AFDEL

ASSOCIATION FRANCAISE DE GENIE LOGICIEL

La LETTRE n° 27

Avril 1997

Le RAD (Rapid Application Development)

Quels outils pour quelle méthode ?

Conçue à la fin des années 80 par James Martin et associés, la méthodologie RAD (Rapid Application Development) a beaucoup fait parler d'elle, essentiellement au travers des outils qui s'y réfèrent.

Quel est le rapport entre la première et les seconds ?
Six ans après l'introduction du terme, quel bilan peut-on en tirer ?
La méthode est-elle dépassée ?
Les outils tiennent-ils leurs promesses ?

La méthodologie

La méthode a été décrite pour la première fois par James Martin en 1991. Bien qu'elle ait pu avoir évolué depuis, c'est cet imposant ouvrage (plus de 750 pages) qui nous servira de référence pour la suite. Remarquons d'emblée que le livre ne décrit pas seulement une méthode, mais surtout une méthodologie, c'est-à-dire une réflexion sur ou autour d'une méthode ou d'un ensemble de méthodes. En particulier, l'ouvrage décrit ce que devrait être la méthode idéale de développement rapide.

Une méthode RAD devrait, d'après son auteur, apporter trois avantages compétitifs à l'entreprise :

- une rapidité de développement (cette caractéristique donne son nom à la méthode) ;
- un faible coût de développement ;
- une application de grande qualité.

Ce dernier aspect est souvent négligé par les vulgarisateurs de la méthode, y compris souvent par les fournisseurs des outils qui s'en réclament, au profit de la rapidité du développement.

Le constat de départ concerne les carences des méthodes actuelles : les applications sont très longues à développer. Lorsqu'elles arrivent à l'utilisateur final, elles ne correspondent plus au besoin, car nous vivons dans un monde où tout s'accélère, et les besoins se modifient au même rythme. De plus, l'utilisateur final est peu impliqué dans la définition des besoins, sans parler de la conception de l'application.
Pour pallier ces inconvénients, pas de recette miracle. Mais un nombre restreint d'idées-forces.

Les principes

Les principes du RAD sont simples et ils découlent du bon sens. S'il y a quelque chose de révolutionnaire dans la méthode, c'est leur formalisation.

Voici les principes de base :

- Tout d'abord, le développement devrait être effectué par de petites équipes, expérimentées et ayant reçu toute la formation nécessaire.
- Le développement doit s'effectuer en faisant appel à une méthodologie formalisée.
- Une équipe de développement RAD doit disposer d'outils puissants qui automatisent le développement dans sa globalité, tant en ce qui concerne les étapes du développement que l'enchaînement de ces étapes.
- L'équipe de développement doit être correctement gérée, par un encadrement encourageant la réutilisation des composants, et attentif aux aspects humains du management de projet.
- Enfin, les utilisateurs finaux devraient s'impliquer dans le processus de développement.

La méthode RAD est donc fondée sur quatre ingrédients de base :

- Les **outils** : de conception, de prototypage, de génération de code, disposant d'un langage de haut niveau (L4G), ces outils étant fédérés par un référentiel et constituant un atelier puissant.
- Les **personnes** : elles doivent être compétentes, expérimentées, correctement formées aux techniques et aux outils utilisés, et, cela ira mieux en le disant, ...motivées ! L'auteur revient souvent sur cet aspect fondamental, et un chapitre entier y est consacré. Ce qui est tout aussi essentiel, c'est que l'utilisateur final soit directement impliqué dans chaque phase du projet.
- Le **management** : une gestion du projet, en particulier en ce qui concerne les aspects humains, est essentielle. Le pire ennemi du RAD, c'est la bureaucratie, dit James Martin. Combien de projets n'a-t-on pas vu dériver ou stagner, ensablés par une bureaucratie toujours plus lourde ?
- La **méthodologie** : Rien ne se fera sans une méthodologie, à la fois bien formalisée et souple. L'auteur recommande que la méthode soit consignée, non sur un document qui remplirait la moitié d'une armoire, mais dans un « hyperdocument » que chacun pourrait parcourir rapidement et de façon non linéaire en fonction de ses besoins et de son rôle sur le projet. Ce document doit être adapté d'un projet à l'autre, toujours affiné, et consultable par tous. Seul un document électronique disponible sur un réseau peut convenir.

Les quatre phases

Un projet RAD se découpe en quatre phases :

1. **Phase de définition des besoins**. Elle se matérialise par des sessions de travail appelées **JRP** (Joint Requirements Planning ou définition conjointe des besoins).
2. **Phase de « conception utilisateur »** (user design). Elle se caractérise par les sessions **JAD** (Joint Application Development ou développement conjoint d'application), qui est un des signes distinctifs du RAD.
3. **Phase de construction**, mêlant les spécifications détaillées et le codage.
4. **Phase de finalisation** et mise en place (en anglais : cutover).

Signalons brièvement deux points essentiels :

- tout d'abord, ces quatre phases constituent un processus itératif. Le « cutover » débouche à nouveau sur une nouvelle définition des besoins, tant que cela s'avère nécessaire ;
- d'autre part, ces phases peuvent, dans une large mesure, être parallélisées.

Les JRP

Le principe de la « définition conjointe des besoins », autrement dit JRP, est simple, mais pourra sembler quelque peu révolutionnaire à certains : sélectionnez les meilleurs éléments, à la fois chez le maître d'œuvre et chez le maître d'ouvrage, et faites les plancher ensemble, dans une salle de réunion sans téléphone et le plus loin possible de toute source de distraction. Ces personnes doivent travailler ensemble, définir ensemble les besoins du système à réaliser.

Lorsque l'auteur parle des personnes-clés, il ne s'agit ni du management seul, ni des « as de la programmation », mais d'une sélection de l'ensemble des profils impliquées dans le projet. Une de ces personnes clés : l'« executive owner », en d'autres termes celui qui paye pour avoir le produit final (ou son représentant attitré). Il doit s'engager à fond dans le processus.

On s'en doutait un peu, mais mieux vaut le répéter que de laisser planer le moindre doute : l'animateur de ces groupes de travail doit être un communicateur hors pair !

Les JAD

L'idée derrière les JAD, qui constituent la substance de la phase de conception, est d'augmenter la productivité globale du développement en réunissant lors d'une même session les utilisateurs et les concepteurs. Encore une fois, son efficacité repose sur deux facteurs clés :

- le facteur humain : dynamique de groupe et suppression des barrières organisationnelles ;
- l'outillage : référentiel puissant et outils de prototypage rapide, les prototypes étant, bien sûr, réutilisables pour le développement du produit final ;

Les « timebox »

Dans la plupart des projets de développement d'application (si ce n'est dans TOUS les projets), il y a des glissements dans les délais. Cela tient à plusieurs causes. Mais on sait en particulier que 80% des fonctionnalités sont réalisées en 20% du temps, et que les 80% du temps restant sont pris par la réalisation des 20% des fonctionnalités à réaliser. En fait, plus le développement semble s'approcher de la fin, moins on avance vite. Cela ressemble à du polissage de pièces. Plus on avance, plus l'abrasif doit être fin. Et plus l'abrasif est fin, moins on avance !

En RAD, la loi est dure, mais simple : le glissement est interdit !
Pour un habitué du développement classique, c'est une révolution ! Comment peut-on interdire tout glissement dans les délais ? Et si je n'ai pas fini de développer, comment faire ?

La solution, ici aussi, est simple. Plutôt que d'autoriser des glissements dans les délais, on va tolérer un glissement dans la réalisation : réduisez les fonctionnalités, mais quoi qu'il arrive, vous finirez à temps.

N'oublions pas que le RAD, et les outils qui sont supposés le supporter, encouragent le développement par affinements successifs. Pour reprendre l'analogie précédente, si on n'a pas le temps d'obtenir un poli parfait, on va se contenter d'une pièce un peu rugueuse (ou même, dans les cas pathologiques cette fois, d'une pièce à peine dégrossie). Mais on ne va pas se donner du temps pour polir encore et encore.

Un timebox s'achève par une revue, qui décidera s'il faut ou non réitérer une deuxième boucle de la spirale de développement.
Le développement itératif ne rend pas seulement le « timeboxing » possible. Il le rend nécessaire. Car le danger des « fonctionnalités rampantes » (je raffine encore, encore et encore) menace de faire dériver un projet à l'infini.
Le management d'un développement de ce style est délicat. Une petite équipe (deux personnes en moyenne) va être mise sous pression pendant une durée fixée (en général 60 jours). Les actions à réaliser alors dans cette boîte temporelle[1] doivent être correctement estimées.

Critique de la méthode

Outre les aspects « outillage », abordés plus loin, la méthode comporte un certain nombre de points faibles.

