Programmation Orienté Objet
– La modélisation objet –

Christophe BLANC

– BUT MMI1 –
IUT Clermont Auvergne
Université Clermont Auvergne

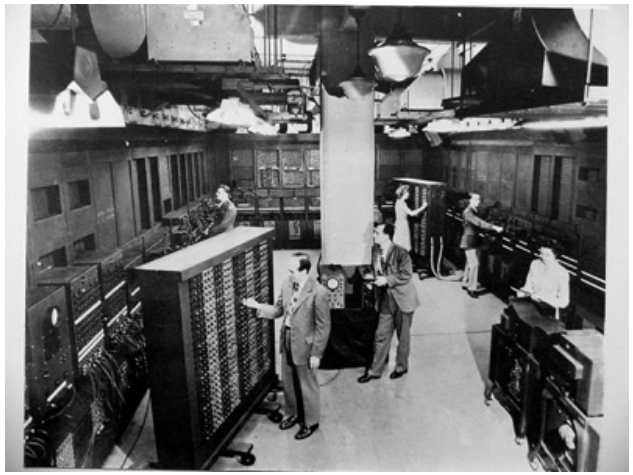
Février 2024

Plan

1. Rappels succinct sur l'évolution de l'informatique.
2. La complexité du logiciel.
3. La gestion progressive de la complexité.
4. Les limites de la programmation structurée.

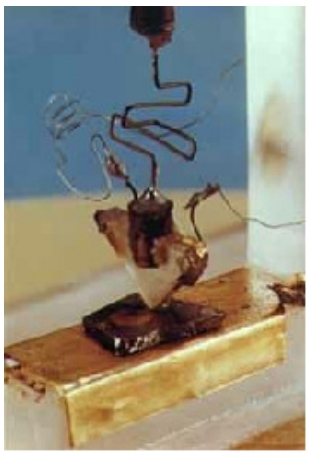
Quatre générations d'ordinateurs – Première génération

- La première génération utilisait la technologie des tubes à vide.
- Ces outils étaient tellement novateurs que seul un petit groupe de personnes concevait, construisait, programmait, utilisait et effectuait la maintenance des machines.



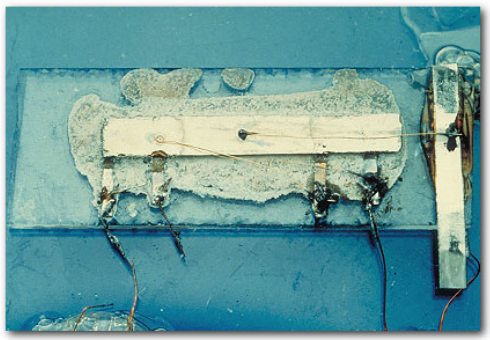
Quatre générations d'ordinateurs – Deuxième génération

- La deuxième génération est apparue avec l'invention du transistor.
- Les ordinateurs devinrent suffisamment fiables pour être construits et vendus à des clients. A ce moment là une séparation nette existait entre les concepteurs, les constructeurs, les opérateurs, les programmeurs et le personnel de maintenance.



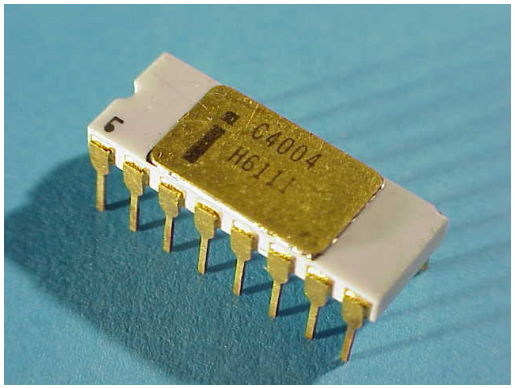
Quatre générations d'ordinateurs – Troisième génération

- La troisième génération est née de la création du circuit intégré.
- La pénétration de l'informatique dans la société (l'industrie, les universités) s'est accentuée grâce à l'apparition des mini-ordinateurs (vendus moins chers).



