PETITES HITOIRES

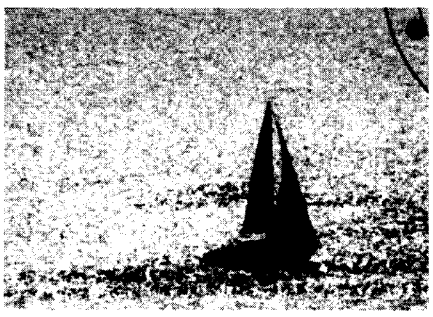
- Une fois que l'on a prouvé que le problème est décidable, c'est-à-dire qu'il existe théoriquement un algorithme, il suffit de le programmer, dit le Logicien. D'ailleurs les langages de programmation universels n'ont-ils pas la capacité théorique d'exprimer tout algorithme ?
- Eh bien pas tout-à-fait, dit l'Informaticien. Laissons même de côté l'insigne faiblesse des machines, avec leur mémoire de travail ridicule, leur fréquence d'horloge poussive et leur registre d'instructions insignifiant. Il faut bien que quelqu'un l'écrive, ce programme. Au delà de 20 à 30 000 lignes, aucune chance pour qu'un individu en suive la logique. Si vous le voulez plus long, le programme sera écrit par toute une équipe, et attention à l'intégration. Il me faut donc un langage concis et modulaire. Ça marchera si l'algorithme est honnêtement court. Si l'algorithme lui-même est très complexe, avec beaucoup de cas très combinatoires, tous les langages séquentiels n'y parviendront pas !
- Bon, dit le Chercheur en informatique, essayez alors ces langages de style norvégien qui vous découpent les problèmes en petits algorithmes répartis dans des objets. Ça devrait devenir plus simple.

Sa voix est alors recouverte par celle du marketing qui susurre :

- Objets, objets, qui veut mon langage à objets, ma base de données à objets, ma programmation en objets, ma conception par objets...

FRANGLAIS

Ce n'est pas parce que LOO veut dire "chiottes" en Anglais qu'il faut baptiser Orientés Objets tous les langages à objets. D'ailleurs "loo", c'est une déformation de "l'eau", vous voyez, au fond du couloir, là où les français disent "water closet".



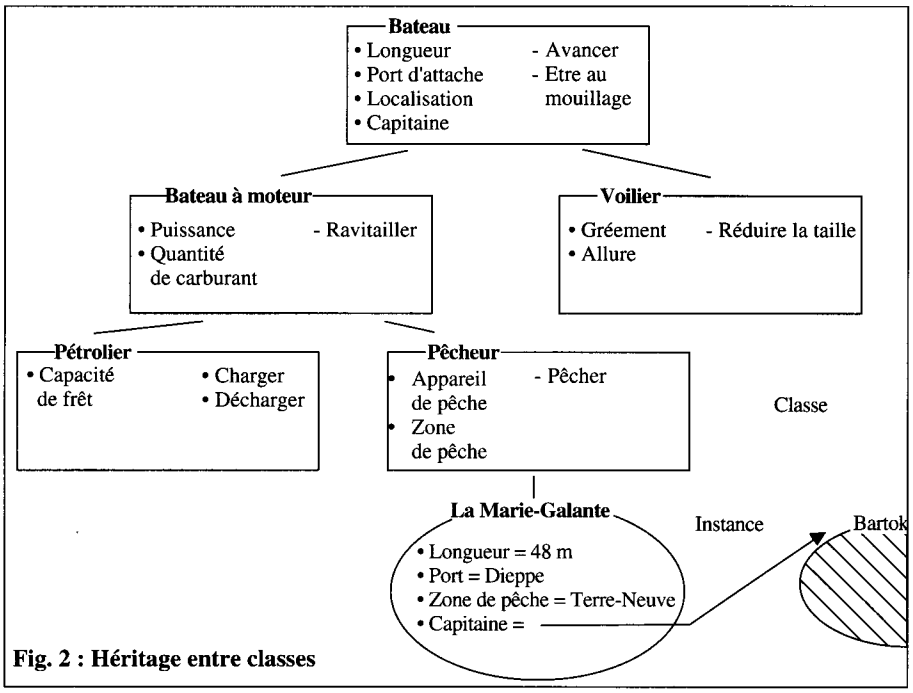


Fig. 2 : Héritage entre classes

pas confondre avec les exceptions d'exécution d'un programme). Par exemple tous les salariés relèvent de la Sécurité Sociale et votre programme de paye traite les salariés de cette manière. Mais voilà justement que la société ouvre des chantiers à l'étranger et que leurs salariés relèvent d'un autre régime : votre programme Cobol ne peut traiter leurs salaires. C'est comme ça que se mettent à foisonner les programmes-verrues et les passerelles. Avec un langage à objet "il suffit" de créer une sous-classe Expatrié de la classe Salarié. Dans la nouvelle classe Expatrié, la procédure Prélèvements sociaux est programmée selon la règle de gestion des chantiers à l'étranger.

