Livre Blanc

F. BERTON / CIEL .S.A. / Groupe SAGE / FRANCE fberton@cielsoft.com

Introduction au concept objet.

Objet :

Le but de ce document est de sensibiliser l'ensemble des acteurs, intervenant dans le cycle de développement d'un logiciel, à la modélisation et la conception orientées objets.
Ce document ne prétend pas faire une approche détaillée de la technologie objet, mais seulement une vulgarisation de celle-ci. Si vous souhaitez approfondir le sujet, je vous conseille de vous reporter à la bibliographie.

Table des matières :

Les composants objets, une nouvelle approche du développement

Extrait de "Au cœur de ActiveX et OLE de David Chappel, Microsoft Press "

Au cours des 35 dernières années, les concepteurs de matériel informatique sont passés de machines de la taille d'un hangar à des ordinateurs portables légers basés sur de minuscules microprocesseurs. Au cours des même années, les développeurs de logiciels sont passés de l'écriture de programmes en assembleur et en COBOL à l'écriture de programmes encore plus grands en C et C++. On pourra parler de progrès (bien que cela soit discutable), mais il est clair que le monde du logiciel ne progresse pas aussi vite que celui du matériel. Qu'ont donc les développeurs de matériel que les développeurs de logiciels n'ont pas ?
La réponse est donnée par les composants. Si les ingénieurs en matériel électronique devaient partir d'un tas de sable chaque fois qu'ils conçoivent un nouveau dispositif, si leur première étape devait toujours consister à extraire le silicium pour fabriquer des circuits intégrés, ils ne progresseraient pas bien vite. Or, un concepteur de matériel construit toujours un système à partir de composants préparés, chacun chargé d'une fonction particulière et fournissant un ensemble de services à travers des interfaces définies. La tâche des concepteurs de matériel est considérablement simplifiée par le travail de leurs prédécesseurs.
La réutilisation est aussi une voie vers la création de meilleurs logiciels. Aujourd'hui encore, les développeurs de logiciels en sont toujours à partir d'une certaine forme de sable et à suivre les mêmes étapes que les centaines de programmeurs qui les ont précédés. Le résultat est souvent excellent, mais il pourrait être amélioré. La création de nouvelles applications à partir de composants existants, déjà testés, a toutes chances de produire un code plus fiable. De plus, elle peut se révéler nettement plus rapide et plus économique, ce qui n'est pas moins important.

Qu'est ce qu'un objet

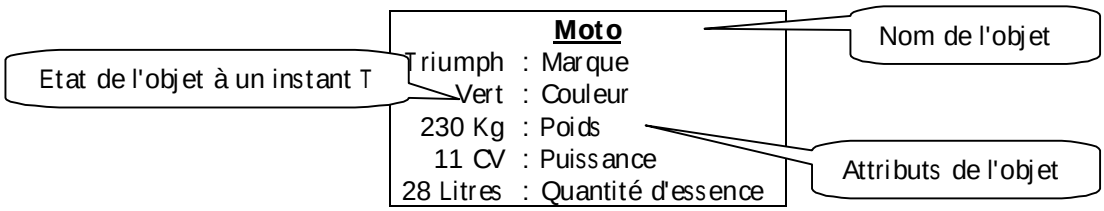
Le monde dans lequel nous vivons est constitué d'objets. Ils sont caractérisés par leur taille, leur poids, leur couleur, etc. Certains de ces objets sont très petits tel un grain de sable, d'autres sont très grands comme un immeuble. Parallèlement au sable et aux immeubles qui ont une existence très concrète, grâce à leur aspect matériel, nous utilisons également des objet de nature "virtuelle" n'ayant pas d'existence matérielle comme : un compte en banque, une police d'assurance,... Nous sommes donc confrontés en permanence à des objets. Une des grandes caractéristiques de l'intelligence humaine est sa capacité à manipuler des concepts complexes à différents niveaux d'abstraction ([1]) et ce, de façon quasiment innée. Ainsi, lorsque je demande une feuille de papier à quelqu'un, je dis : "Passe moi une feuille, s'il te plaît" et non : "Passe moi une représentation non euclidienne d'un plan, composé de fibres végétales savamment blanchies, s'il te plaît".
Les méthodes actuelles de développement informatique permettent de manipuler le concept d'objets. L'informaticien et le non informaticien peuvent avoir un langage commun basé sur ce concept. Un objet au sens informatique du terme permet de désigner une représentation "abstraite" d'une chose concrète du monde réel ou virtuel. Un objet (au sens informatique, que j'appellerai maintenant objet) présente les 3 caractéristiques suivantes :

