Vers la modélisation orientée objet

Stéphane Ducasse
Stephane.Ducasse@univ-savoie.fr
http://www.iam.unibe.ch/~ducasse/

Stéphane Ducasse

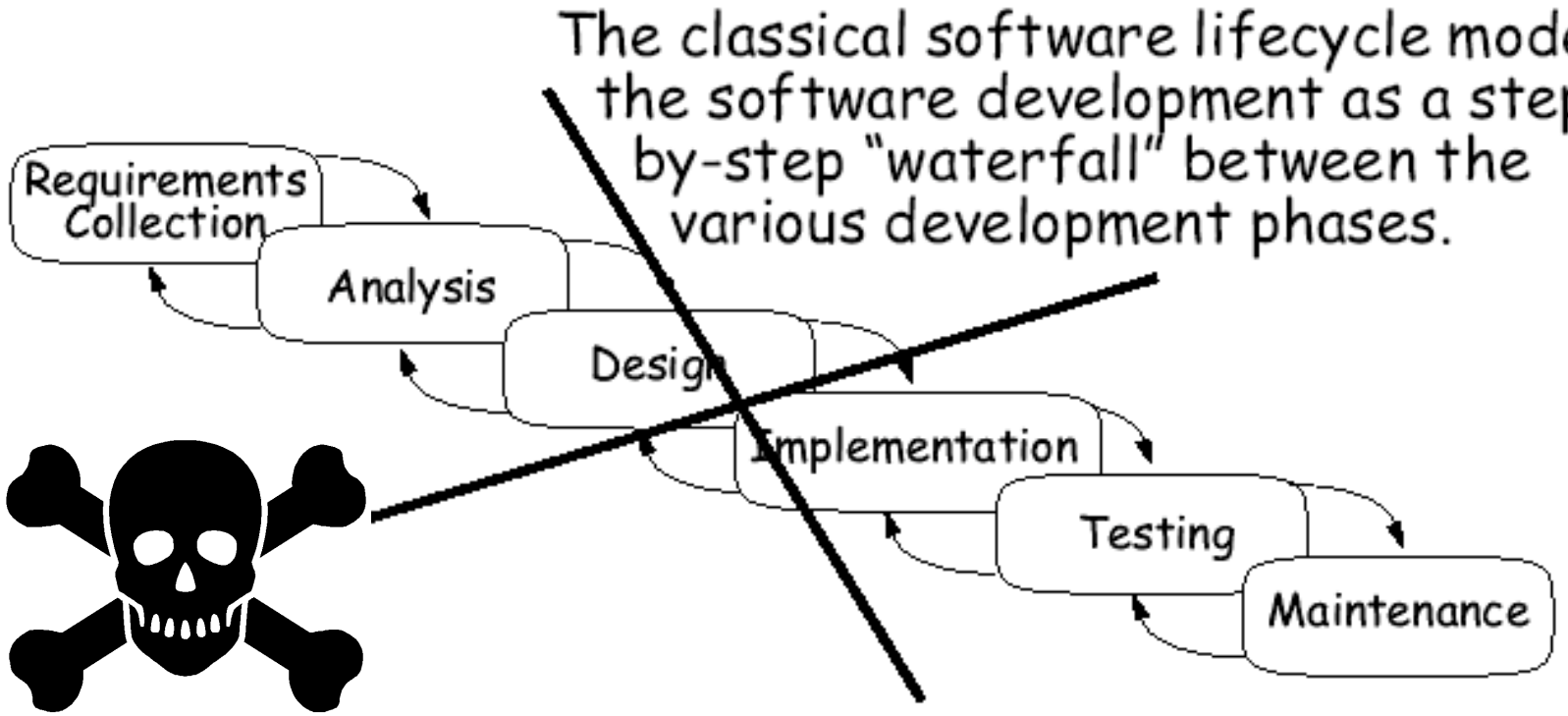
Plan

- Phases
- Méthodes
- Familles de langages

Différentes Phases

- Analyse: identification du cahier des charges
- Conception: identification des acteurs/responsibilités
- Implementation et tests: mise en exécution

Develop. en Cascade est mort



The waterfall model is unrealistic for many reasons, especially:

- requirements must be "frozen" too early in the life-cycle
- requirements are validated too late

Naifs et autres

Certains “experts” disent qu’une bonne conception permet de ne pas changer le code après...

Vrai pour des domaines figés dans le marbre pour les 10 prochaines années et complétement spécifiés

