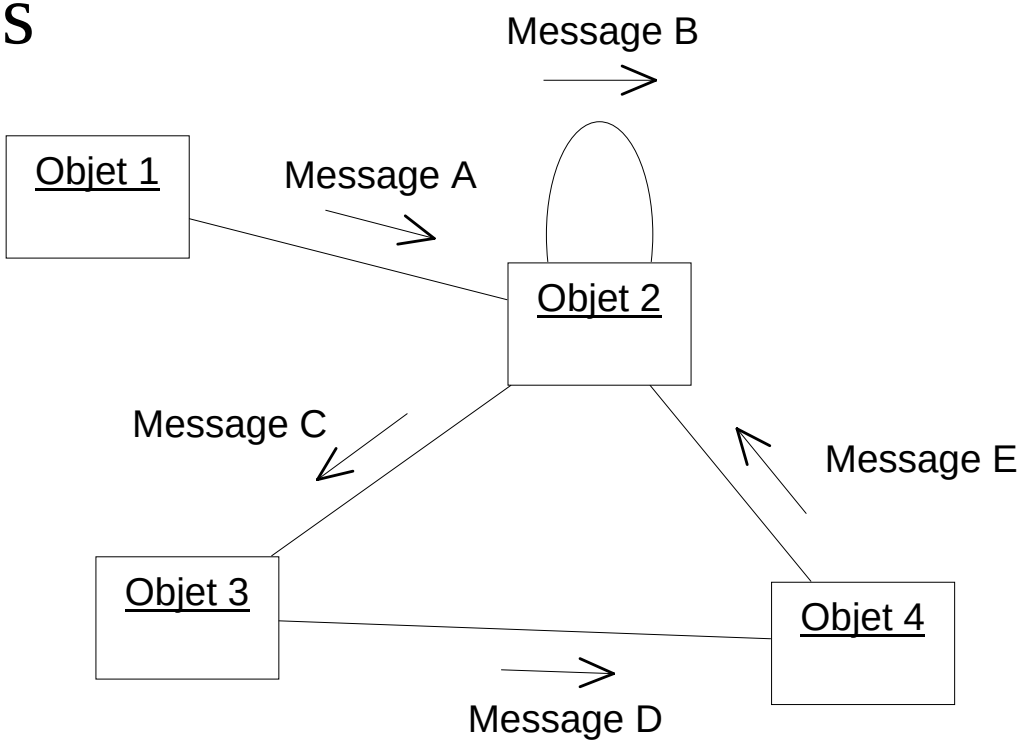


Identité (suite)

- Les langages objets utilisent généralement des pointeurs pour réaliser un identifiant
- Une clé primaire dans une base de donnée relationnelle est une manière de réaliser l'identité (en l'insérant dans l'état)
- Exemple : notre numéro de sécurité sociale

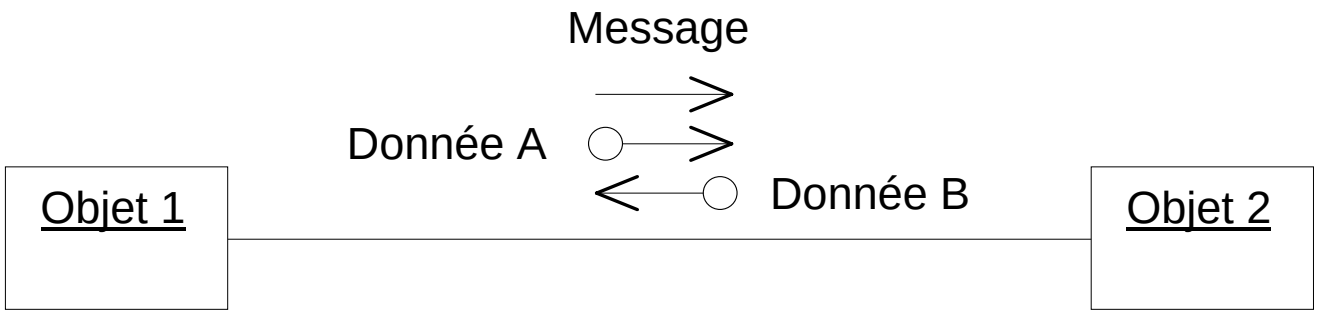
Les objets ne vivent pas en ermites

- Les objets interagissent les uns avec les autres



Communication entre objets

- Les objets communiquent en échangeant des messages



Conclusion

- L’objet est actuellement la démarche la plus performante pour gérer la complexité
- L’objet est adapté à toutes les phases du cycle de vie, tous les types de systèmes
- UML est la notation normalisée pour la documentation des modèles objets