TABLE RONDE "PARADIGME ORIENTE OBJET"
animée par André Delacharlerie.

Jean-Pierre Peyrin :

Plutôt que d'opposer la programmation par objets à la programmation impérative, il serait plus pertinent d'opposer, d'une part, la programmation impérative à la programmation déclarative, et, d'autre part, la programmation par objets à la programmation structurée.

La programmation impérative est l'expression du "comment faire" ; elle décrit la succession des opérations que doit effectuer l'ordinateur pour construire un résultat qui n'est pas précisé. La programmation déclarative est l'expression du "quoi faire" ; elle précise le résultat attendu sans décrire le processus qui permet à l'ordinateur de l'obtenir. Ainsi Pascal, Lisp et Eiffel sont des langages essentiellement impératifs, alors que Prolog et Hypercard sont des langages essentiellement déclaratifs.

La programmation structurée [1] est fondée sur une analyse descendante : le problème posé est résolu par une action (une fonction) qui, si elle n'est pas primitive, est l'expression d'une composition d'actions (de fonctions) plus élémentaires qui, si elles ne sont pas primitives, etc. Une extrême rigueur dans l'analyse des problèmes a été particulièrement nécessaire après les années noires du "tripatouillage" (c'était la "guerre du goto" et les enseignants étaient les "incorruptibles"). Cette rigueur contrait les penchants des élèves, naturellement constructifs et opératoires.

La programmation par objets est fondée sur une analyse essentiellement ascendante : le problème posé est d'abord résolu par une réutilisation rationnelle de classes existantes. Plus précisément, cette programmation rétablit un sain équilibre entre deux modes de raisonnement nécessairement complémentaires.

La programmation modulaire est centrée sur la relation "avoir" (importer). Par exemple, un avion a deux ailes (le module "avion" importe le module "aile").

La programmation par objets ajoute la relation "être" (hériter). Par exemple, un avion est un moyen de transport (la classe "avion" hérite de la classe "moyen de transport", c'est-à-dire qu'elle en a toutes les propriétés ; elle peut aussi hériter de la classe "engins volants", l'héritage peut donc être multiple).

Dans l'élaboration d'un programme, on peut avoir un certain choix sur le type d'une relation à définir : ce choix signifie un point de vue particulier et se traduit alors par une expression particulière. Par exemple, on peut dire qu'un homme a une tête, mais on peut dire aussi qu'un homme est une tête (la classe "homme" peut donc importer la classe "tête", mais elle peut aussi en

[1] telle qu'elle a été définie en 1972 par Dahl, Dijkstra et Hoare dans "Structured Programming".

hériter). La programmation par objets permet ainsi un raffinement de l'analyse et de l'expression [2].

La programmation par objets n'est pas le dernier gadget à la mode des informaticiens. C'est simplement l'aboutissement actuel de la réflexion sur la programmation et de ses réels dangers. La question sur sa place dans l'enseignement ne se pose donc pas. Si la programmation doit être enseignée (et ceci est une vraie question !), elle ne peut qu'être "par objets".

Programmer "par objets" ne signifie pas utiliser un langage "à objets". On peut "tripatouiller" avec Eiffel et "bien programmer" avec Turbo Pascal. La conciliation entre l'approche "traditionnelle" et l'approche "objets" se fait naturellement si l'on pense "méthode" plutôt que "langage". Certes, une approche "objets" des problèmes conduit à préférer les langages "à objets", comme la programmation structurée conduisait à préférer Pascal à Basic.

La situation de programmation [3] rencontrée dans l'utilisation d'un logiciel bureautique est souvent proche de la programmation par objets. On définit des "objets" par "instanciation" de classes prédéfinies (une rubrique, un champ, …), on en précise les attributs (caractéristiques, format, …), on en précise les méthodes (formules, liens, …). Il ne s'agit pas d'en déduire nécessairement un discours conceptuel, mais d'en déduire surtout une pratique raisonnée.

Si l'on doit (veut ?) vraiment enseigner la programmation dans le cadre d'une formation générale, il semble hors de question d'imposer la rigueur conceptuelle et syntaxique de langages tels que SmallTalk ou Eiffel, voire C++. A l'heure actuelle, des environnements tels qu'HyperCard ou ToolBook sont vraisemblablement ce que l'on peut faire pratiquer avec le moins de contraintes inutiles. Ce ne sont pas des langages à objets, mais les concepts sous-jacents en sont proches et suffisants pour un premier niveau cohérent de compréhension ; en outre, l'expression est essentiellement déclarative. Réunir en un seul environnement les deux paradigmes "orienté objets" et "déclaratif" n'est-il pas un vœu souvent formulé ?

[2] discours "hérité" de Jean-Claude Boussard, Professeur à l'Université de Nice - Sophia Antipolis.

[3] faire faire quelque chose à un ordinateur, en différé.