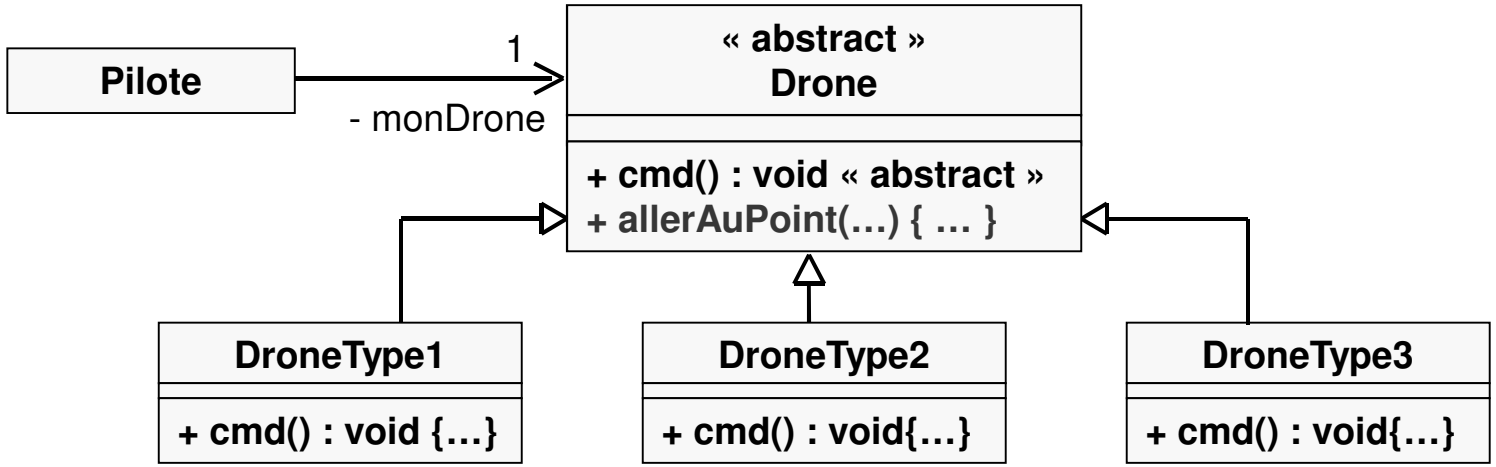
Un autre problème de conception…

- Une application à base de drones, dans un cadre de maintenance évolutive (*)…
  - Cf. doc. support



Concevoir (bien) est un « art » difficile !

- Même avec les technologies « objet »
- « Clé » de la qualité du produit
  - Satisfaire les exigences relatives au produit
    - Exigences fonctionnelles
    - Exigences extrafonctionnelles, par exemple
      - Performance, sécurité…
      - Maintenabilité, flexibilité, capacité d'évolution
        - Intelligibilité (structuration, documentation…)
      - Etc.
  - Satisfaire les exigences relatives au projet, par exemple
    - Productivité
      - Développer spécifiquement *vs* réutiliser des solutions existantes ?
      - Éventuellement développer pour réutiliser ultérieurement

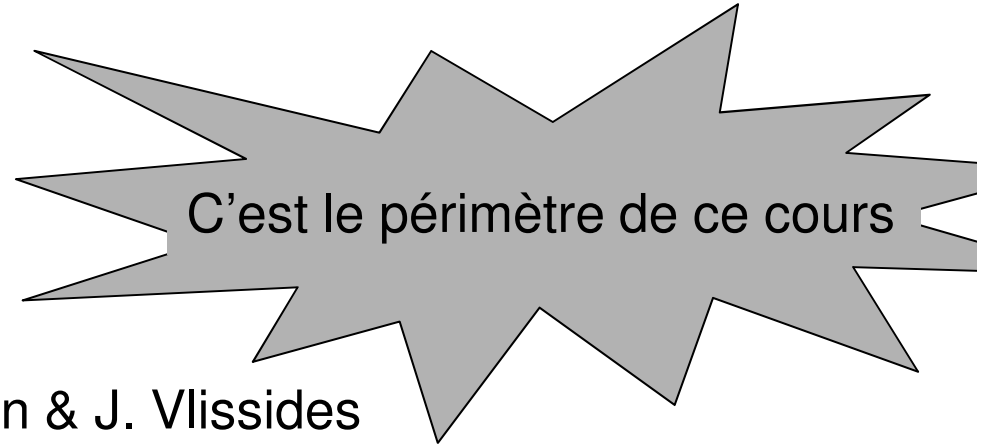
Concevoir (bien) est un « art » difficile !

- L'expérience et l'habileté du concepteur sont primordiales
  - Ne pas « réinventer la roue » !
    - Il existe des problèmes de conception récurrents (génériques) …
      - ☛ Savoir les identifier
    - … pour lesquels il y a des solutions éprouvées
      - ☛ Savoir les (ré)utiliser
  - Réutiliser l'expérience de conception (*vs* réutiliser le code)
    - Savoir-faire en matière de structuration et de composition
      - ☛ Solutions « architecturales » génériques
      - ☛ Modèles de conception
        - Définissent l'organisation et les relations entre classes
        - Sont réutilisables d'une application à une autre
  - Comment capitaliser l'expérience ?
    - Besoin de documentation

Patterns GRASP

- General Responsibility Assignment Software Patterns (GRASP)
  - Principes pour l'assignation de responsabilités aux classes et objets
    - Patterns de conception de base, assez simples et intuitifs
    - « Bonnes pratiques » de conception
  - Exemples : expert en information, faible couplage, forte cohésion, création, délégation, contrôleur, polymorphisme...
  - *Applying UML and Patterns*, Craig Larman, Prentice Hall, 2004

Design patterns du GoF



- GoF = « Gang of Four »
  - E. Gamma, R. Helm, R. Johnson & J. Vlissides
  - Ont proposé un catalogue de design patterns

- Organisation du code dans un cadre objet
  - Expression en termes de classes, d’interfaces, d’objets… et de relations entre ces éléments
  - Fondamentalement, le concept « objet » donne des mécanismes pour construire des logiciels flexibles, maintenables, évolutifs
    - Abstraction, encapsulation, héritage et délégation, polymorphisme…

Patterns d'architecture

- Structuration à grande échelle du système logiciel
  - Forme générique d'architecture en termes d'éléments architecturaux, de relations entre ces éléments, de propriétés et de contraintes
  - Exemples
    - Client-serveur, architectures n-tier…
    - MVC