Quatre générations d'ordinateurs – Quatrième génération

- La quatrième génération a démarré avec la mise au point du microprocesseur.
- Le développement des micro-ordinateurs a alors permis de démocratiser l'accès à l'outil informatique.
- Le développement des gros ordinateurs a permis d'atteindre une puissance de calcul gigantesque.



L'utilisateur et son ordinateur - Vers la complexité logicielle

- Cette évolution a amené l'ordinateur dans les mains d'utilisateurs novices qui apprennent (parfois avec difficulté) à le manipuler comme un simple outil.
- Ces personnes, qui ne savent pas programmer cet ordinateur, doivent faire appel à des concepteurs de logiciels pour adapter l'ordinateur à leurs besoins (en faire un outil utilisable).

L'utilisateur et son ordinateur - Vers la complexité logicielle

- Ce dialogue s'effectue en boucle : un utilisateur exprime des besoins au concepteur qui crée un outil logiciel suscitant de nouveaux besoins...
- La multiplication des utilisateurs et des concepteurs a amené à la création d'outils logiciels de plus en plus complexes.

Ses origines

Selon Grady Booch, la complexité est une caractéristique inhérente au logiciel et elle provient de quatre éléments :

- la complexité des problèmes ;
- la difficulté à contrôler le processus de développement ;
- la flexibilité dans la programmation ;
- le passage du monde continu au monde discret .

La complexité des problèmes

- Les logiciels doivent parfois traiter des éléments complexes (par exemple, le pilote automatique d'un avion de ligne) auxquels viennent s'ajouter des exigences comme la facilité d'emploi, les performances ...
- Ces contraintes forment un ensemble de besoins que l'utilisateur a du mal à exprimer à un concepteur de logiciels qui ne connaît pas nécessairement le domaine d'activité de l'utilisateur.
- Ce dialogue s'effectue aux travers de documents parfois volumineux et sujets à interprétations.
- Cette complexité est encore accrue par l'évolution du dialogue entre le concepteur qui appréhende de mieux en mieux le domaine et l'utilisateur qui saisit mieux les possibilités de l'informatique et exprime mieux ses besoins.

La difficulté à contrôler les processus de développement

- Les logiciels doivent parfois traiter (en interne) des éléments complexes tout en conservant (en externe) une relative simplicité d'utilisation. Cet objectif peut être atteint en utilisant, par exemple, des interface utilisateurs graphiques et intuitives.
- Ce traitement interne, toujours plus complexe, auquel vient s'ajouter un souci de simplification dans sa présentation, entraîne une augmentation du volume de codage des outils logiciels.

La difficulté à contrôler les processus de développement

- Le processus de développement de tels outils logiciels ne peut plus être appréhendé (en un temps raisonnable) par une seule personne et doit faire l'objet d'un travail d'équipe.
- La complexité organisationnelle vient s'ajouter à la complexité du logiciel à développer.

La flexibilité dans la programmation

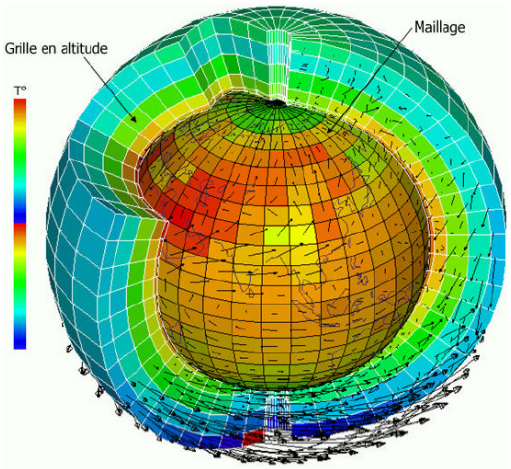
- Des industries comme le bâtiment peuvent utiliser les services d'autres industries comme la métallurgie pour construire des édifices car toutes sont tenues de respecter des normes.
- Un industriel comme Eiffage peut construire un pont mixte en utilisant des poutres d'acier produites par un autre industriel comme sa filiale Effel simplement en spécifiant le type normalisé de poutre dont il a besoin (une poutre d'un type donnée est construite d'une certaine manière et possède des caractéristiques mécaniques particulières).