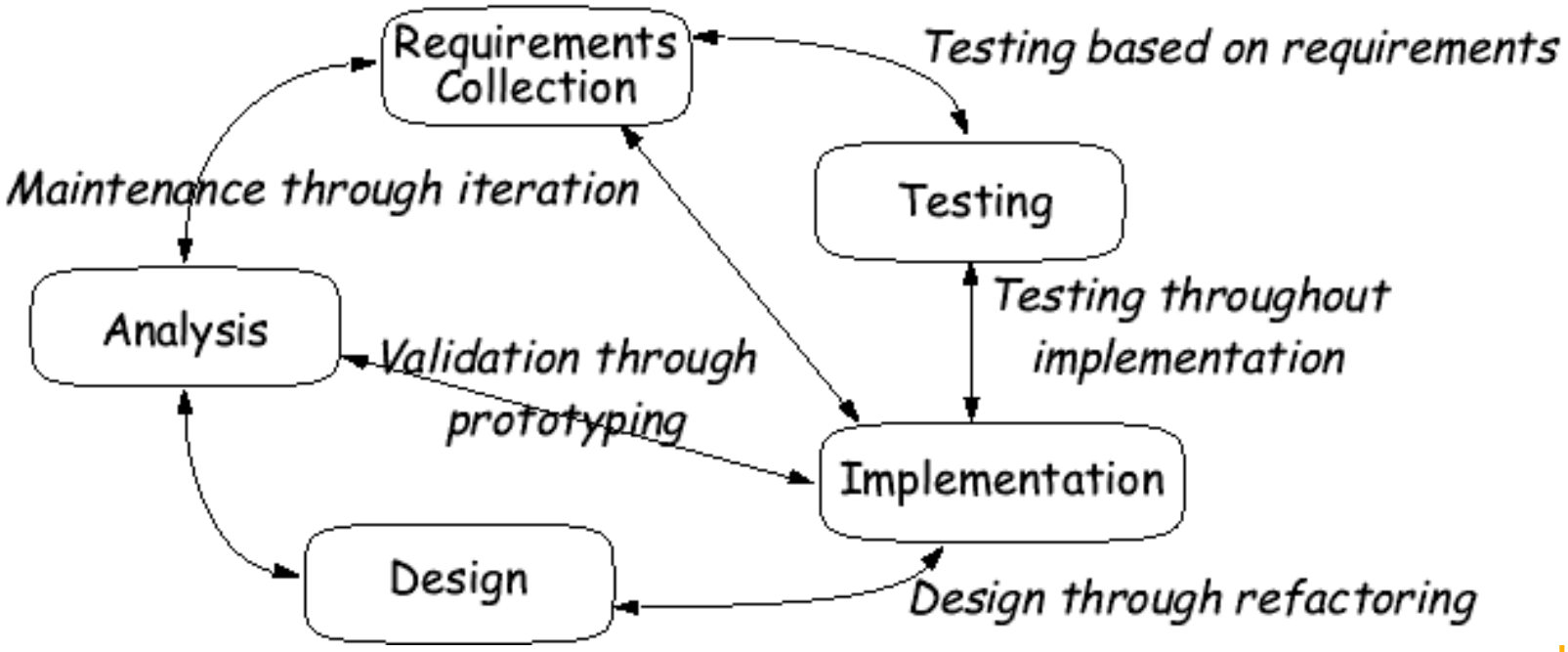
Donc la réalité est toute autre ...

Mais dans la réalité

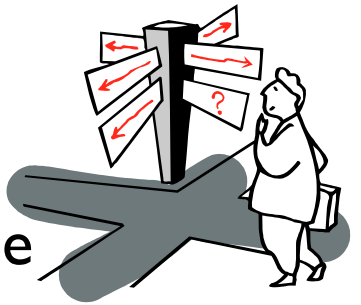
- Extrêmement difficile de faire une conception juste du premier coup
- Difficile de comprendre le domaine
- Difficile de comprendre ce que le client veut, qd il le sait !!!
- Difficile de savoir comment le système devra évoluer dans 5 ans
- La conception originale n'est souvent pas adaptée
- Faire une conception hyper flexible complexifie tout et sans garantie que cela serve

Développement iteratif

In practice, development is always iterative, and *all* software phases progress in parallel.



Evolution des méthodes



- 1970-1990 : orienté données et/ou fonctionelle
- 1980-90 début methodologies objets
- 1990-2000 maturité des methodologies objets
- nouvelles methodes:
  - XP
  - Unified process
  - Catalysis

Types de méthodes de Dev

- Une méthode détermine avec quelle approche la modélisation sera faite pour résoudre un problème.
- La façon de découper influence grandement le type de solution proposée.
- Trois types de découpage existent :
  - Découpage par les fonctions
  - Découpage par les données
  - Découpage par les objets.

Découpage par les fonctions

- Caractéristiques générales:
  - Modélisation orientée traitement, i.e., fait l'inventaire des fonctions à réaliser.
  - Toute description de données est subordonnée à celle d'un traitement.
  - Identification des fonctions globales puis décomposition de façon hiérarchique en sous-fonctions suffisamment petites.
  - Décomposition fonctionnelle : passage du général au particulier.

Pros

- Simplicité de la méthode
- Adéquation à capturer facilement les besoins des utilisateurs

Cons

- Néglige la cohérence et la redondance des données.
- Production de découpages ***différents*** selon les analystes.
- Les fonctionnalités du système sont très volatiles, donc toujours en développement.
- La décomposition fonctionnelle n'implique pas nécessairement la modularité des sous-fonctions obtenues, donc ***limite importante à la réutilisation***.
- Extension de nouveaux cas difficile

Découpage par les données

- Mode de découpage qui met l'accent sur les données, les fonctions venant en second lieu.
- Découper par les données [modèle entités-relations]:
  - Identifier les informations à traiter (entités)
  - Identifier les relations entre ces entités
  - Structurer ces informations (schéma de base de données)
  - Produire des fonctions de manipulation de ces données.