L'auteur insiste sur la réutilisabilité. Or, pour qu'un composant soit réutilisable, il doit être utilisable dans des contextes différents par des applications différentes, ou par des modules différents. Ceci implique soit une réflexion en amont de sa construction, soit une « généralisation » de ce composant a posteriori pour l'ouvrir à d'autres utilisations que celle pour laquelle il a été conçu.
Si les délais sont très courts, c'est bien ce travail de fond qui sera le premier sacrifié. Si un développeur est pris par le temps, on ne voit pas quelle raison le pousserait à chercher à rendre un composant réutilisable à plus long terme, donc à accomplir une tâche dont il ne verra jamais les fruits.

Il n'y a pas de miracles. La réutilisabilité demande un investissement à long terme. Elle doit être gérée. Elle doit faire partie des objectifs de l'équipe de développement. À moins que le développement consiste en un assemblage de composants préexistants, conçus et développés en dehors du contexte du RAD. Et nous retombons alors dans la problématique de la conception et de la réalisation par objets.

[1] *Je ne sais pas s'il existe un terme français pour « timebox ». Si ce n'est pas le cas, je me permets d'en suggérer un seul : délai-capsule.*

Et les outils?

Après cette brève description de la méthodologie, on peut se poser une question : quels rapports peut-on trouver avec les outils qui, à tort ou à raison, se réclament du RAD? À vrai dire, peu de choses.

Ceux que l'on appelle habituellement les « outils RAD » ont en commun un certain nombre de caractéristiques :

- ils sont le plus souvent « orientés objet », ou se présentent comme tels ;
- ils comportent un outil de construction d'IHM, ou de génération automatique de l'IHM ;
- le langage utilisé pour la programmation est un langage de haut niveau, ou un langage dit « de quatrième génération ».

Les trois caractéristiques précédentes encouragent un cycle de développement en spirale. La gestion des liens logiques entre les objets de l'interface graphique et les variables utilisées dans la programmation est entièrement automatisée. L'outil encourage un dialogue permanent avec l'utilisateur final, grâce aux possibilités de maquettage et de prototypage.

Le maquettage et le prototypage

Les outils de développement de nouvelle génération sont, du point de vue du maquettage/prototypage, bien plus puissants que tous ceux que l'on pouvait trouver sur le marché lorsque le livre de James Martin est paru.

Si les « outils RAD » méritent leur qualificatif de « RAD », c'est bien sur cet aspect-là. Les outils « modernes » permettent de construire quelques dessins d'écrans en une heure.
De là à construire l'interface homme-machine en temps réel, en réunissant informaticiens et utilisateurs dans une même salle pendant quelques jours, il n'y a qu'un pas. Si les recommandations de James Martin sont respectées, en particulier en ce qui concerne les aspects humains, ce pas peut être franchi.

L'orientation objet

James Martin ne préconise pas l'orientation objet dans son ouvrage[2]. Tout au long de la description de la méthode est maintenue la séparation entre données et traitements, ce qui va bien entendu à l'encontre de l'approche par objets (principe d'encapsulation).
Par ailleurs, James Martin précise que le RAD n'a de sens que par ce qu'il s'inscrit dans un cadre méthodologique plus large, constitué par l'Information Engineering (qui n'est pas une méthode objet).
Cependant, l'orientation objet est actuellement l'approche la plus prometteuse pour favoriser la réutilisation.

Ce que serait un outil RAD idéal

Lorsque James Martin parle d'outils, il cite IEF, Cohesion et AD/Cycle (nous sommes en 1990). Il ne s'agit pas là d'outils que l'on qualifierait spontanément de « RAD ». Néanmoins, certains aspects du RAD sont mieux traités par ces outils que par les outils RAD des années 95. En particulier, ils sont intégrés, ou tendent vers l'intégration totale, autour d'un référentiel puissant.

Quel serait alors le portrait de l'outil RAD idéal ? On peut tenter de l'esquisser en faisant la liste des fonctionnalités nécessaires à la mise en œuvre de la méthode:

- référentiel puissant, prenant en compte la totalité du cycle de vie ;
- ce référentiel permet un travail en équipe, souple, efficace, non bureaucratique ;
- le référentiel doit être facile à utiliser, même par des personnes y faisant rarement appel (directeur de projet, consultants externes, utilisateur non informaticien) ;

---

[2] *Le terme d'objet est effectivement utilisé à quelques reprises, mais dans le sens d'objets du référentiel. Le fait que le référentiel soit orienté objet n'implique en aucun cas que la conception de l'application se fasse par objets. Par ailleurs, James Martin est très critique vis-à-vis des techniques objet.*

- la gestion de projet devrait être intégrée à l'outil. Elle devrait prendre en compte la gestion du temps, l'enchaînement des étapes, le parallélisme entre étapes, l'itération du cycle de développement, la communication entre individus ;
- l'outil devrait favoriser la réutilisation des composants.

Parmi les outils actuels, combien y en a-t-il qui permettent de répondre « oui » à tous ces critères ? Chacun a son idée sur le sujet.

Personnellement, je me permets les remarques suivantes :

- Seule l'orientation objet permet d'obtenir un niveau industriel de réutilisabilité. La réutilisation par « bibliothèques de routines » a montré ses limites. Quand à la réutilisation par « copier-coller », elle nous mène droit à la catastrophe organisationnelle. Cependant, n'oublions jamais que la réutilisation doit résulter d'une volonté stratégique, et qu'elle a un coût.
- Le référentiel puissant et « intelligent » préconisé par James Martin est introuvable sur la plupart des outils de développement rapide.
- James Martin insiste sur les aspects humains du développement. On ferme parfois les yeux sur une évidence : aucun outil ne réglera les problèmes humains. Même si certains outils, ou l'absence de certains outils, ne peuvent que les envenimer.

Adéquation des outils à la méthode

Le tableau suivant résume les différences entre les caractéristiques du RAD, telles qu'elles sont décrites par James Martin, et la prise en compte de ces caractéristiques en termes de fonctionnalités correspondantes dans des outils RAD.

Aspects	**Méthode RAD de James Martin**	**« Outils RAD »**	**Remarques**
Personnes	De très bon niveau, très motivées, dynamiques ; jouent un rôle important dans le projet	Pas de prise en compte	
Référentiel	Puissant et « intelligent » (fait appel à des techniques d'intelligence artificielle)	Disposent souvent d'un référentiel, mais en général bien moins puissant que celui préconisé	Le référentiel est un point clé pour James Martin
Conception données / traitements	Entité-relation et fonctionnelle ; séparation données / traitements	Entité-relation parfois, mais, de plus en plus souvent, outils orientés objet	
Outils	Intégrés (I-CASE) prenant en compte tous les aspects	Rarement intégrés ; spécialisés sur une partie du cycle	
Cycle de vie	4 phases, elles-mêmes décomposées en étapes ; démarche itérative	Pas de référence au cycle de vie, mais la démarche itérative est favorisée	
Gestion du temps	Timeboxing (blocage des étapes dans le temps)	Pas de référence au temps	
Réutilisation	Importante, mais n'est pas décrite clairement	Possible	L'orientation objet, souvent favorisée par les « outils RAD » permet une réutilisation effective
Construction de l'IHM	Importante	En général, très bien prise en compte	L'aspect le plus positif des outils modernes

Conclusion

Les équipes décrites par James Martin s'appellent SWAT : Skilled With Advanced Tools. En français : compétentes et armées d'outils de pointe. On parle souvent des outils RAD. Et si on parlait un peu plus des personnes ?

De bons outils, nous l'admettons volontiers, sont indispensables. Rares, sans doute. Chers aussi, mais il suffit d'y mettre le prix pour les avoir. Le prix de six mois, un an de salaire par poste. Exorbitant ?

Ce qui est bien plus rare, cher, précieux, c'est une équipe de projet capable de s'embarquer dans une telle aventure. Ce qui est d'une valeur inestimable, c'est un chef de projet capable à la fois de communiquer son enthousiasme, d'animer une équipe, de comprendre le besoin du client, et de secouer sa propre organisation pour y puiser les ressources qui lui manquent.

Car le RAD, à bien lire l'auteur du livre du même nom, c'est avant tout une affaire de personnes et d'organisation. La forme des bureaux et de la salle de réunion y joue un rôle aussi important que la structure du référentiel. La motivation des développeurs y joue un rôle aussi primordial que le générateur d'interface. Vouloir faire du RAD et n'y voir que des outils, c'est vouloir courir un 400 mètres à cloche-pied ! ▲

Yves Constantinidis

Bibliographie

James Martin, Rapid Application Development, Macmillan, 1991
Le livre de référence en matière de RAD, par le père de la méthode. En anglais

Jean Hugues, Bernard Leblanc, Chantal Morley
RAD, Une méthode pour développer plus vite, InterEditions
Un ouvrage clair et concis. En français

Bertrand Meyer, Object success; A manager's guide to object orientation, its impact on the corporation and its use for reengineering the software process, Prentice Hall
Ouvrage intéressant pour ceux qui s'intéressent à l'orientation objet. Ne traite pas spécifiquement du RAD. Accessible aux non-spécialistes. On y trouve en particulier une réflexion intéressante sur la réutilisation, et sur les aspects humains de la conduite de projet, en particulier dans le cas des développements par objets.