La flexibilité dans la programmation

- L'industrie du logiciel possède moins de normes, les concepteurs sont alors tentés de créer leurs propres briques de base pour s'assurer qu'elles répondent parfaitement à leur besoins : le développement logiciel en devient donc d'autant plus laborieux.

Le passage du monde continu au monde discret

- La modélisation du réel, qui est régie par des lois physiques continues, s'appuie sur des éléments discrets possédant un nombre fini d'états.
- En météorologie, l'atmosphère est modélisée par un ensemble de cubes où la température et la pression sont considérées comme homogènes.

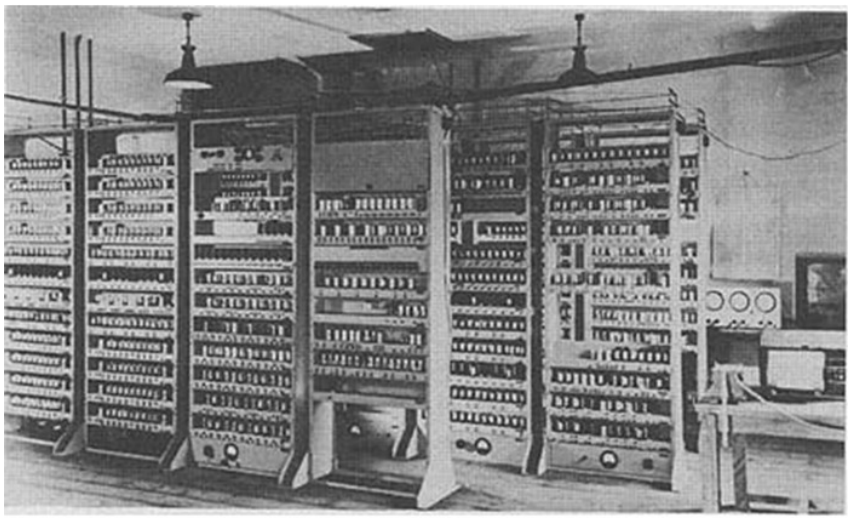


Le passage du monde continu au monde discret

- Cette discrétisation introduit une erreur artificielle qui doit être gérée par le logiciel (par exemple un dépassement de capacité ne doit pas introduire un comportement anormal du modèle).
- L'état du modèle discret est la combinaison des états des éléments qui composent ce modèle, le programmeur doit donc gérer une combinatoire importante qui amène un modèle avec un grand nombre d'états.

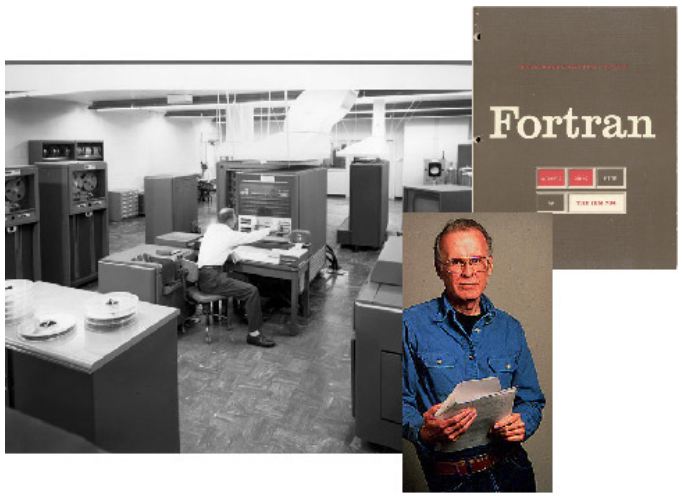
Premiers programmes en langage machine

- Les premiers programmes, écrits en langage machine, dépendaient fortement de l'architecture des ordinateurs utilisés.