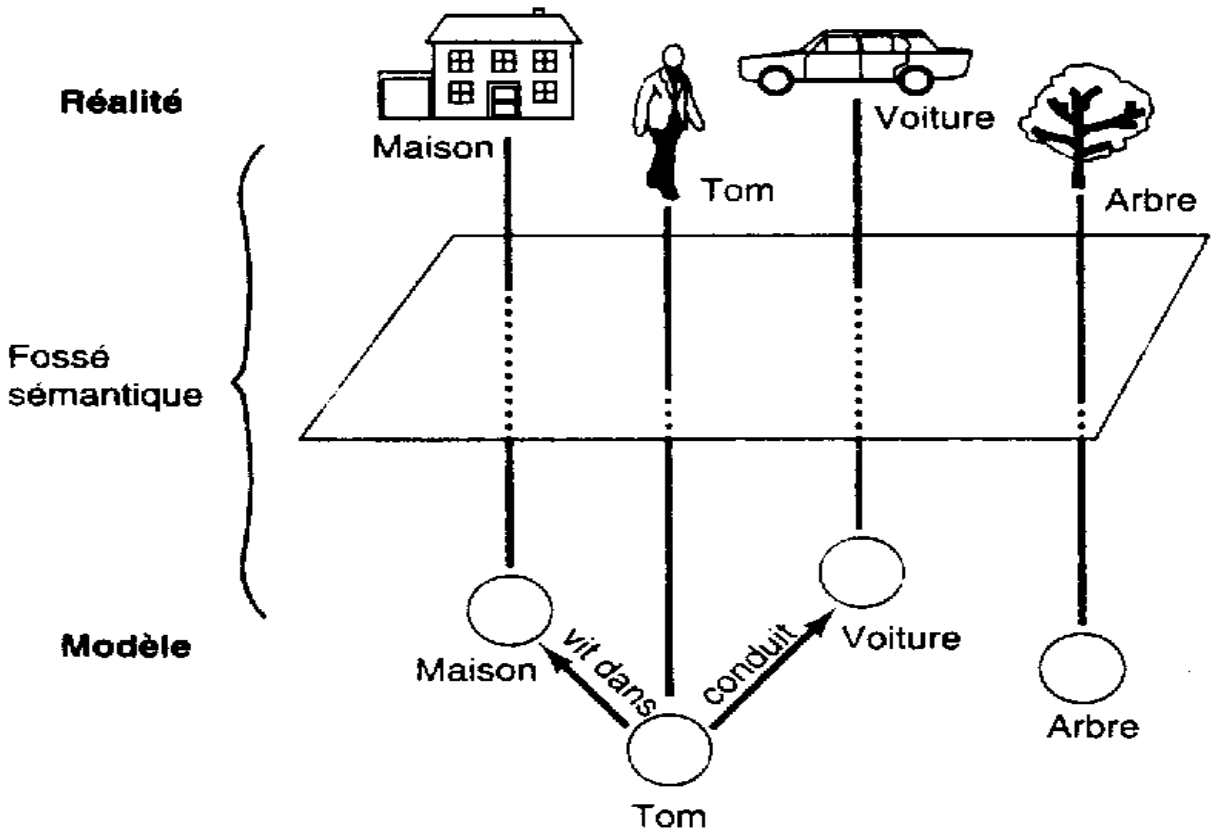
Pros

- Base le découpage sur quelque chose de plus stable (les données).
- Mise en place de système de gestion de base de données (SGBD), principalement relationnel.
- Mise en place d'outils de génération de formulaires permettant de manipuler les données (4GL)
- Adapté aux environnements de gestion (et encore)

Cons

- Les aspects fonctionnels indépendants des données trouvent mal leur place dans ce mode de pensée.
- Modéliser des processus est délicat
- Spécialisation des rôles : modélisateur de données et modélisateur de traitement. Peu de liens entre eux.
- Les validations et les contraintes d'intégrité sont souvent insérées dans les fonctions rendant ainsi difficile l'évolution.
- Un petit changement aux données peut faire boule de neige...
- Réutilisation difficile

Le fossé sémantique



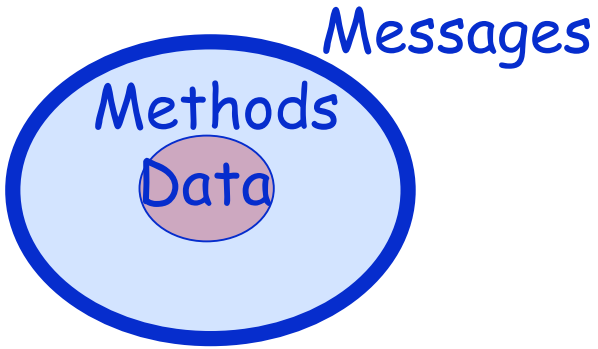
Découpage par objets

- La modélisation se fait en tenant compte autant des données que des fonctions.
- Lien fort entre données et fonctions.
- Objet responsable de ses données.
- Objet offre des services et interagit avec d'autres objets.
- Approche plus naturelle de découpage car ce qu'on modélise est proche de la réalité.
- L'approche orientée objet réduit le fossé sémantique.
- Meilleur de compréhension de la conception et du code.