Objet = État + Comportement + Identité

- **L'état** d'un objet définit la valeur des données (ou attributs), par exemple dans le cas d'un objet Moto, celui-ci pourrait être caractérisé par les attributs suivants : la marque, la

[1]) *Comme le décrit WULF, "Nous (les humains) avons développé une technique exceptionnellement puissante pour traiter la complexité. Nous en faisons une abstraction. Incapable de maîtriser la totalité d'un objet complexe, nous avons choisi d'ignorer ses détails non essentiels et de traiter le modèle généralisé et idéalisé de l'objet".*
Extrait de Analyse & conception orientées objets, Grady Booch, Addison Wesley,2me édition, 1994

couleur, le poids, la puissance, la quantité d'essence... Ce que l'on représente graphiquement par :



L'état de l'objet peut être amené à changer durant son cycle de vie. Par exemple, la quantité d'essence et le poids de la moto varient en permanence lorsque celle-ci roule.

- **Le comportement** d'un objet indique toutes les compétences de celui-ci et décrit les actions et les réactions qu'il peut avoir. Chaque élément de base du comportement est appelé opération. Les opérations d'un objet sont déclenchées suite à une stimulation externe de l'utilisateur qui appuie sur un bouton ou du programmeur qui appelle une opération. Il existe un lien très étroit entre le comportement d'un objet et son état. Effectivement, l'état d'un objet peut être modifié par l'appel d'une opération et les actions effectuées par une opération dépendent de l'état de l'objet. Reprenons notre exemple de l'objet Moto. Nous pourrions définir les opérations Démarrer, Rouler, Stopper. L'opération Démarrer n'a de sens que si l'information "moteur éteint" est vérifiée. De même, l'opération Stopper n'aura d'effet que si les opérations Démarrer et Rouler ont déjà été exécutées modifiant ainsi l'état de l'objet moto de "moto stoppée" à "moto en déplacement".

- **L'identité** : En plus de son état et de son comportement, un objet possède une identité qui caractérise son existence propre. L'identité permet de distinguer tout objet de façon non ambiguë, et cela indépendamment de son état. Cela permet, entre autres, de distinguer deux objets dont toutes les valeurs d'attributs sont identiques. L'identité est souvent construite à partir d'un identifiant issu naturellement du domaine du problème. Ainsi une moto est identifiée par son numéro de série, un compte en banque par son numéro de compte.

Voici une représentation complète d'un objet Moto selon la méthode UML ([2]).

TROPHY 4 : Moto
0001 : Numéro de Série Triumph : Marque Vert : Couleur 230 Kg : Poids 11 CV : Puissance 28 Litres : Quantité d'essence
Démarrer() Rouler() Stopper()