Programmation en langages évolués

- Lorsque le nombre d'architectures différentes a augmenté, un premier effort a été produit pour séparer les concepts manipulés dans les langages de leur représentation dans la machine et a aboutit à la création de langages comme FORTRAN.

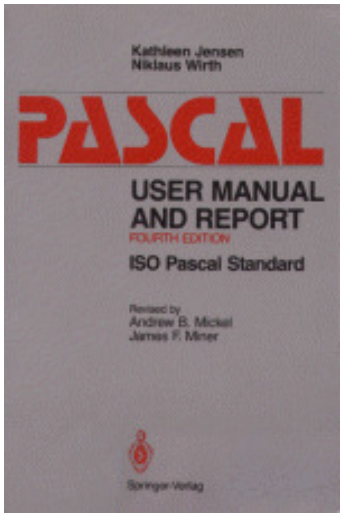


Méthode d'analyse par décomposition

- La complexité croissante des programmes a de nouveau suscité un effort pour mieux structurer les programmes (en abandonner le goto et la programmation spaghetti).
- Les méthodes d'analyse consistait alors à « diviser pour mieux régner », autrement dit à découper les tâches en modules indépendants considérés comme des boites noires [Edsger Dijkstra].

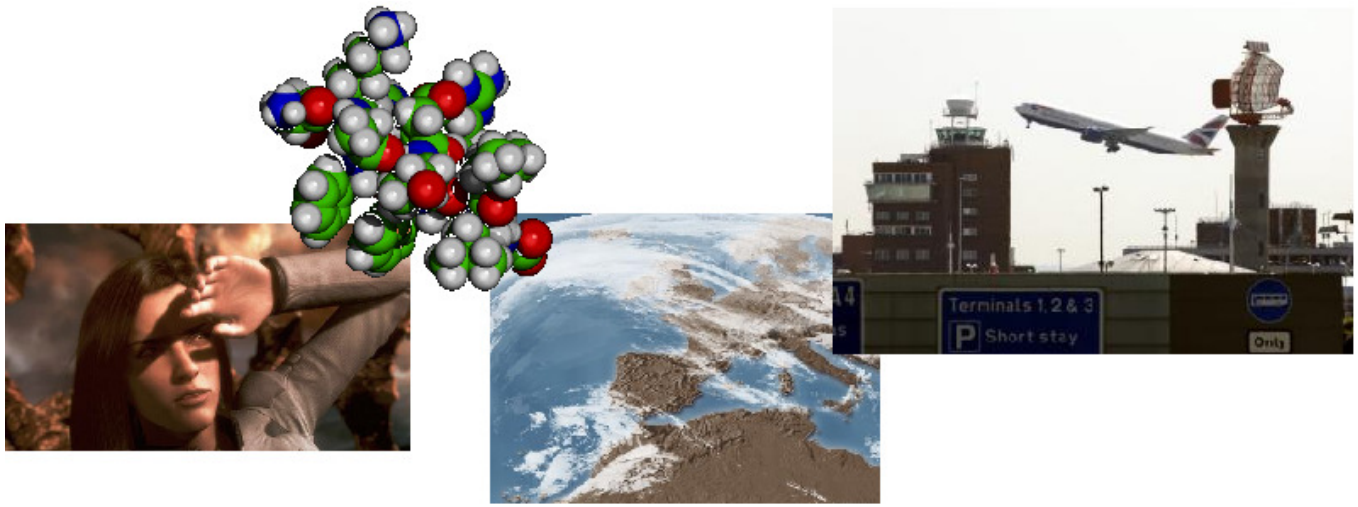
Structuration des programmes

- Niklaus Wirth alla plus loin en considérant les programmes comme la « somme » d'une structure de données et d'un ensemble distinct d'algorithmes chargés de les manipuler.
- La programmation structurée étaient alors synonyme de programmation dirigée par les traitements.