Histoire d'une révision

La norme ISO 9000-3

Connaissez-vous la norme ISO 9000-3 (prononcer neuf mille tiret trois pour éviter la confusion avec la norme ISO 9003) ?

La norme ISO 9000-3 date de 1991. Elle a été transposée en norme européenne en 1993 et a pris de ce fait le statut de norme française homologuée en novembre 1993. Il s'agit de la troisième partie de la norme ISO 9000 qui contient un ensemble de lignes directrices pour l'application des normes de références ISO 9001, 9002 et 9003. La norme ISO 9000-3 n'est donc pas elle-même une norme de référence, pouvant servir de base à une opération d'audit, mais une norme explicative. Elle s'intitule précisément : « Lignes directrices pour l'application de l'ISO 9001 au développement, à la mise à disposition et à la maintenance du logiciel ». Elle traite à l'origine des situations de réalisation de logiciel spécifique répondant à des spécifications établies par le client.

Cette norme, dans sa version 1993, souffrait de défauts de jeunesse multiples, dont le principal et non des moindres était de fournir à un métier en constante évolution une interprétation figée. Elle s'appuyait en particulier sur un modèle de cycle de vie prédéfini, de type cycle en V, certes hautement respectable mais qui a une tendance fâcheuse à se croire seul au monde. Elle maintenait une confusion terminologique sur le mot « conception » utilisé tantôt au sens de phase du cycle méthodologique, tantôt au sens ISO 9001 de conception avant mise en production.

L'articulation entre plan qualité et plan de développement y était particulièrement confuse, chacun de ces deux plans pouvant faire appel à l'autre.

De plus, les exigences formulées étaient insuffisamment modulées en fonction de la taille et du niveau de risque des projets et ne rendaient la norme applicable qu'aux très gros projets.

Enfin, j'allais oublier un point important, le plan de la norme 9000-3 était absolument différent de celui de la norme ISO 9001, ce qui obligeait à la consultation de tables de correspondance (cross-reference, dites-vous ?) fort discutables.

L'ensemble de ces points a suscité dès l'origine de la norme, de multiples critiques, dont celles du Canada.

Depuis 1991, l'ISO 9000-3 avait pris deux voies contrastées suivant les pays utilisateurs : Certains pays, comme la France, l'utilisaient comme une norme explicative fournissant une interprétation de l'ISO 9001, mais ne s'y substituant pas en matière de référentiel de certification. D'autres, tels les anglais, avaient bâti autour de cette norme un système de certification, le TickIT, en la prenant comme norme de référence.

Entre temps, pour compliquer encore la tâche des utilisateurs, la norme ISO 12207 était venue couvrir le champ de la « typologie des processus du cycle de vie du logiciel » en fournissant un cadre beaucoup plus ouvert que celui de la 9000-3 pour la description des processus.

À qui se fier ? A qui se référer ? Aux deux extrêmes on pouvait se croire obligé d'appliquer à tous ses projets un cycle de vie unique, déposé dans la bible méthodologique de l'entreprise, ou autorisé à construire au cas par cas le cycle le mieux adapté au contexte (si vous n'êtes pas vous-même un adepte de la qualité tyrannique et sadomasochiste, vous pouvez deviner sans peine où va ma préférence).

La publication de la version 1994 de l'ISO 9001 était un prétexte supplémentaire pour reprendre le texte de l'ISO 9000-3 et le mettre au moins en conformité avec celui de sa norme mère, l'ISO 9001. L'ISO 9000-3 est donc rentrée depuis deux ans dans le parcours complexe des opérations de révision.

Je vous épargnerai le détail des réunions internationales de l'ISO. Je n'ai pour ma part que modestement participé au groupe d'experts AFNOR (GE9) chargé de la préparation de la position française et je n'ai pas eu la chance de faire le voyage à Brisbane, ni à Londres[1]. Les débats furent fort houleux, car au départ les anglo-saxons s'attachaient de façon fort conservatrice à leur version d'origine et refusaient absolument de toucher au plan de la norme ou de rajouter quelque référence que ce soit à l'ISO 12207.

Où en sommes nous aujourd'hui ?

La version, actuellement soumise dans chaque pays à enquête probatoire (en France la procédure d'instruction de l'AFNOR était ouverte jusqu'au 5 avril dernier), est devenue beaucoup plus lisible que la précédente : le plan adopté est calqué sur celui de l'ISO 9001. On trouve donc pour chacun des 20 paragraphes d'exigences de l'ISO 9001, un paragraphe de recommandation de l'ISO 9000-3 qui reprend en encadré le texte intégral de l'ISO 9001 en y rajoutant l'interprétation particulière recommandée pour le logiciel lorsqu'il en existe une, ainsi que des références à la norme ISO 12207.

Dernière bataille, non encore gagnée, l'introduction d'un tableau de référence croisée entre ISO 9000-3 et ISO 12207, demandée par la France rencontre l'opposition farouche des Japonais.

La nouvelle norme devrait voir le jour international d'ici la fin de l'année et apporter un éclairage renouvelé à la qualité dans les métiers du développement du logiciel. Souhaitons lui la bienvenue.

Afin de l'utiliser de la façon la plus efficace dans nos relations client-fournisseur, nous devrons cependant continuer à l'utiliser comme un guide et revenir, chaque fois que nous aurons un doute à la norme de référence qui reste toujours l'ISO 9001.

Vieux proverbe d'actualité : Il vaut mieux se fier à Dieu qu'à ses saints (auteur inconnu). ▲

Martine Otter

[1] *Contrairement à d'autres pays, aucune subvention n'est prévue en France pour la participation aux travaux de normalisation internationale. Seules les plus grosses entreprises peuvent déléguer des experts à leur frais.*

Le télétravail

Présentation d'un ouvrage en préparation

L'ADELI prépare actuellement un ouvrage sur les motifs, voies et moyens pour mettre en place un système d'information où interviennent des télétravailleurs. Cette étude pose les bases de la nouvelle donne engendrée par le télétravail et analyse les impacts, sociaux et professionnels, sur les télétravailleurs, voire sur le système d'information de l'entreprise et son organisation.

Le télétravail est une activité multiforme chez la plupart des personnes qui en parlent. Ce n'est pas ceux qui en parlent le plus qui y goûtent ou la pratiquent, voire la mettent en pratique.

Au fil de l'histoire

Le coureur de Marathon qui porte un message de victoire n'est pas un télétravailleur, tout au plus un « facteur » délivrant un message.

Nos tours de guet contre les invasions nordiques ou mauresques étaient destinées à transmettre des messages par des moyens et des signaux appropriés. Les guetteurs étaient-ils pour autant des télétravailleurs ? Oui, mais au seul sens "étymologique" du terme. En effet, leur travail consistait, entre autres activités de guetteur, à transmettre à distance le résultat de leur travail, ou à prendre des consignes à distance. Ces temps ne sont pas révolus dès lors que nos gardes forestiers signalent des risques d'incendie par signal téléphonique ou par radio (voire tout autre moyen aussi ancestral que les signaux Chappe ou tout autre mode de transmission non sonore).

Les représentants de commerce qui parcouraient à cheval nos campagnes, ou ceux qui jusqu'à un proche passé utilisaient d'autres moyens de locomotion, pour aller glaner quelques commandes avant de les rapporter à l'unité de production ou à celle de distribution en gros, ont utilisé par la suite le téléphone ou la radio pour transmettre le fruit de leur travail. Sont-ils pour autant des télétravailleurs ? Oui, mais au seul sens "étymologique" du terme. En effet, leur travail consistait (et consiste encore), entre autres activités de commercial, à transmettre à distance le résultat de leur travail, ou à prendre des consignes à distance.

Un pigiste photographique qui expédiait quelques photos par un système « bélino » ou un journaliste qui envoie par télécopie ou tout autre moyen, un article à un journal... Sont-ils pour autant des télétravailleurs ?

Deux types de populations peuvent être définis. Il y a les télétravailleurs et ceux qui ont une téléactivité. La répartition est réalisée, dans un premier temps, par plusieurs niveaux de restriction. Le premier type concerne les entreprises mettant en place un (cas par exemple d'une personne hospitalisée, ou d'un secrétariat spécifique), ou plusieurs ateliers de télétravail. Le deuxième type concerne les télétravailleurs eux-mêmes relevant de ces entreprises.

Premier niveau de restriction

Les opérateurs de signaux, cités ci-dessus, ne sont pas des télétravailleurs. Cette restriction ne retiendra comme télétravailleurs que ceux qui mettent en œuvre, aux deux bouts de la chaîne de traitement de l'information un outil informatique quelque soit le mode de transmission entre ces deux extrêmes. Cela ne nie pas la téléactivité de ces opérateurs en matière de télétransmission d'un signal.