SURCHARGE : L'AUTRUCHE NE VOLE PAS

Ce procédé de *surcharge* évite la multiplication des noms de variables et de procédures. Selon l'objet qui est concerné, la variable et la procédure sont identifiées par le langage. Naturellement si la classe ne contient pas de définition propre, le langage cherche dans la sur-classe, qui fonctionne ainsi comme une définition par défaut. Voici en objets la solution de casse-tête logique des au-

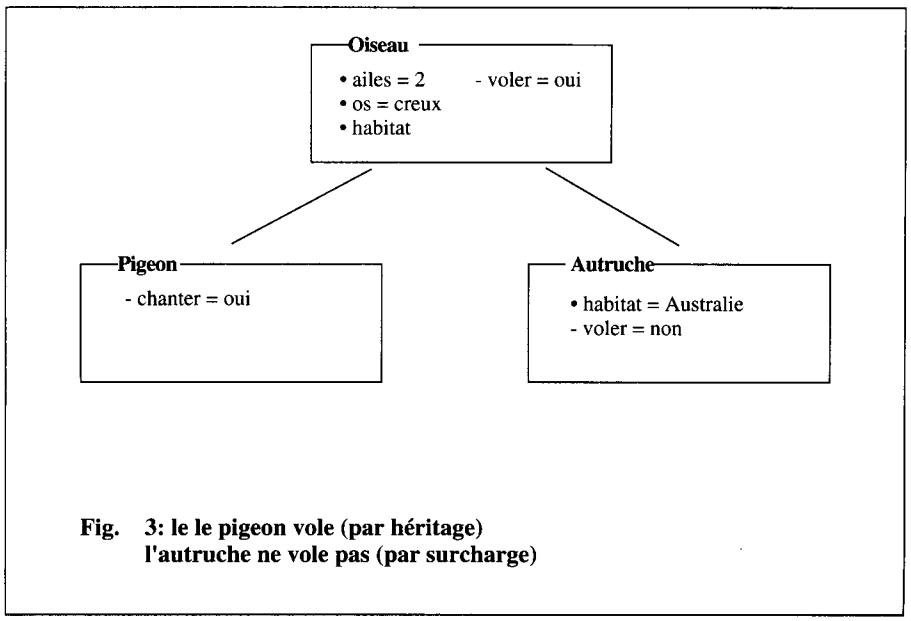


Fig. 3: le le pigeon vole (par héritage) l'autruche ne vole pas (par surcharge)

truches : tous les oiseaux volent, les autruches sont des oiseaux, donc...

Dans les langages typés classiques comme C, Pascal ou Ada, la liaison entre la variable et sa valeur est statique : une variable est d'un type et d'un seul, celui qui a été déclaré explicitement et qui est contrôlé à la compilation. Il faut déclarer "l'addition des choux" et l'addition des carottes", et malheureusement autant d'opérateurs addition que votre petit commerce des quatre saisons comporte de légumes. Certains langages à objets utilisent les classes avec la *liaison dynamique* entre variables et valeurs, et c'est ce qui permet la surcharge. Le type de la variable (ou le corps de la procédure) n'est pas défini par son seul nom mais par son association avec un nom d'objet, éventuellement inconnu au moment de la compilation. On peut exprimer l'addition en disant seulement que les termes doivent être du même type. Si les premiers langages à objets *(Simula, Smalltalk)* opéraient des liaisons statiques, leurs descendants utilisent la liaison dynamique.

UN COCOTIER POUR LES VIEUX PROGRAMMEURS

Du point de vue d'une direction des études informatiques à qui l'on demande de produire rapidement et de maintenir des applications fiables, choisir un langage à objets est difficile.