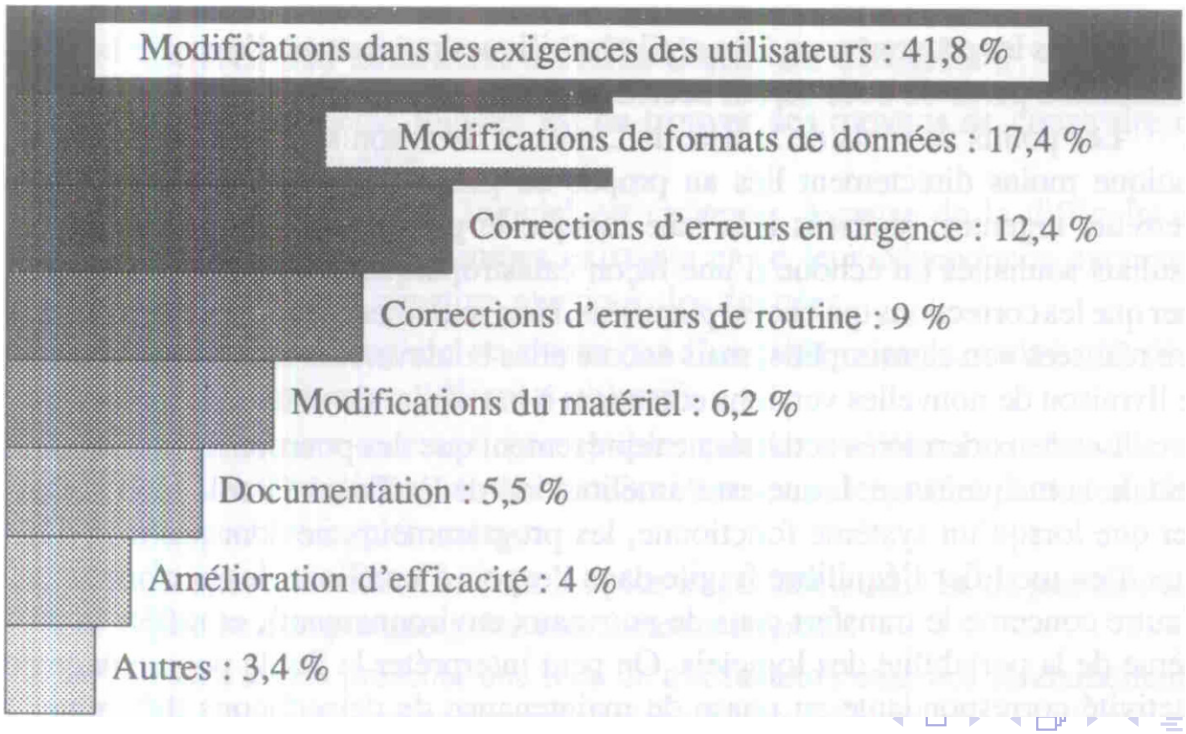
L'explosion des besoins

- Le coût du matériel informatique a fortement décrût et ce matériel est devenu un bien de consommation courant (tant dans des entreprises et des universités que chez les particuliers).
- La clientèle s'est diversifiée, les besoins ont explosé.



Des besoins très variés

- Certains besoins concernent la création de logiciels mais d'autres besoins concernent la « maintenance » de logiciels (70 % du coût du logiciel).
- 40 % de cette maintenance sont des extensions ou des modifications demandées par les utilisateurs



La diffusion de la structure des données dans le code

- La programmation structurée nécessite de faire des hypothèses sur les données à traiter .
- La structure physique des données à traiter est diffusée dans une part importante du code.
- Une simple modification des exigences des utilisateurs ou une modification des données peut entraîner d'importantes modifications au niveau des codes chargés de les traiter.
- Par exemple, en 1982, les postes américaines sont passées du code postal à 5 chiffres à celui à 9 chiffres. Beaucoup de programmes gérant des fichiers d'adresses ont dû être récrits à grands frais.