Deuxième niveau de restriction

Ne peuvent être retenus comme télétravailleurs tous ceux dont la fonction de télétransmission se limite à l'émission ou la réception d'un message quelqu'en soit l'importance volumétrique et économique, même si aux deux bouts de la chaîne de traitement de l'information il y a un outil informatique. Ceci est notamment le cas des commerciaux qui émettent des rapports ou adressent des commandes, tout en recevant des consignes. Ceci concerne généralement la quasi-totalité des "télétravailleurs nomades". Cela ne nie pas la téléactivité de ces "télétravailleurs nomades".

Troisième niveau de restriction

Ne peuvent être retenus comme télétravailleurs que ceux qui sont liés à une personne morale par un contrat de travail et dont le travail ne se situe pas "in situ" par rapport à l'entreprise, mais dont l'activité serait (quasiment) la même si physiquement ils se trouvaient en entreprise.

Les études de faisabilité du télétravail du document à paraître sont celles qui ont trait à la problématique de l'entreprise qui veut développer ce mode de fonctionnement pour tout ou partie de ses effectifs non directement en production « sur site ». Cette entreprise devra adapter les aspects mentionnés par cette étude à ses besoins spécifiques et en fonction de son métier, ainsi que de sa localisation géographique et de l'état de maturité technique de l'architecture de son système d'information.

Les difficultés de définition et les risques d'interprétation d'un néologisme

Le néologisme "télétravail" se comprend aisément par son radical "télé" et son signifiant "travail". Cependant, par abus de langage, ce mot recouvre plusieurs sens. Les définitions les plus courtes qui en aient été trouvées dans le Larousse sont :

- Travail : effort, application pour faire quelque chose
- Travailler : faire un ouvrage, exercer un métier

Adjoindre le radical "télé" fait penser qu'il s'agit de quelque chose réalisée "à distance". Dans la réalité le radical "télé" inclut obligatoirement un système de télécommunication. L'abus de langage vient du fait :

- non seulement qu'il est omis de préciser que ce système de télécommunication, implique obligatoirement des moyens informatiques qui utilisent un système filaire ou hertzien pour communiquer entre un point d'émission et un point de réception ;
- mais aussi et surtout, qu'il s'agit d'un mode de fonctionnement entre une entreprise et ses collaborateurs directs (télétravail) ou certains de ses sous-traitants (téléactivité).

Le "télétravail" revient ainsi non seulement à qualifier les moyens mis à la disposition d'un télétravailleur mais aussi à qualifier son mode de fonctionnement et d'action par rapport à son donneur d'ordre : l'employeur ou le client.

De nombreuses personnes omettent d'associer les termes de télétravail et de télétravailleur à la notion juridique qui détermine et précise l'ensemble des liens contractuels entre le téléacteur et son donneur d'ordre, sans pour autant modifier la finalité de son travail dont seul le lieu de réalisation diffère.

De cette "relocalisation" naît un autre abus de langage qui consiste à considérer que le télétravailleur est un travailleur à domicile. Ce peut être vrai, comme faux. Le concept de télétravail revient à accomplir hors des locaux de l'entreprise (mais grâce à un système de télécommunication doté à chaque extrémité d'un système informatique), une activité qui se réalise "normalement" en entreprise à partir d'un système informatique (intégré ou non à un réseau télématique).

Une secrétaire qui "télétravaillerait" à domicile (ou ailleurs) en emportant du travail à faire sur un système informatique et qui utiliserait une disquette pour restituer son travail serait une travailleuse à domicile et ne saurait pas être reconnue comme une "télétravailleuse".

La même secrétaire qui utiliserait un système de télécommunication doté d'un terminal informatique pour collectionner les éléments constitutifs de son travail (fichiers, tarifs, rapports, ...) domiciliés sur le système informatique chez son donneur d'ordre et qui transmettrait ensuite les résultats de son travail par l'émission d'un fichier représentatif de ce travail serait reconnue comme une "télétravailleuse" ou comme ayant une téléactivité.

De nombreuses publications existent sur le télétravail. Elles ont pour objectif soit de vanter des succès, soit de camoufler certains échecs et le plus souvent les difficultés de mise en œuvre du télétravail au sein d'une entreprise.

Certains rapports et parlementaires y ont vu un processus d'aménagement du territoire ainsi que la création d'emplois directs ou induits.

Tentative de définitions

Le concept de télétravail est polysémique. Non seulement il correspond à une forme de travail fait « hors les murs » de l'entreprise, mais aussi au résultat du travail lui-même. Les termes doivent être « normalisés » afin de ne pas faire en permanence la confusion entre tel ou tel sens possible.

Il n'existe aucune définition, d'ores et déjà normalisée, des termes mentionnés ci-après. L'ADELI ne peut prétendre à ce pouvoir normatif. Les définitions ci-dessous sont celles retenues dans l'ouvrage qui sera prochainement publié.

Téléactivité

Toute forme de travail ou service accompli à distance, faisant intervenir un système de télécommunication entre deux points (l'un et l'autre étant à tour de rôle, ou non, : émetteur ou récepteur).
Ce terme est communément utilisé en tant que représentant et le télétravail et le téléservice.

Téléacteur

Toute personne physique ayant une téléactivité (cas typique du pigiste).

Téléservice

Service accompli à distance, faisant intervenir un système de télécommunication entre deux personnes (morales ou physiques) ayant établi entre-elles un contrat d'entreprise.

Télétravail

Travail accompli à distance, faisant intervenir un système de télécommunication entre deux personnes (l'une morale et l'autre physique) ayant établi entre-elles un lien de subordination dans le cadre d'un contrat de travail.

Pluriactivité

Travail ou service accompli par une personne (morale ou physique) pour le compte de plusieurs personnes morales, ayant établi entre-elles un contrat de prestation de service et/ou un lien de subordination dans le cadre d'un contrat de travail.

Cette étude concerne le télétravail et la situation matérielle et juridique du télétravailleur. ▲

Jean-Marc Bost

Le réseau sémantique universel (2ème partie)

L'intégration de quelques concepts permet d'étendre considérablement les possibilités de modélisation. Voici la suite de l'analyse de cette intégration.
L'auteur vous renouvelle ici ses excuses pour la densité et la non-exhaustivité de cette étude (et pour les égratignures faites à MERISE), mais il s'agit d'une suite d'articles à considérer comme un condensé – et un résumé – d'une étude plus complète qui sera publiée dans un proche avenir.

Rappel de la 1ère partie

Nous avons vu dans la première partie de cet article (La Lettre n° 26) qu'il était possible d'utiliser un seul et unique symbolisme pour modéliser pratiquement n'importe quel système, ceci ayant été présenté avec les sept systèmes suivants :

- Modèle Conceptuel de Données (MCD) de Merise ;
- Modèle Conceptuel de Traitements (MCT) de Merise ;
- Système à Base de Connaissance (SBC) ;
- Tableur (considéré comme un croisement entre une Base de Données et un SBC) ;
- Système NeuroMimétique (SNM) ;
- Modèle de Traitements en Temps Réel (TTR) ;
- Programmation Orientée Objets (POO).

Bien que le symbolisme utilisé soit - à la base - celui des MCD de Merise, il a été sensiblement élargi, prenant en compte les concepts suivants :

- **Objets** (modèles, populations, occurrences, singularités) ;
- **Attributs** (propriétés simples, faits, tableaux, agrégats, méthodes ; appartenant à des objets, des associations, ou des actions) ;
- **Associations** (modèles, populations, particularités, héritages) ;
- **Actions** (les traitements) ;
- **Prédicats** (attributs contribuant à un héritage par spécialisation ou au déclenchement d'actions) ;
- **Pattes** (reliant entre eux objets, associations, actions, prédicats).

Notre objectif, maintenant, est d'élaborer un métamodèle - c'est-à-dire modéliser les modèles - qui sera capable de permettre la définition de tout modèle. Ce métamodèle jouera tout à la fois les rôles de dictionnaire, générateur, concepteur, etc.

À partir de ce métamodèle, l'étape suivante consistera à élaborer un modèle physique qui, implanté en tant que SGBD entité-relation, permettra de concevoir, prototyper et réaliser n'importe quelle application sans programmation.

Définition

Pour bien comprendre l'objectif actuel, il est indispensable de bien définir ce qu'est un métamodèle.

Dans un modèle, on trouve par exemple des objets, contenant des attributs. Dans un métamodèle, on trouvera la description d'un objet-type (modèle d'objet), stockée sous forme d'un ensemble de composants (entités "objet" et "attributs", reliées par des associations "posséder").

En résumé, un métamodèle est un modèle dans lequel sont définies des entités permettant de décrire un modèle, et ce selon le même mécanisme[1 et 2].