Objet en trois mots

- Ensemble d'entités collaborant pour accouplir une tache
- Entités communiquent par envoi de message pour accéder aux services proposés

- Objet
  - a des responsabilités
  - offre des services
  - protége ses données



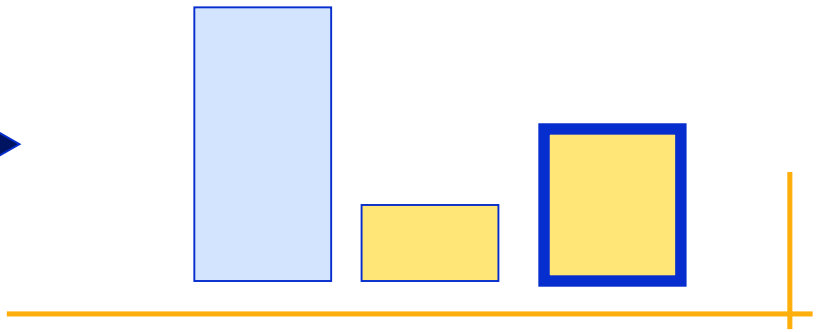
Un objet

- possède une identité
  - Nous sommes humain mais je ne suis pas vous!
- a des responsabilités: accomplir la tâche demandée
- offre des services (a un comportement)
- protége ses données
  - Ne laissez pas qq jouer avec votre numero de carte de crédit!
- peut déléguer à un autre objet

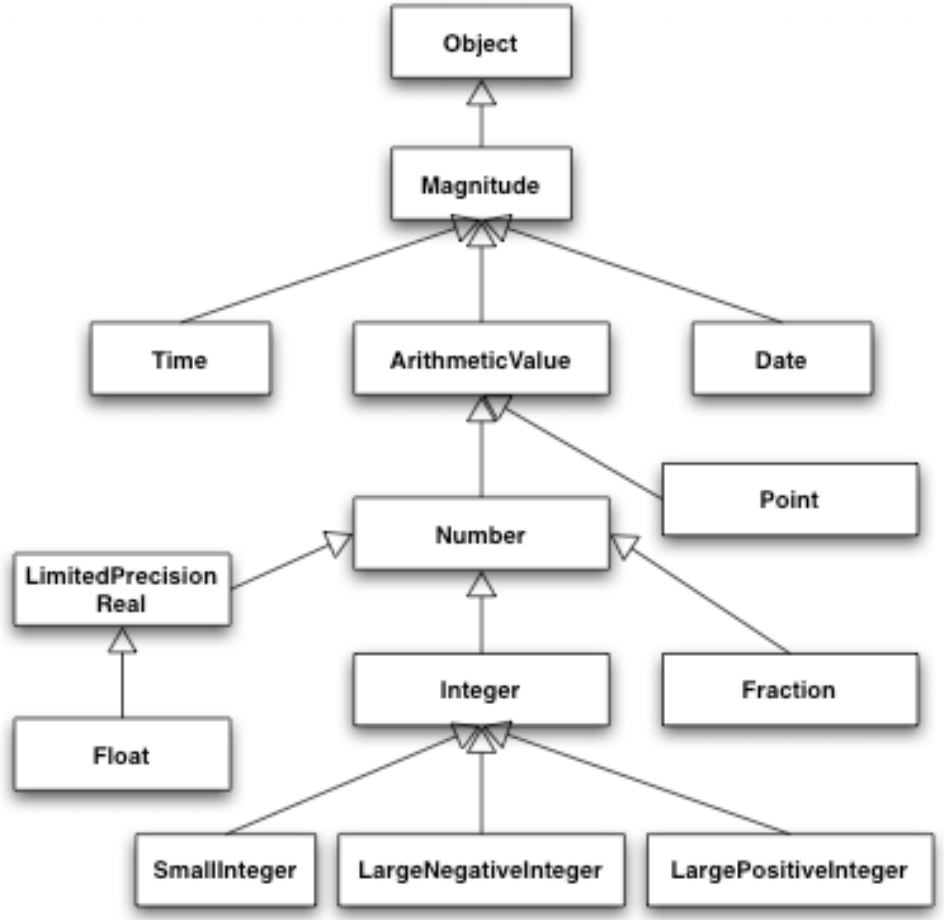
Classes: Usines à objet

- Classes sont des abstractions
- Classes sont des moules à objets
- Classes définissent le comportement des objets

Rectangle
length, width, origin, color, borderSize, borderColor
area intersect: center sideNearestTo:



Comparable Quantity Hierarchy



Cons

- Parfois plus lourd pour des applications triviales

Pros

- Plus proche du domaine modélisé
- Un seul paradigme pour differentes phases OOA, OOD et OOP
- Excellent pour des domaines et applications complexes
- Meilleure possibilité d'extension
- Réutilisation accrue