Synthèse de la situation avant la POO

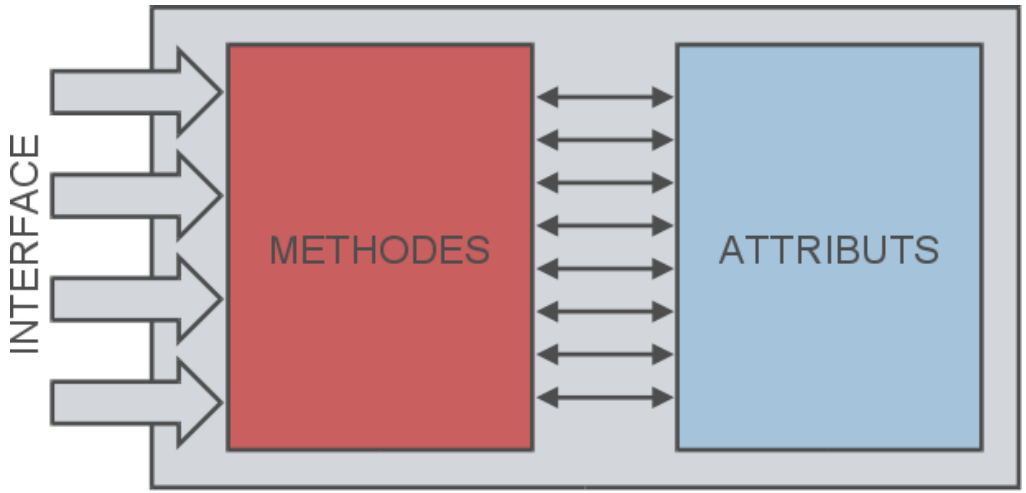
- D'un côté des utilisateurs formulaient des exigences qui leur semblaient simples
- De l'autre des concepteurs de logiciels ne maîtrisaient pas la complexité logicielle notamment à cause de la programmation structurée.
- Nous avons alors abouti à une explosion des délais et des budgets informatiques ce qui s'est traduit à grande échelle par une crise du logiciel.

Plan

1. Le principe d'encapsulation.
2. Le principe de modularité.
3. Le principe d'abstraction.
4. Synthèse.

Le principe

- Le principe de l'encapsulation est « d'enfermer » (de cacher) les données dans des entités logicielles appelées objets qui fournissent des méthodes pour accéder « proprement » en lecture et en écriture à ces données que nous appelons attributs (ou champs).



L'objectif et les conséquences

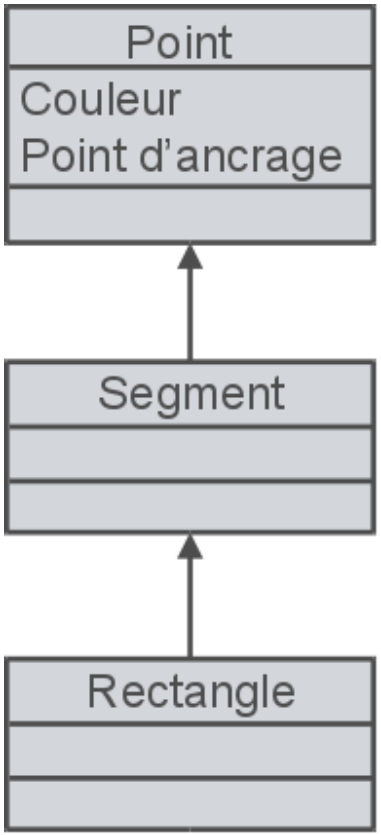
- Cette encapsulation avait pour principal objectif de confiner la structure physique des données à l'intérieur de l'objet afin de limiter l'impact d'une éventuelle modification dans la structure des données.
- L'adaptabilité du code est améliorée donc l'écriture du code et sa maintenance sont facilitées.

L'héritage et le polymorphisme

- D'autres concepts comme l'héritage et le polymorphisme ont été mis en place plus tard dans les langages de POO afin de faciliter la ré-utilisabilité du code :
  - productivité accrue des équipes de programmeurs ;
  - extensibilité du code améliorée.