Enrichissement du symbolisme

L'élaboration d'un métamodèle prenant en compte des concepts très élargis, il est nécessaire et/ou utile de légèrement enrichir le symbolisme défini dans la première partie de ce dossier.

Héritage simple et multiple

L'héritage simple n'ayant été précédemment que survolé, et l'héritage multiple non abordé, nous allons maintenant en faire une analyse plus complète, car indispensable pour définir notre métamodèle.
Dans la pratique, ces notions ne figurent que dans deux des systèmes étudiés : le MCD et la POO, et ce avec des implémentations différentes :

- Dans un MCD, si un objet conceptuel hérite d'un autre objet conceptuel, on a toujours affaire à un même individu. De même, un objet conceptuel ne peut hériter de plusieurs objets que si ces derniers ont une même base sémantique, c'est-à-dire qu'ils héritent tous, à quelque degré que ce soit, d'un objet de base unique. Par exemple, un *Enseignant-Chercheur* pourra hériter des objets *Enseignant* et *Chercheur*, car ceux-ci héritent tous deux (héritage simple) de l'objet *Personne*. Dans tous les cas, ces quatre objets conceptuels ne constituent qu'un seul et unique individu, mais avec des vues différentes. On a bien affaire ici à un **héritage sémantique**.
- A contrario, en POO, une classe peut être constituée par héritage d'une ou plusieurs autres classes, même si celles-ci n'ont aucun rapport sémantique entre elles. Ceci n'est possible que parce qu'on ne définit en fait que des modèles, et que les objets générés seront toujours physiquement distincts. Par exemple, si on définit une classe C qui hérite des deux classes A et B, ceci permettra à un objet C d'être vu comme un objet A ou comme un objet B, mais il n'existera aucune occurrence de A ni de B pouvant être vue comme un objet C. Nous appellerons ce type d'héritage : **héritage génétique**, car l'objet héritier bénéficie des attributs de son ou ses parents, et reste un individu distinct. C'est d'ailleurs le cas entre une occurrence d'objet et son modèle.

Il s'agit donc bien de deux concepts distincts, et nous verrons plus loin qu'ils coexistent dans notre modélisation unifiée, sachant que l'héritage Merise concerne les objets et les individus, et l'héritage POO les modèles, notamment au niveau des agrégats.

Il est donc nécessaire de pouvoir les différencier au premier coup d'œil, et donc d'enrichir le symbolisme déjà mis en place :

- Un héritage sémantique reste symbolisé par une association (1,1)-(0,1) représentée par une flèche à double tracé (), orientée vers l'objet parent. En cas d'héritage multiple, il y a une flèche par parent.
- Un héritage génétique se différenciera de l'héritage sémantique par l'utilisation d'une flèche à double tracé également, mais doublement barré (), pour montrer que les individus sont différents, les cardinalités de ce type de lien étant également (1,1)-(0,1).

La contrainte d'égalité

Merise propose indifféremment l'usage du symbole –(s)– ou –(=)– pour symboliser une *contrainte de simultanéité*, c'est-à-dire présence simultanée d'au moins une occurrence de chacune des entités concernées. L'expérience a montré que le second symbole pouvait prêter à confusion.

Nous proposons donc de différencier la signification de ce symbole –(=)– : nous l'utiliserons désormais pour matérialiser une *contrainte d'égalité*, c'est-à-dire impliquant d'avoir de part et d'autre le même nombre d'occurrences (désolé pour les puristes de Merise qui l'ont choisi au lieu du premier).

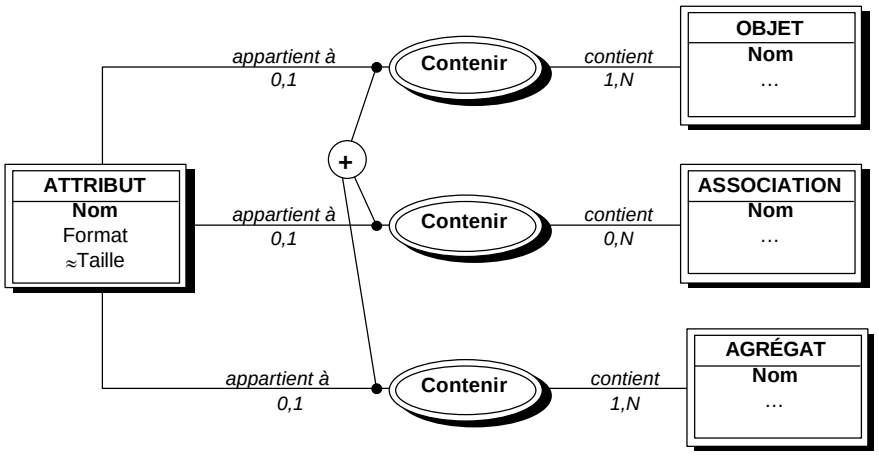
---

[1] *Dans le respect de Merise, qui est une méthode réflexive, c'est-à-dire qui peut se modéliser elle-même.*

[2] *…et je suis reparti pour vous submerger de notes de bas de page (mais c'est ça ou les apartés entre parenthèses) !*

Les pattes alternatives

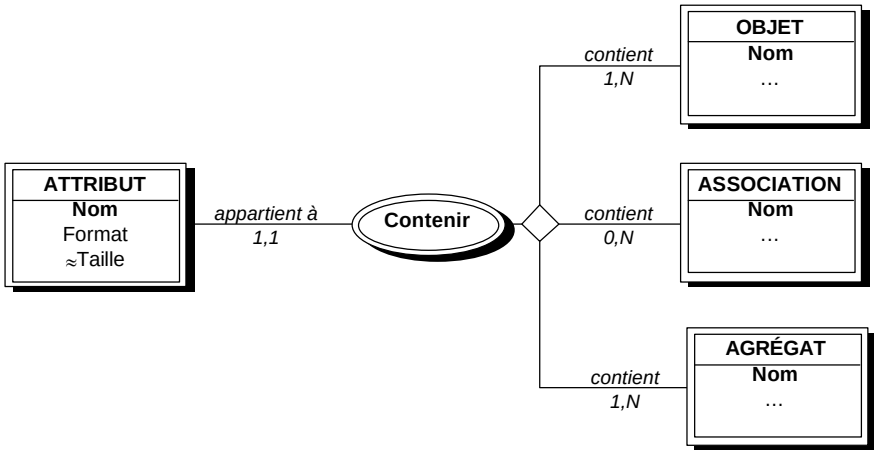
Sous ce nom "barbare" se cache une notion très simple, que nous allons illustrer immédiatement. Par exemple, sachant qu'un attribut appartient soit à un et un seul objet, soit à une et une seule association, soit à un et un seul agrégat, nous devrions symboliser cet ensemble comme ceci :



Mais en fait, les trois associations :

- sont sémantiquement équivalentes pour l'attribut ;
- contiendront des informations semblables ;
- une et une seule d'entre elles existera.

Il semblerait donc intéressant de mieux formaliser cet état de choses, et d'utiliser la représentation ci-dessous :



Ce symbolisme permet de mettre la véritable cardinalité du lien d'appartenance de l'attribut (1,1), de ne modéliser qu'une seule association, et d'éliminer le besoin d'une contrainte ensembliste explicite (la cardinalité (1,1) implique qu'un seul des trois objets de droite sera relié à l'attribut).

Bien sûr, il ne s'agit ici que d'une facilité de représentation, et il est toujours nécessaire de définir chacune des associations possibles, car il s'agit bien d'autant d'entités distinctes.

Le réseau récursif

Certaines associations relient les mêmes types d'objets, soit directement (OBJET hérite de OBJET), soit indirectement (ATTRIBUT appartient à TYPE, TYPE définit ATTRIBUT ...). Dans certains cas, il est impératif qu'un parcours au travers de ces associations n'aboutisse jamais sur un objet déjà rencontré dans ce même parcours : si A hérite de B, B hérite de C, en aucun cas C ne peut hériter de

A[3], car nous avons en fait une arborescence. La simple contrainte d'exclusion –(x)– entre les pattes n'interdisant que d'avoir le même individu de part et d'autre, il faut pouvoir être plus explicite.

Un second cas de figure est encore plus restrictif. En effet, si - par exemple, pour un héritage génétique - A hérite de B et C, C hérite de D, B hérite de E, il serait gênant que D hérite également de E, car on se retrouverait avec les mêmes attributs en double dans A[4]. Il faut donc dans ce cas interdire de pouvoir retrouver le même objet, même par un autre chemin, c'est-à-dire qu'il ne peut exister qu'un seul chemin pour aller de X à Y utilisant cette association, quel que soit le nombre d'étapes.