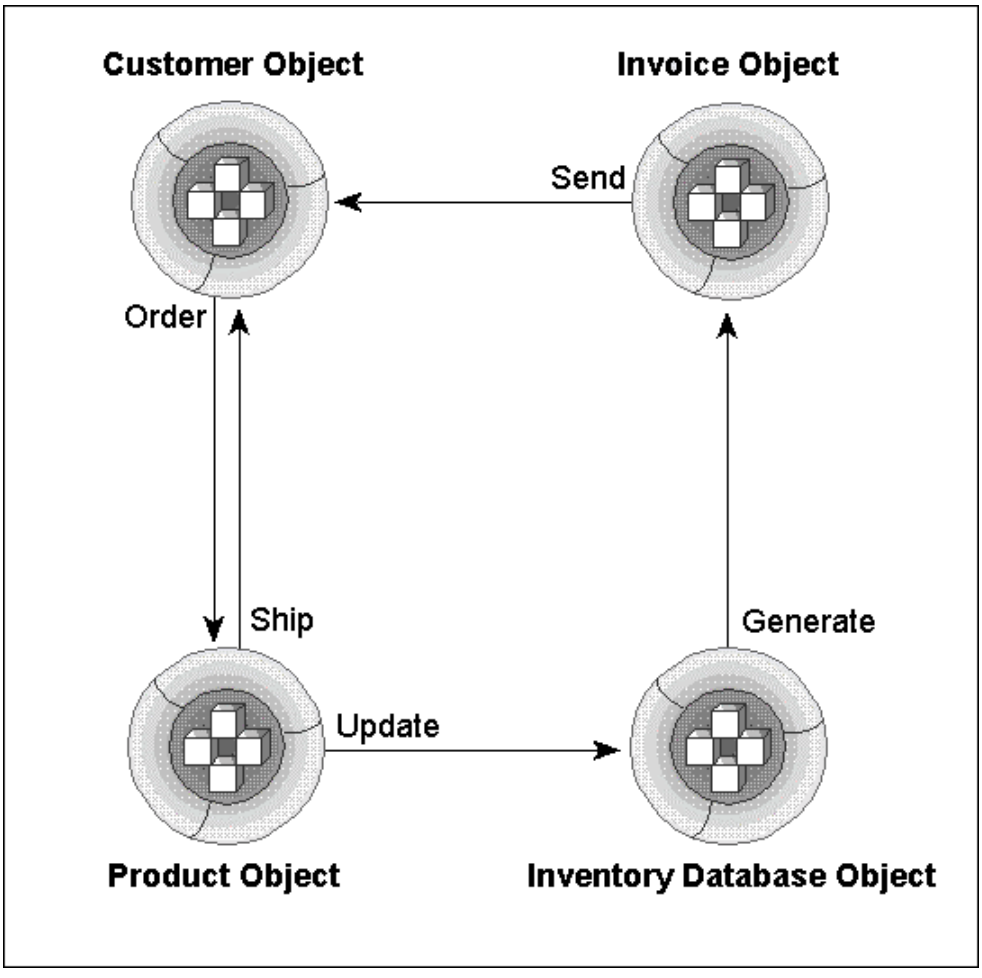
[2]) **UML (Unified Modeling Langage)**, développée en réponse à l'appel à propositions lancé par l'OMG (Object Management Group), dans le but de définir la notation standard pour la modélisation des applications à l'aide d'objets. La notation UML représente l'état de l'art des langages de modélisation objet. Elle se place comme le successeur naturel des notations des méthodes de Booch, OMT (Object Modeling Techniques) de Jim Rumbaugh et OOSE (Object Oriented Software Engineering) de Ivar Jacobson.

La partie gauche illustre un objet particulier (ou instance) la "Trophy 4" de classe "Moto" illustré dans la partie droite. Mais qu'est ce donc que cette histoire de classe ?
Comme son nom l'indique une classe permet de définir une classification. Ainsi tous les objets "moto" partagent une même classe à savoir la "classe moto". Les notions de classe et d'objet sont relativement difficiles à définir indépendamment l'une de l'autre. La classe d'un objet définit par ses attributs et ses opérations : l'état, le comportement, et l'identité d'un objet de façon abstraite ; alors qu'un objet est une représentation concrète d'une classe. On parle d'instance d'une classe. Dans notre exemple la classe moto définit que tout objet de classe moto comporte un numéro de série, et l'instance TROPHY 4 de la classe moto défini un objet unique via son numéro de série.