L'héritage

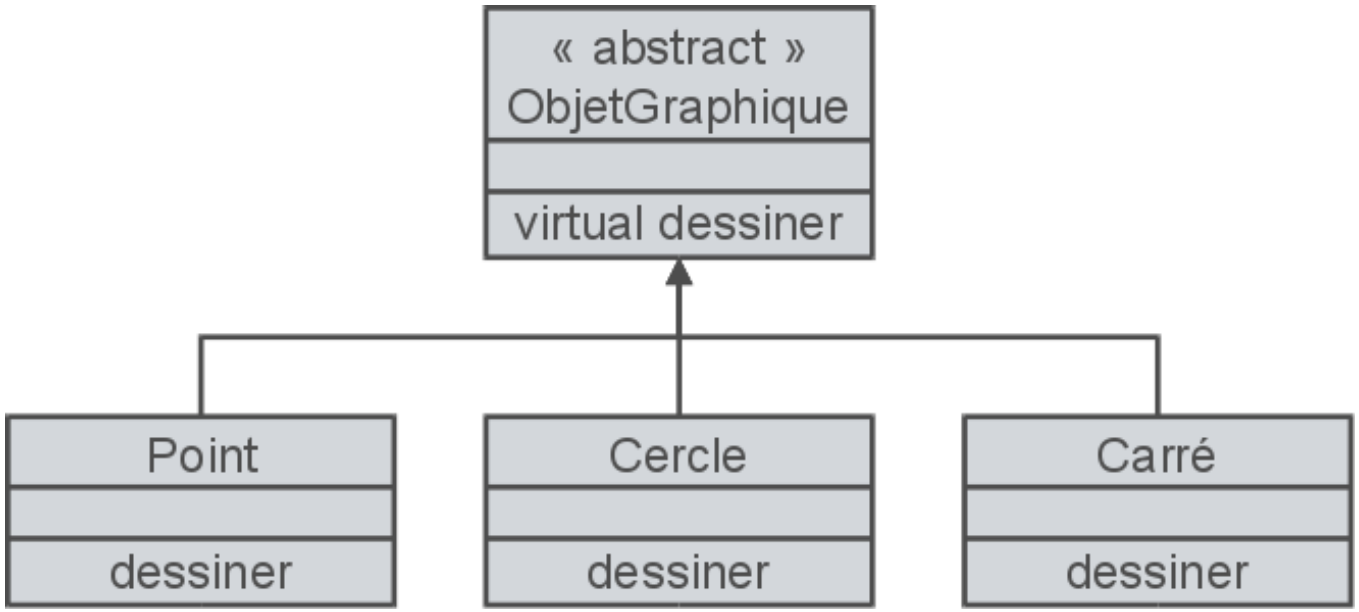
- Ces objets graphiques ont des points communs comme la couleur ou le point d'ancrage : ces attributs peuvent être déclarés une seule fois au niveau de l'objet Point, ils seront connus de la descendance.
- De même les méthodes permettant de lire ou de modifier ces attributs seront codées une seule fois au niveau de Point pour être connues de la descendance.



Le polymorphisme

- Soit des objets graphiques (le point, le cercle, le carré ...) munis d'une même méthode ≪ dessiner ≫ qui se comporte différemment selon le type d'objet.
- Nous créons un objet ≪ abstrait ≫ appelé objetGraphique muni d'une méthode virtuelle ≪ dessiner ≫.
- On créé un tableau d'objetGraphique et lorsque l'on implante une boucle où est appelée la méthode dessiner de chaque élément, le système détermine automatiquement la bonne méthode dessiner à utiliser.