Nous utiliserons donc la notation –(x→)– (flèche simple) pour expliciter l'interdiction de boucler sur un chemin en allant toujours dans le même sens, et la notation –(x↔)– (flèche double) pour expliciter l'interdiction de revenir sur le même objet quel que soit le chemin parcouru.

Éventuellement, le même type de formalisme pourra servir avec d'autres contraintes que l'exclusion (notamment l'inclusion –(i)–).

Les clones

Il ne s'agit plus ici d'enrichir réellement le symbolisme, mais d'améliorer la lisibilité : au-delà d'un certain degré de complexité, on en vient à avoir des pattes qui se croisent et s'entrecroisent, transformant le schéma en sac de nœuds. Il serait donc intéressant, pour éviter cela, de prévoir une possibilité de duplication de certains objets.

Attention : il ne s'agit pas d'augmenter le nombre des objets, mais uniquement de les faire apparaître plusieurs fois dans le modèle. Comme il est indispensable que le lecteur du modèle soit informé de la chose, et ne s'étonne pas de voir apparemment plusieurs objets porter le même nom, nous ajouterons à tout objet dupliqué un symbole[5] :
- ❑ dans l'original, pour signaler qu'il a été dupliqué ;
- ❑ dans chaque copie, pour signaler qu'il s'agit d'un duplicata.

Pour être plus lisible encore, il est inutile de répéter les attributs d'un objet dans un clone : le symbole de copie peut aussi signifier : pour plus d'information, voir l'original.

Le métamodèle

Tout le formalisme nécessaire étant maintenant défini, nous pouvons commencer à élaborer ce modèle étape par étape, en prenant les différents types d'entités une par une et en les raccrochant à ce qui aura été défini.

Le lecteur observateur remarquera dans ce qui suit qu'en fait, parmi les différents systèmes analysés dans la première partie de ce dossier, les concepts modélisés proviennent en quasi-totalité de trois d'entre eux seulement : le MCD Merise, le Système à Base de Connaissance (ou système-expert), et la Programmation Orientée Objet.

Les attributs

Un attribut s'identifie par son nom, et il a un format (entier, flottant, caractère, binaire, blob[6],...) et éventuellement une taille. S'il est de type agrégat, il contient d'autres attributs ; on a donc une relation récursive **contenir**, dont les pattes sont nommées *contient* et *contenu dans*. Dans cette association, on placera des attributs tels que le nom de l'attribut dans l'agrégat, le nombre d'occurrences et le numéro de la première occurrence si on inclut un tableau, la facultativité de l'attribut.

---

[3] *Nous parlons bien entendu d'héritage sémantique ou génétique, et pas d'héritage testamentaire (quoique même dans ce cas... si A et B se désignent mutuellement comme héritiers, un seul des deux pourra hériter au final...).*

[4] *Ceci est autorisé en C++, mais implique une gymnastique pour définir si les attributs en double sont sémantiquement les mêmes ou pas, et dans ce dernier cas il faut procéder à un renommage. Bjarne Stroustrup (créateur du C++) lui-même déconseille ceci.*

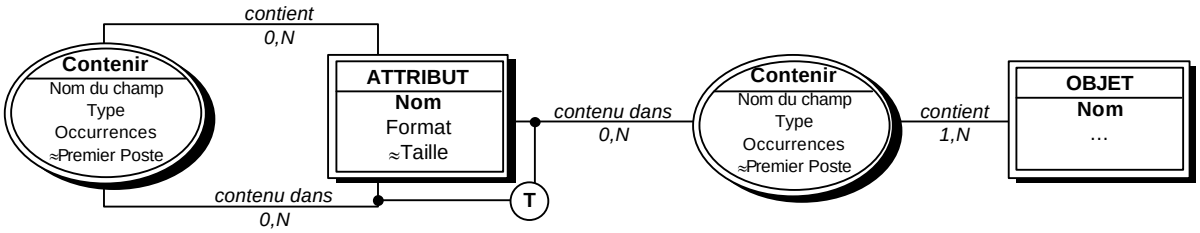
[5] *Certains AGL utilisent déjà cette technique, et pour les mêmes raisons (mais pas nécessairement le même symbolisme, ce qui est secondaire).*

[6] *BLOB = Binary Large Object, pouvant être un texte, un son, une image, etc.*

Un objet pouvant être considéré - du point de vue des attributs - comme un agrégat, on trouvera également ces attributs dans l'association objet-attribut.

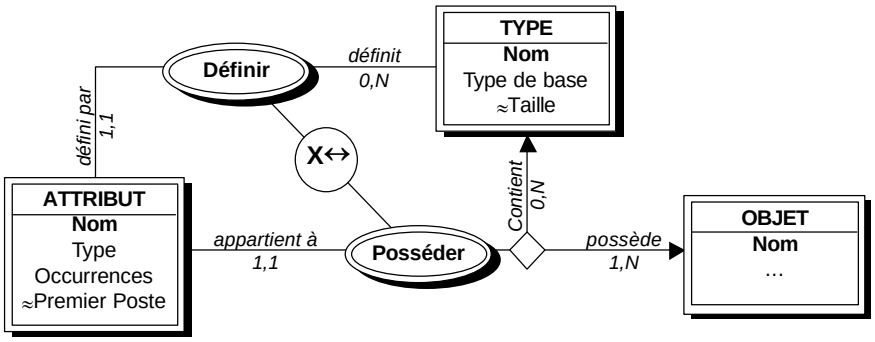
Un attribut n'ayant d'utilité que s'il est utilisé quelque part, il doit obligatoirement être associé à au moins un agrégat ou un objet (ou une association, nous compléterons plus loin).

Il semble donc que l'amorce du métamodèle devrait être :



Mais si on regarde d'un peu plus près, on s'aperçoit qu'on a déjà tout faux : logiquement, rien n'empêche d'utiliser dans un objet ou un agrégat deux (ou plus) attributs semblables[7] (exemple : *adresse de livraison* et *adresse de facturation*), ce que ne permet pas le modèle ci-dessus.

L'attribut *Nom du champ* est en fait un identifiant relatif [8]de l'attribut par rapport à son contenant. C'est pourquoi nous devons distinguer les notions d'attribut "physique" et d'attribut "type" (équivalent sémantique du *typedef*, *struct* ou *class* du C++, ou du *type* ou *class* du Pascal). Nous appellerons tout simplement celui-ci **TYPE**, et nous modéliserons l'amorce du métamodèle, finalement plus simple que le précédent, comme ceci :



Cette première ébauche appelle, toujours au niveau des attributs, plusieurs compléments.

a) Tout d'abord, dans un grand nombre de cas, un attribut simple (ou un fait) ne peut contenir n'importe quoi : son contenu fait partie d'une liste de valeurs et/ou exclut certaines valeurs, est compris dans une ou plusieurs fourchettes, doit avoir un format précis (masque), etc. Nous définirons donc un autre type d'objet pour le métamodèle : la **VALEUR**.

   Tout individu **VALEUR** contient un ensemble de données permettant de valider le contenu d'un attribut, et n'a (en principe) d'utilité que s'il est effectivement associé à au moins un attribut (association **Vérifier**, dont les pattes sont nommées *vérifie* et *vérifié par*).

   Nous verrons en détail la définition de cette entité lors de la mise en œuvre pratique[9].

b) Ensuite, il est utile d'introduire ici la notion d'héritage génétique entre types (un type agrégat *A* contenant un autre agrégat *B* nécessitera de référencer les attributs de ce dernier en *A.B.x*, alors qu'un agrégat héritant d'un autre agrégat disposera directement d'attributs identiques : *A.x*).

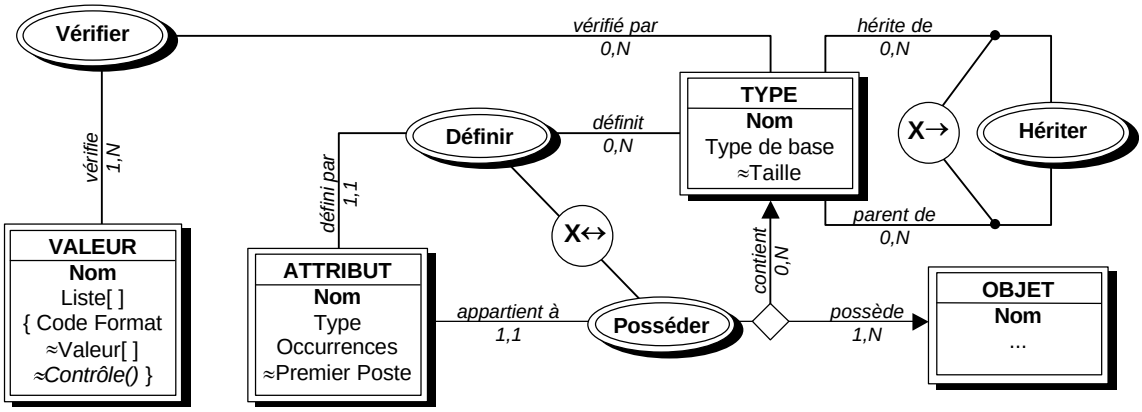
---

[7] *"Semblable est ici utilisé dans son sens mathématique : deux triangles semblables ont exactement la même forme, mais peuvent être de tailles différentes, donc non identiques). Deux attributs semblables ont les mêmes caractéristiques, mais pas le même usage.*