Plan

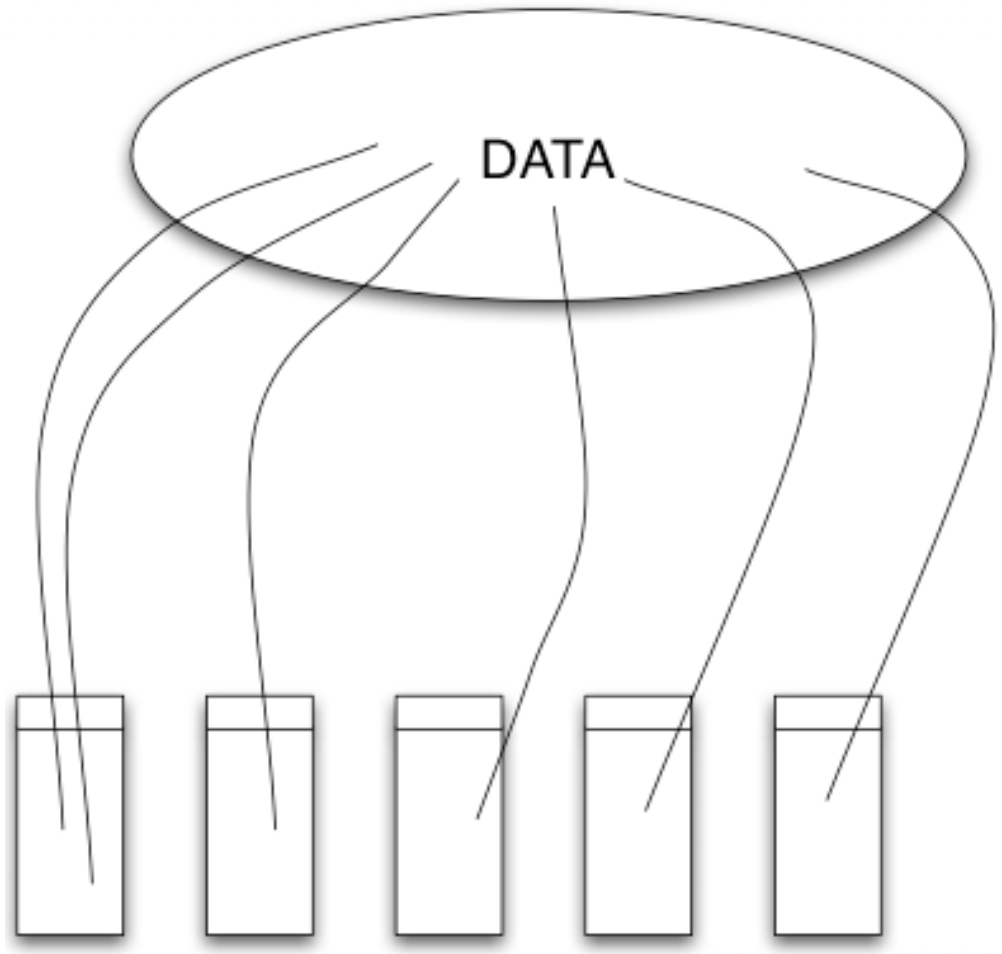
- Phases
- Méthodes
- Familles de langages

Un court résumé des

La programmation orientée objet est le résultat de l'évolution des mécanismes d'abstraction comme outils pour gérer la complexité :

programmation non structurée
programmation procédurale
programmation modulaire
programmation orientée objet

Programmation non structurée



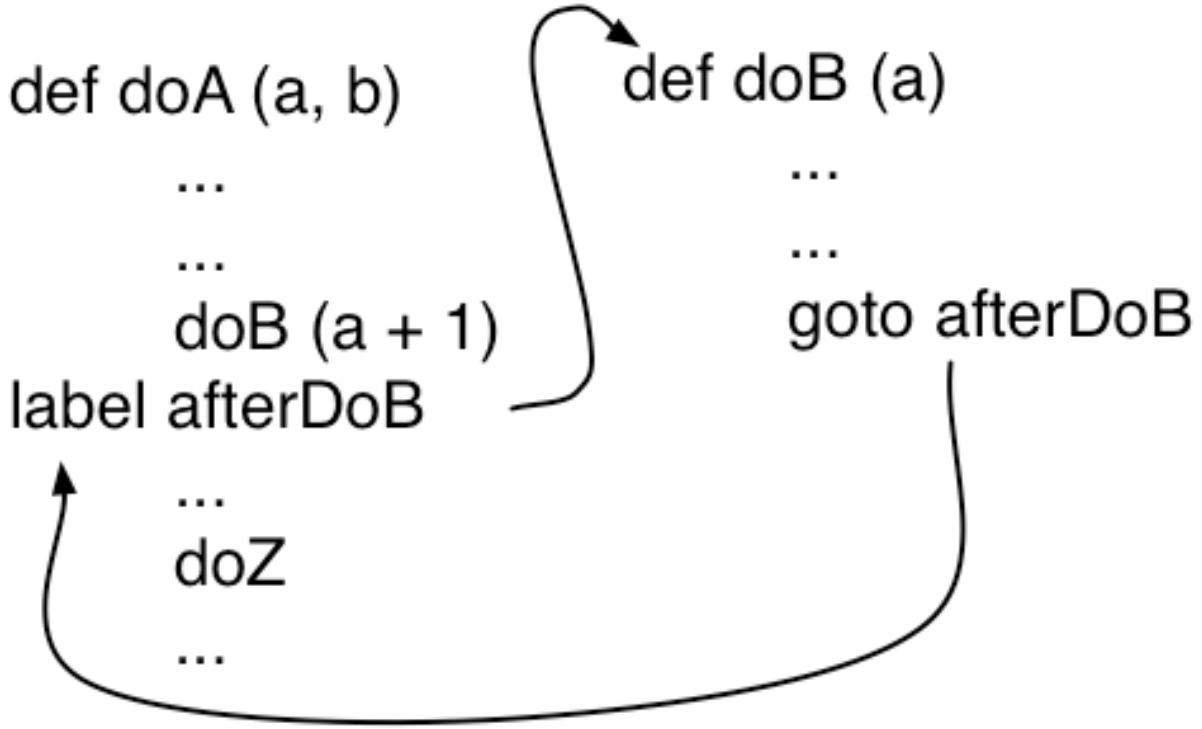
Programmation non-structurée

- Dans ce type de langage, on ***écrit*** des programmes qui effectuent des traitements sur des données de format connu.
- On produit des rapports, des bilans, des mises à jour, calcul scientifique...
- Peu de comportement
- Feuille et calcul de payes