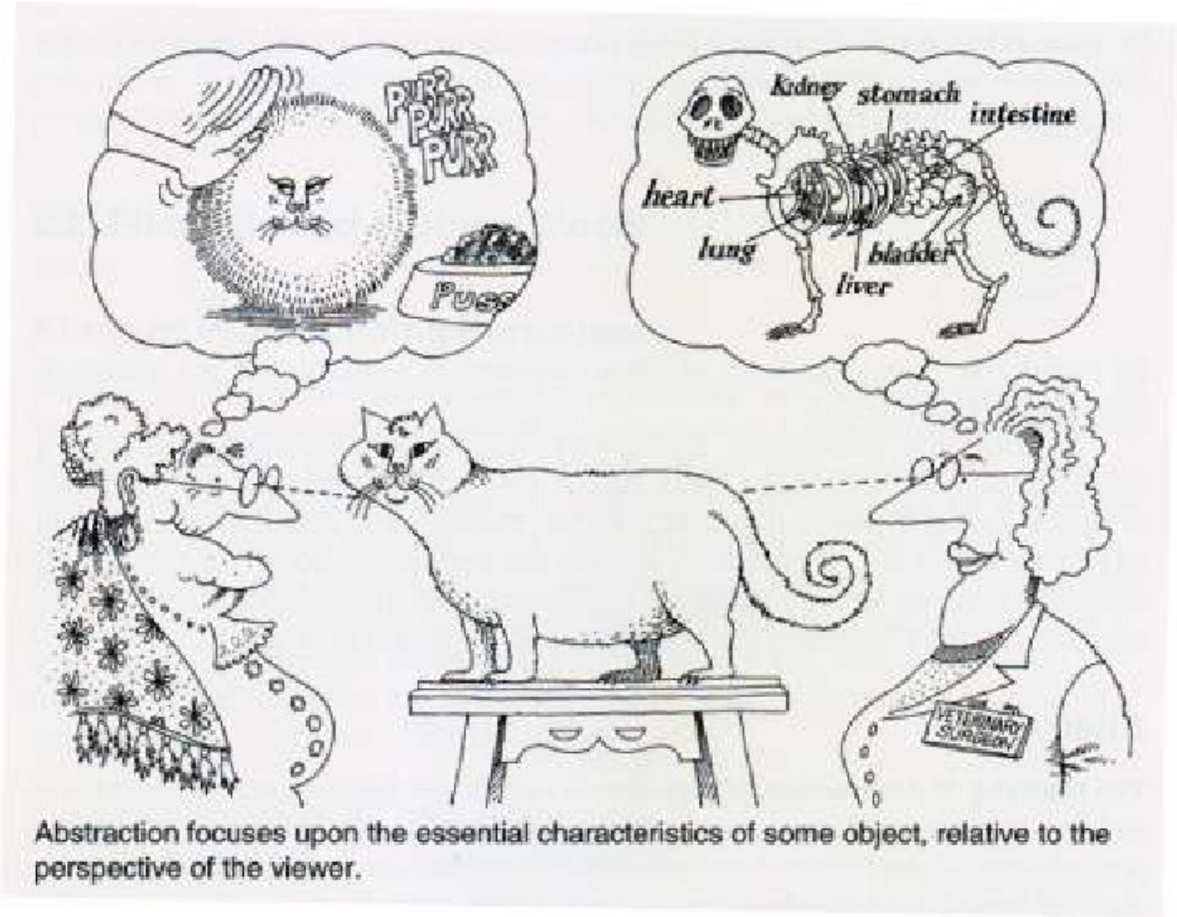
Le polymorphisme



Le principe

- Chaque acteur de la conception possède son propre point de vue sur l'objet qui est conditionné par son domaine d'activité professionnelle, par sa culture et sa sensibilité.
- Les niveaux d'abstraction perçus par chacun sont alors différents
- L'objet à concevoir doit pouvoir être décrit en respectant différents niveaux d'abstraction (du général au précis).

Les différents points de vue (selon G. BOOCH)



Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

La pratique

- D'un point de vue pratique, ce principe d'abstraction est atteint en utilisant une hiérarchie d'objets :
  - des objets qui englobent
  - d'autres objets qui englobent
  - d'autres objets plus petits qui contiennent
  - des objets atomiques.
- Les concepteurs peuvent donc manipuler un objet dans sa globalité ou dans un niveau de détail plus fin.

Les 3 principes de la modélisation objet