[8] *De par leur implémentation, certains AGL du commerce ne permettent pas cette identification relative, tant pour les atributs que pour les associations, obligeant ainsi à donner des noms différents à des concepts parfois semblables. Nous déplorons cette restriction, qui ne fait pas partie de Merise.*

[9] *Notons cependant immédiatement qu'une étude complète entraînerait des associations et des contraintes entre objets VALEUR : un paramètre défini selon un certain type, avec des valeurs définies, peut très bien correspondre avec un attribut d'un type légèrement différent, si son format de base est compatible et que ses valeurs possibles sont incluses dans celles du paramètre…*

Nous en arrivons donc à une deuxième ébauche de métamodèle :



On utilise ici pour définir l'héritage entre types une relation normale, et non pas le symbole d'héritage, car nous sommes dans un métamodèle, donc nous définissons ce qui permettra de gérer les modèles : la relation **TYPE hérite de TYPE** signifie que les attributs du type parent feront également partie du type héritier.

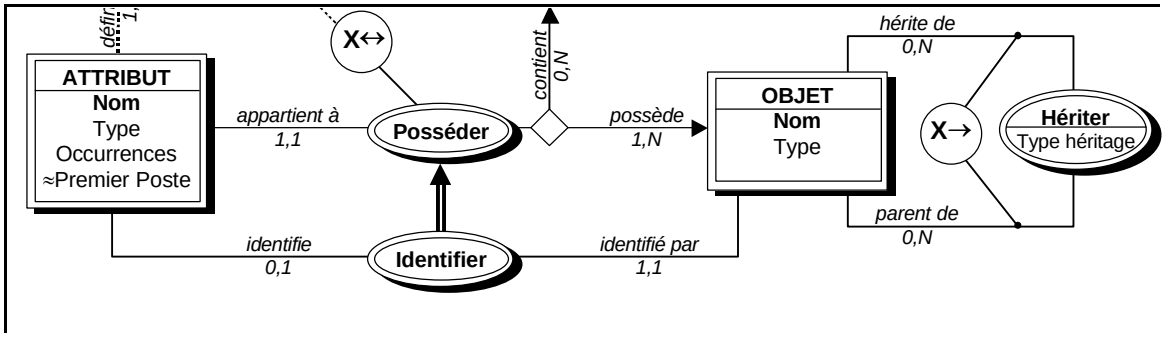
Anecdotiquement, notez que l'objet **VALEUR** nous fait ici la totale quand aux enrichissements sur les attributs : nous avons un tableau d'agrégats Liste[ ]{...} contenant notamment une méthode facultative *≈Contrôle()* !

Par ailleurs, on constatera que les méthodes, bien qu'appartenant à des objets, des agrégats ou des associations, tout comme les attributs "données", n'ont pas été prises en compte ici. C'est normal, puisqu'il s'agit en fait d'une forme d'action : contrairement à une action "ordinaire", dont les entrées et sorties sont parfaitement définies, une méthode appartient à une entité dans ce sens qu'une partie de ses entrées/sorties ne concerne que des attributs de cette entité, le reste étant paramétré (paramètres en entrée ou résultat seront associés à diverses entités, selon les cas). Ceci sera donc étudié après les actions.

Les objets

Tout objet a un nom et un type (modèle, occurrence, singularité). Il contient un ou plusieurs attributs, dont l'un est l'identifiant (par rapport à Merise standard, nous n'accepterons qu'un seul attribut comme identifiant, quitte à utiliser un agrégat).

Il peut également hériter des attributs et relations d'un ou plusieurs autres objets, cet héritage pouvant être sémantique ou génétique. Ceci nous permet de compléter notre métamodèle comme suit[10] :



Évidemment, la contrainte d'exclusion de la relation **OBJET hérite de OBJET** dépend du type d'héritage : ⊣(X↔)⊢ pour un héritage sémantique, ⊣(X↔)⊢ pour un héritage génétique. Une représentation complète au niveau du symbolisme est ici impossible, puisque concernant un chemin complet.

[10] *Les pointillés indiquent qu'on ne montre qu'une partie du schéma.*

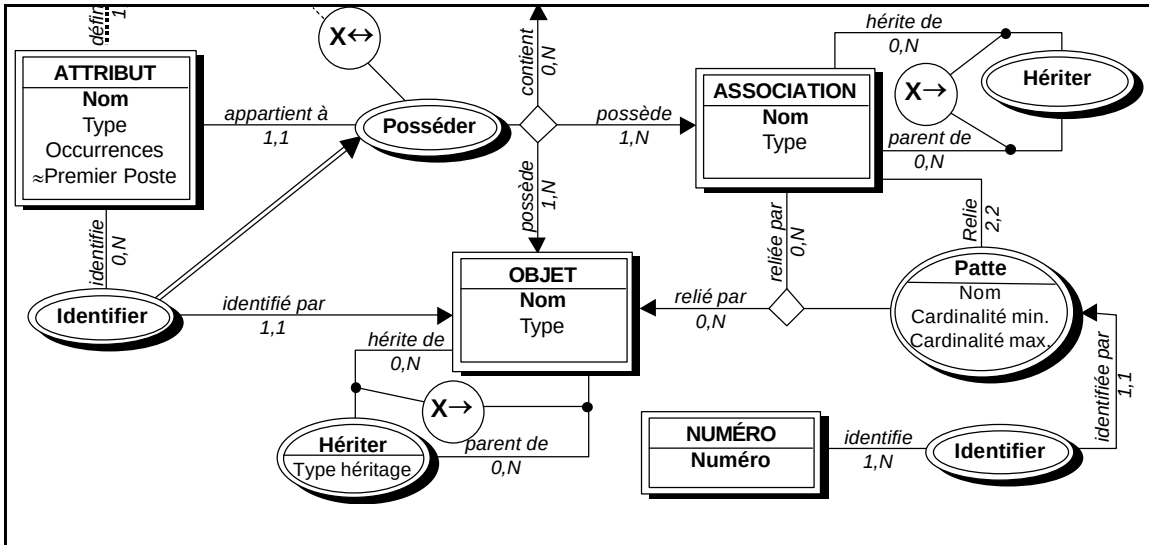
Les associations et les pattes

Comme les objets, les associations ont un nom et un type. Elles peuvent contenir des attributs, et chacune est reliée par deux pattes (ayant un nom et des cardinalités) à deux objets et/ou associations. Elles peuvent hériter d'une ou plusieurs autres associations (sous réserve qu'elles relient les mêmes objets et dérivent toutes d'une même association de base : c'est la même restriction que pour les objets).

La spécificité d'une association, c'est qu'elle a toujours deux pattes, pas plus, pas moins[11]. De plus, ces pattes peuvent toutes deux relier le même objet (association réflexive) et/ou porter le même nom : par exemple l'association associé entre deux personnes est symétrique : si Pierre est associé de Paul, on a en sens inverse Paul est associé de Pierre[12].

L'association Patte peut donc exister en deux exemplaires entre une même association et un même objet ou association. Ceci est interdit puisque tout identifiant doit être unique. Pour permettre la modélisation, nous allons donc devoir utiliser un objet fictif différenciant ces pattes. Sa population se composera en tout et pour tout de deux individus, nommés 1 et 2 (ou A et B, Gauche et Droite, Vrai et Faux… ceci est sans importance, du moment qu'il en existe exactement deux).

Ce qui nous permet d'enrichir le métamodèle (après une petite réorganisation spatiale) ainsi :



Il serait intéressant de pouvoir prendre en compte les pattes alternatives dans ce métamodèle (pour pouvoir ne faire qu'une seule définition), mais nous garderons ceci pour une étude ultérieure plus approfondie, donc hors du cadre de cet article.

Les prédicats

Il existe deux sortes de prédicats : ceux qui déterminent les héritages sémantiques par spécialisation, et ceux qui permettent de déclencher des méthodes et actions.