Pas de retour d'appel

Réutilisation restreinte

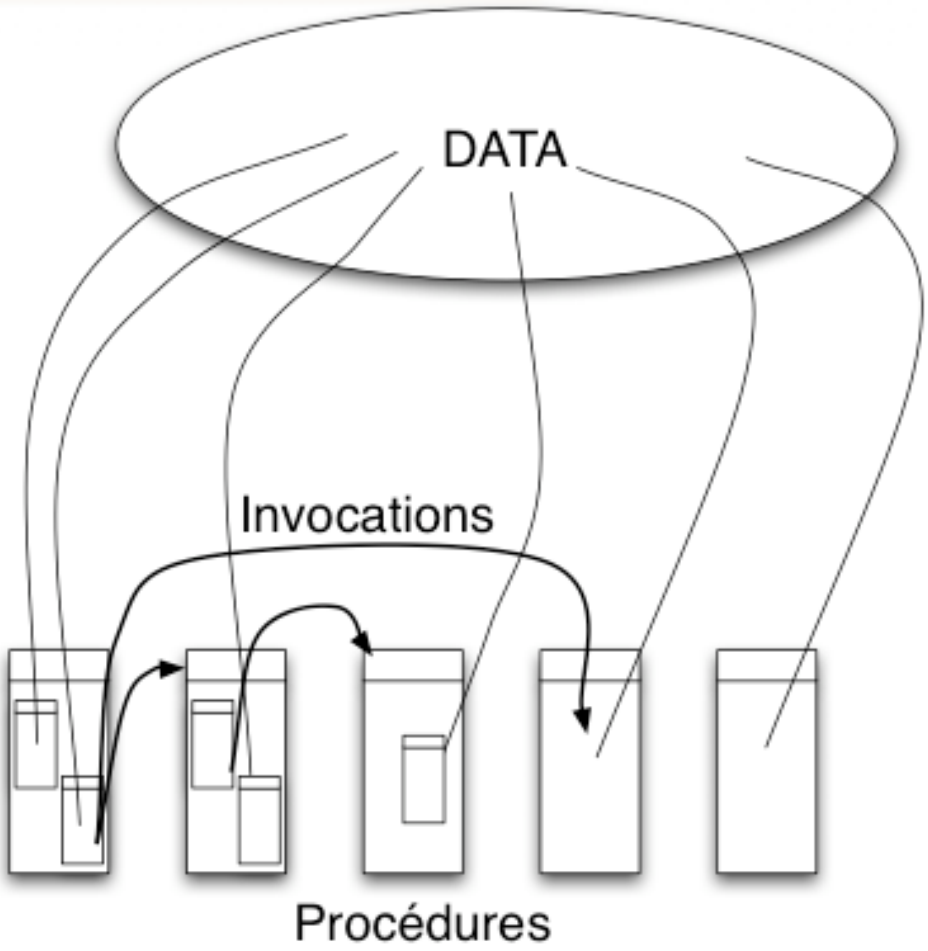
Spaghetti



Problèmes

- Données globales
- Couplage élevé des données
- Le programme est l'unité de base
- Partage de données à grande échelle peut rendre le code instable, extrêmemment difficile à maintenir
- Duplication des traitements communs
- Vieilles version de Cobol, Fortran

Programmation procédurale



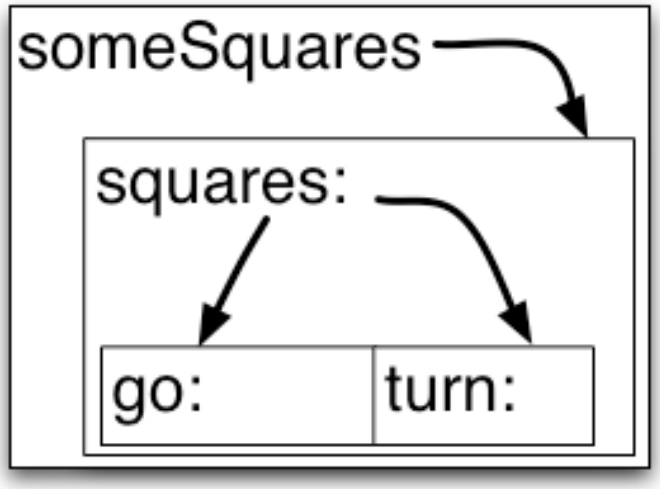
Programmation procédurale

- Il est possible de ***diviser*** la tâche à réaliser en plusieurs sous-tâches.
- On commence à réutiliser des fonctions dans plusieurs contextes.
- Les programmes commencent à être considérés comme un moyen d'***abstraire*** les problèmes et de les résoudre à l'aide de l'ordinateur.