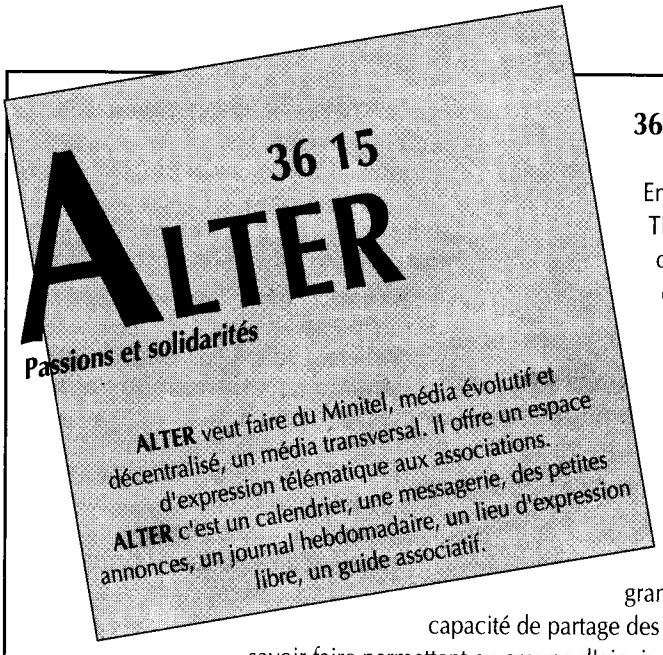
Sur le papier il y a déjà de nombreux handicaps, à commencer par les langages eux-mêmes. Certains sont seulement interprétés, d'autre compilés sans pouvoir utiliser de bibliothèques externes, sans parler des manques d'accès aux systèmes de gestion de bases de données existants. Peu disposent d'un environnement de programmation élaboré sur des systèmes d'exploitation usuels, ou même seulement sur *Unix*. Cela les limite souvent au maquettage, comme Smalltalk et son système graphique original, ou des applications particulières, comme les extensions objets de *Lisp*.

C++ et Objective C se greffent au moins sur les bibliothèques et sur les aides à la programmation du langage C, c'est probablement une des raisons de leur essor.

Mais le langage lui-même n'est qu'un ingrédient de l'efficacité de la production de logiciel. Un des plus gros morceaux reste le facteur humain, et attention, pas seulement l'embauche d'un *ingénieur-bien-formé-aux-dernières-techniques*, mais une équipe entière entraînée et motivée. C'est un mélange de compétences individuelles assez nombreuses pour créer une pensée commune, de rodage de manières de travailler ensemble, d'outils de développement personnalisés qui concrétise des choix de méthodes de travail.

Or, s'il est hélas possible de programmer en C dans le style Cobol, comme l'évolution vers C pousse nombre d'équipes, le passage à la programmation en objets est un saut de pensée et de méthode difficile. Et faute de méthode, les petits mécanismes des langages à objets transforment vite un programme en un labyrinthe sans plan. A l'inverse, ceux qui ont acquis méthode et expérience avec des langages à objets programment ensuite avec les langages classiques en identifiant clairement des structures de données. Voici venir le temps de la conception par objets...

■



36 15 ALTER

En 1986, la revue TERMINAL et une dizaine d'associations décident d'ouvrir sur 36 15 un espace Minitel, animés par l'idée que ce nouveau média ne doit pas rester étranger au mouvement alternatif. Une grande ouverture, une capacité de partage des connaissances et du savoir-faire permettent au groupe d'aiguiser son expérience et de surmonter les difficultés financières et techniques.

Aujourd'hui, **ALTER** est consulté plus de 1 500 fois et inscrit dans son service les associations suivantes : **Aldea** (Agence de liaison pour le développement de l'économie alternative), **Cigales** (Club d'investisseurs et d'épargne de solidarité), **Réas** (Réseau de l'économie alternative), **Racine** (Réseau d'accompagnement des créations et des initiatives pour une nouvelle épargne de solidarité), **Solidarité emploi, Coordination infirmières, Union infirmières, Gerpla** (Groupe d'échanges et de recherches pour la pratique en lieu d'accueil), **Laav** (Lieux alternatifs accueil et vie), **La cardabela** (paysans du Larzac), **Ice** (Initiative de Citoyens en Europe), **Mir** (Mouvement international de la réconciliation), **Espace du possible** (vacances alternatives), **Crimgg** (Collectif de réflexion et d'information sur les média dans la guerre du Golfe), **Convivence** (savoir vivre avec l'autre), **Diffusion populaire** (diffusion de revues), **Nature et Progrès** ("pour votre santé et celle de la Terre"), **Terminal-CIII** et **Chimères**.