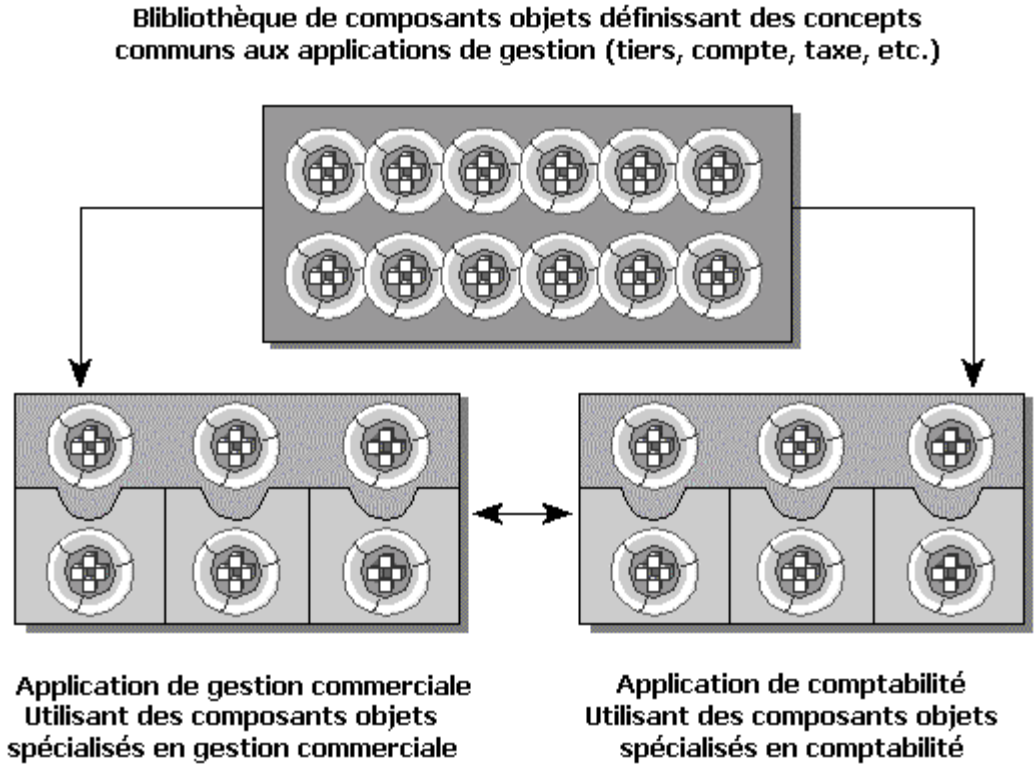
Objets et composants objets.

Nous avons maintenant une idée de ce qu'est un objet. Comme nous l'avons vu, il existe des objets plus ou moins complexes. Dans certains cas, la complexité est telle qu'un seul objet ne peut permettre de résoudre à lui seul la problématique, on utilise donc plusieurs objets représentant chacun une petite partie du problème. Les objets interagissent ensemble afin de produire une solution. Il s'agit de composants objets. Des composants objets peuvent, à leur tour, être assemblés afin de former un système encore plus complexe. Cette approche "système, composant, objet" permet de résoudre les problèmes les plus complexes.

Le dessin illustre un composant objet de gestion de commande client (le cadre autour des objets symbolise le domaine du composant). Il fait collaborer 4 objets: Customer (client), Invoice (commande), Product (article), Inventory database (mouvement de stock).



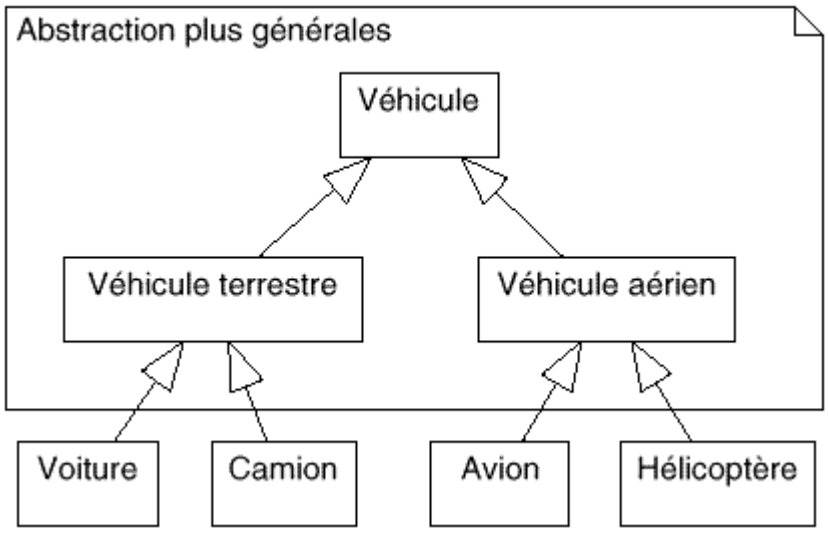
Le dessin suivant illustre un système complexe faisant intervenir plusieurs composants objets, afin de produire deux logiciels, utilisant des composants objets communs au domaine des deux applications et des composants plus spécifiques au domaine de chaque logiciel. Les flèches indique les communications entre les composants objet. Ainsi le logiciel de gestion commerciale peut utiliser des composants objets proposés par le logiciel de comptabilité pour générer des écritures comptables liées à une facture.