Décomposition fonctionnelle

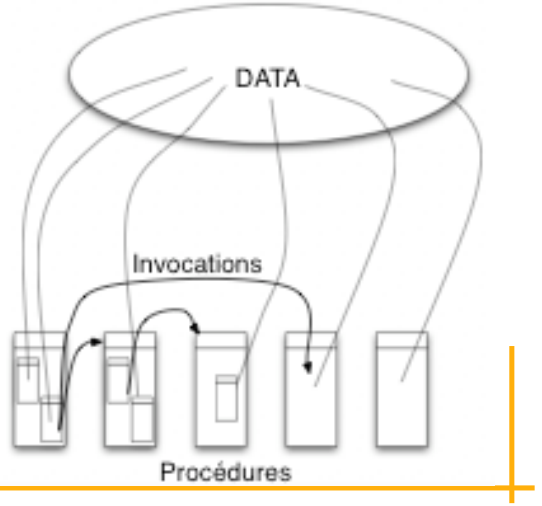
```
someSquares
   1 to: 10 do: [:i | self square: 10 * i]

square: size
    4 times:
        [ self go: size.
        self turn: 90 ]
```

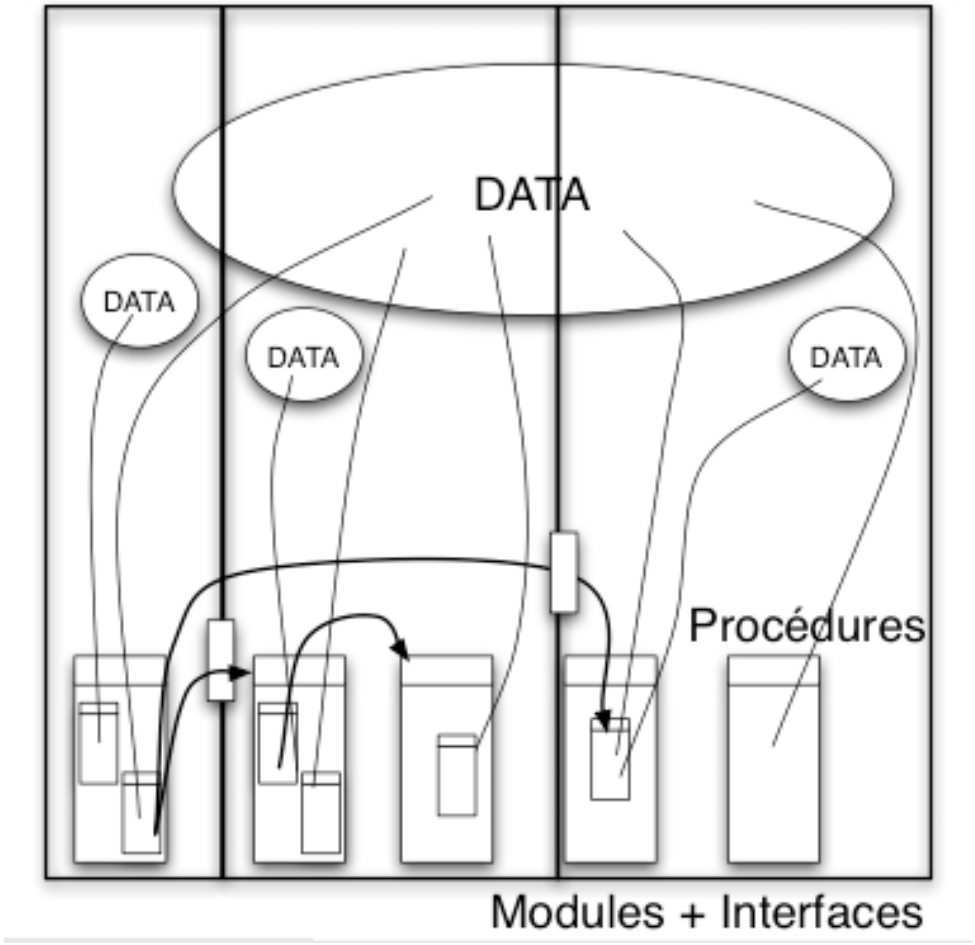


Evaluation

- Pros :
  - Apparition des procédures et des fonctions
  - Mécanisme de passage de paramètres
  - Développement des structures de contrôle
  - Émergence de méthode de conception structuré
  - Réutilisation des procédures plus facile
- Cons:
  - Données globales
  - Couplage élevé des données
  - Changement problématique
  - Extension délicate



Programmation Modulaire



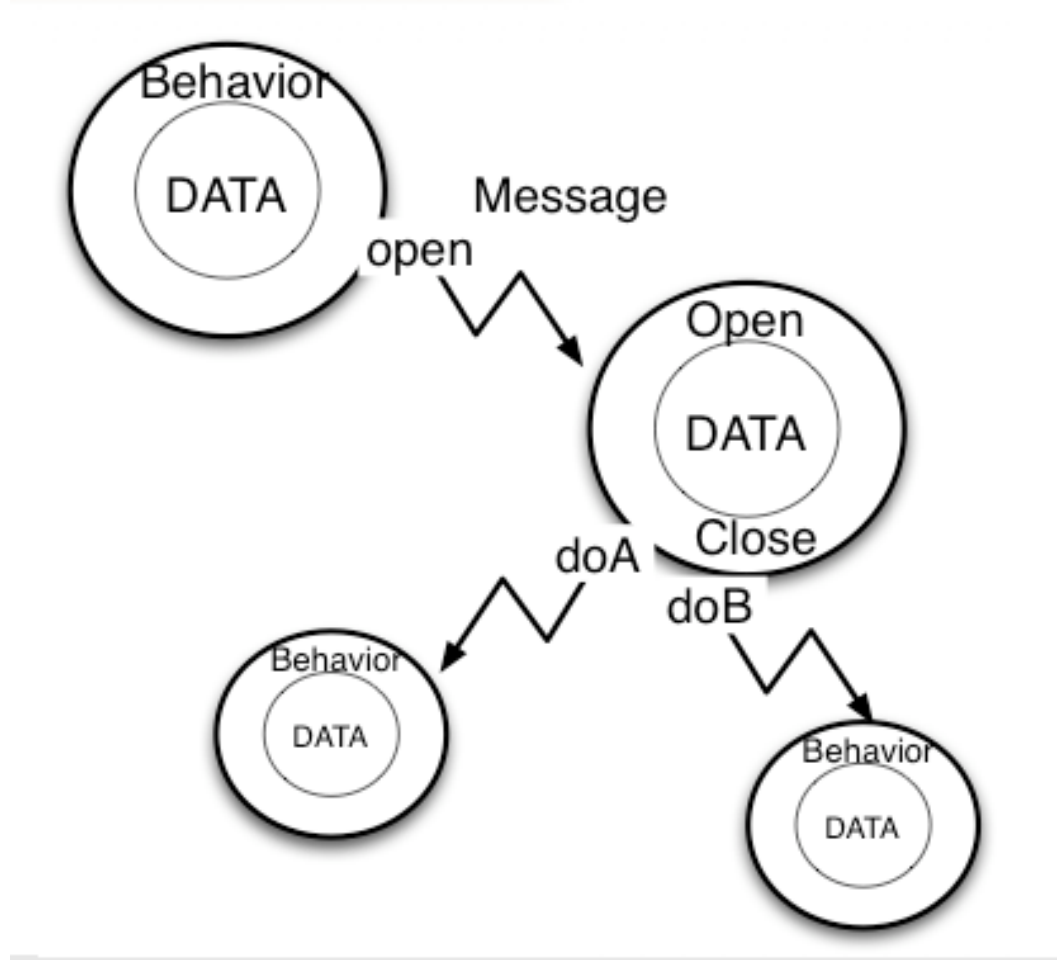
Programmation Modulaire

- Possibilité d'encapsuler des données à l'intérieur d'un module.
- Une portion ***publique*** au module, i.e., ce qui est accessible à l'***extérieur*** du module.
- Une portion ***privée*** au module, i.e., ce qui est accessible à l'***intérieur*** du module.

Evaluation

- Compilation séparée
- Compréhension plus facile:
  - Module
  - Puis interaction entre les modules
- Plus approprié pour la programmation dans des équipes de développement.
- Modula-2, C, Pascal
- Extension difficile
- Réutilisation délicate

Programmation à objets



Programmation à objets

- Un programme est un ensemble d'objets faiblement couplés qui communiquent par envoi de messages.
- Peu ou pas de variables globales
- Réduction du couplage entre données
- Composition de procédures => hiérarchie de classes et objets communiquant

Nouvelle façon de penser!

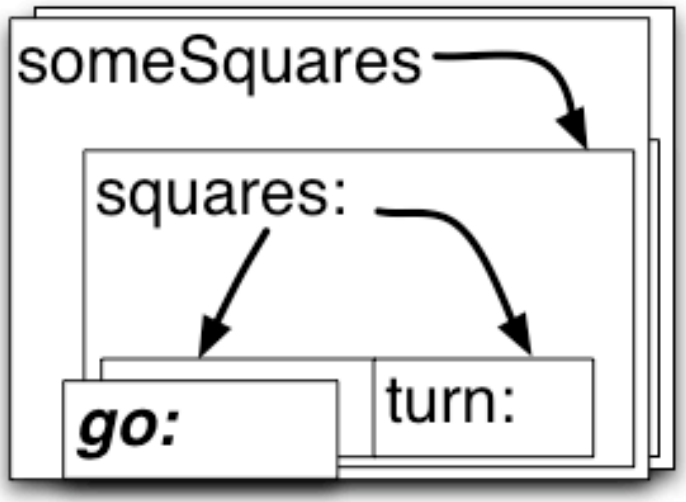
- Pas juste des mots clefs en plus...
- Nouvelle façon de décomposer les problèmes et de les résoudre
- Objets responsables de leur données
- Objets offrent des services
- Classes
  - définissent comportement et structure communs à un groupe d'objets similaires
  - créent des objets
  - étendent/réutilisent comportement d'autres classes

Composition with Extension

```
Turtle>>someSquares
   1 to: 10 do: [:i | self square: 10 * i]

Turtle>>square: size
    4 times:
        [ self go: size.
        self turn: 90 ]

Turtle>>go: dist
    location := location + dist
```




```
TurtleWithMemory>>go: dist
    self memorize.
    super go: aDistance
```

Points Clefs

Différentes méthodes de décomposition avec différentes propriétés (réutilisation, distance par rapport au domaine, fragilité en présence de changes)

Différents familles de langages de programmation

Modélisation objet

- Comportement et données groupés en une entité : l'objet
- objets responsable de ses données privées
- objets offrent des services :"communiquent" par envoi de messages
- objets interagissent pour accomplir une tâche