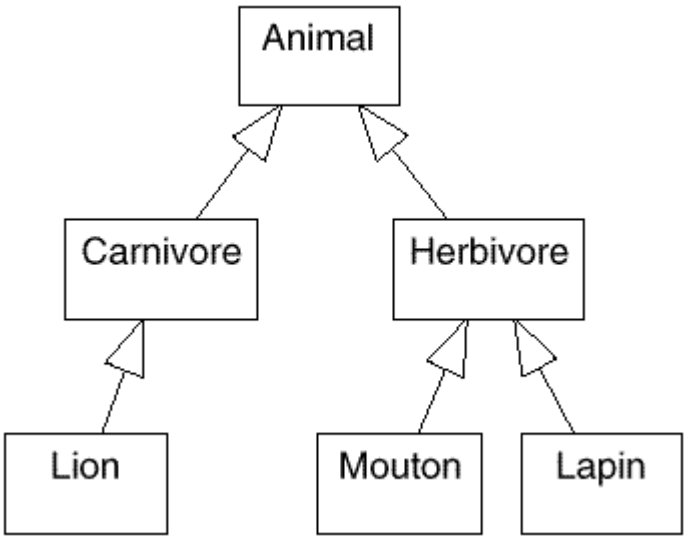
La généralisation (ou héritage) et la spécialisation

Comme nous l'avons vu, les objets sont définis par des classes, pour apporter une notion de classification, or lorsque l'on fait de la classification, on se rend compte que certaines classes ont des comportements communs. La généralisation consiste à factoriser les éléments communs (attributs, opérations) d'un ensemble de classes dans une classe plus générale (on simplifie le problème en augmentant son niveau d'abstraction) appelée super-classe. Les classes sont ordonnées selon une hiérarchie ; une super-classe est une abstraction de ses sous-classes. La généralisation est une démarche assez difficile car elle demande une bonne capacité d'abstraction. La mise au point d'une hiérarchie optimale est délicate et itérative.

L'exemple suivant montre une hiérarchie des moyens de transport. La flèche qui symbolise la généralisation entre deux classes pointe vers la classe plus générale.



La généralisation indique toujours une relation de type "*est un*" ou "*est une sorte de*" (indiqué par une flèche de la sous classe vers la super classe). Dans l'exemple suivant, le lion est une sorte de carnivore : un lion *est un* carnivore.



Comme vous l'avez peut être constaté, les arbres de classes ne poussent pas à partir de leur racine. Au contraire, ils se déterminent en partant des feuilles car les feuilles appartiennent au monde réel alors que les niveaux supérieurs sont des abstractions construites pour ordonner et comprendre.

Conclusion

J'espère que ce document vous à permis de mieux comprendre ce qu'est la conception d'application par objet.
Les méthodes d'analyse objet définissent maintenant des outils adaptés aux différants niveau de besoins des acteurs du développement de logiciel. Ainsi ses méthodes permettent de formaliser des besoins, de définir des cas d'utilisation, de spécifier une architecture, et de géré des cycles de conception/développement itératif et incrémental.

Bibliographie

En français

Le monde de Sophie, Jostein Gaarder, Seuil, 1995, (tout publique)
Modélisation objet avec UML, Pierre-Alain Muller, Eyrolles, 1997, (tout publique)
OMT Modélisation et conception orientées objet, James Rumbaugh et al., Masson/Prentice Hall, 1994 , (tout pulique)
Analyse & conception orientées objets, Grady Booch, Addison-Wesley, 1994, (Informaticiens)
Le langage C++, Bjarne Stroustrup, Addison-Wesley, 1992, (Informaticiens)
Au cœur de ActiveX et OLE, David Chappell, Microsoft Press, 1996, (Informaticiens)

En anglais

What OLE is Really About, Kraig Brockschmit, White paper Microsoft, (tout publique)
The Microsoft object technology strategy: Component software, White paper Microsoft, (tout publique)