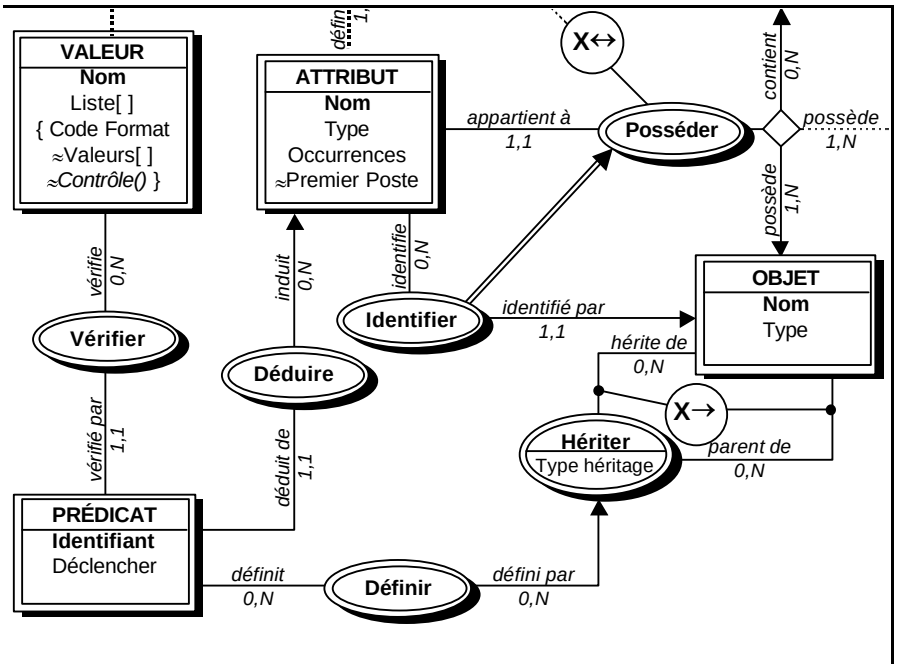
Le second cas étant indissociable des méthodes et actions, occupons-nous ici uniquement du premier cas : un ou plusieurs attributs déterminent, selon leur(s) valeur(s), à quelle classe se rattache un objet.

Dans notre métamodèle, nous allons donc définir un nouveau type d'objet : **PRÉDICAT**, relié d'une part à **ATTRIBUT** et d'autre part à l'association **Hériter** :

---

11 Nous avons montré dans la 1ère partie que toute association pouvait être réduite à 2 pattes, en utilisant les associations d'associations.

12 En théorie, on pourrait même avoir une association réflexive symétrique sur le même objet…



Les cardinalités s'expliquent aisément : un héritage n'est pas nécessairement défini par un prédicat ; a contrario, plusieurs attributs – et donc autant de prédicats – peuvent être nécessaires[13].

L'association **Vérifier** avec **VALEUR**, qui permet d'associer une liste de valeurs à un prédicat, est sémantiquement légèrement différente de son homonyme avec **TYPE**, car elle définit en fait un sous-ensemble des valeurs associées au type de l'attribut. Les occurrences de **VALEUR** concernées ne sont donc pas identiques, mais il existe entre elles une contrainte, dans le détail de laquelle nous n'entrerons pas maintenant[14].

Les contraintes ensemblistes

Arrivés à ce point, nous avons à peu près parcouru l'ensemble des concepts statiques. Je dis bien à peu près, car il y manque notamment les diverses contraintes ensemblistes (inclusion, exclusion, totalité, unicité, etc.). Celles qui mettent en cause des associations ne seront pas traitées ici[15].

Certaines contraintes (contraintes dites "de stabilité") peuvent cependant être prises en compte immédiatement :

- Une patte verrouillée est définie par un attribut à ajouter dans l'association **Patte**.
- Un attribut constant (non modifiable) est défini par un attribut à ajouter dans l'objet **ATTRIBUT**.
- Une relation définitive (dont toutes les occurrences doivent être créées en un seul processus) est définie par un attribut à ajouter dans l'objet **ASSOCIATION**.

Les actions

Nous arrivons maintenant à une partie légèrement plus complexe. Jusqu'ici, nous n'avons modélisé que des données, c'est-à-dire des choses relevant exclusivement du domaine "statique". Maintenant, nous nous attaquons aux traitements.

Nous pouvons décomposer les actions en trois familles, que nous étudierons l'une après l'autre :

- les **actions** proprement dites, correspondant à des règles de gestion, et dont l'ensemble des données manipulées et des règles de déclenchement est parfaitement défini ;
- les **méthodes**, actions dont ne sont conceptuellement définis que des liens avec des données appartenant à un objet (ou agrégat, ou association…), qui peuvent avoir des paramètres en

---

13 En théorie, une synchronisation est également nécessaire, mais nous ne nous en occuperons pas ici.

14 Les amateurs pourront essayer de modéliser cette contrainte (objets "objet valeur" associés à des objets "attribut valeur", etc.). Réponse dans la 3ème partie de ce dossier.

15 Mais le seront dans l'ouvrage plus complet à venir.

entrée et/ou renvoyer une donnée en sortie, paramètres et valeur renvoyée ne correspondant pas à des attributs connus lors de la définition.

- Les **routines**, méthodes complètement paramétrées n'ayant aucun lien conceptuel en-dehors du format de leurs entrées-sorties – c'est-à-dire non liées à un objet spécifique.

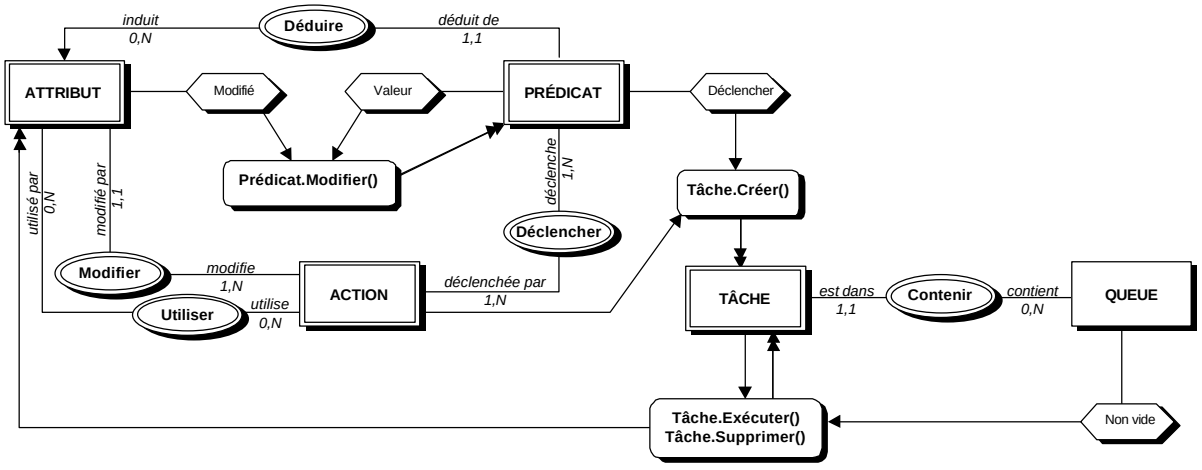
Les actions ordinaires

Rappelons que si les objets et associations peuvent constituer des occurrences ou des populations issues de modèles, il n'en est pas de même des actions[16], car toute action associée à des attributs d'un modèle est automatiquement applicable à chaque individu issu de ce modèle et n'est donc pas dupliquée.

Par contre, un attribut utilisé comme prédicat a une action individuelle. Pour mieux comprendre ceci, détaillons le fonctionnement d'un traitement dans le mode prédicat-action :

1) Un attribut d'un objet est modifié suite à l'exécution d'une action.
2) Si cet attribut fait l'objet d'un ou plusieurs prédicats[17], chacun d'eux est analysé pour vérifier si la nouvelle valeur a un impact sur une ou plusieurs actions (par exemple, si un prédicat déclenche une action avec la condition >100, la modification de la valeur de l'attribut de 80 à 90 n'aura aucun effet : on ne déclenche une action qu'à bon escient).
3) Si c'est le cas, un objet **TÂCHE** est créé (ou mis à jour dans le cas d'une action à plusieurs prédicats synchronisés) pour chaque action, et associé aux individus dont il utilise ou met à jour des attributs. Si l'ensemble des conditions de déclenchement sont remplies, cet objet est mis en queue (création d'une association avec une singularité[18] **QUEUE**, en tenant compte de la priorité de l'action).
4) Quand la tâche parvient en tête de liste, elle est exécutée puis supprimée (le résultat de son exécution pouvant déclencher l'activation d'autres actions).

Essayons donc de modéliser ce cheminement[19] au moyen d'objets, associations, prédicats et actions :



Comme vous pouvez le constater, une fois qu'un traitement est bien décrit, rien de plus simple que de le modéliser avec notre formalisme unifié, c'est-à-dire MCD Merise enrichi. Il ne reste plus, en effet, qu'à compléter le premier jet ci-dessus pour l'intégrer au métamodèle, après avoir répondu à quelques questions complémentaires :

- Quels sont les attributs de ces entités ?
- Le **PRÉDICAT** ci-dessus peut-il être conceptuellement fusionné avec le **PRÉDICAT** définissant l'héritage par spécialisation ?
- Si un métamodèle d'action contient des actions, comment celles-ci seront-elles définies et gérées dans la pratique ?

---

16 Si l'objet ACTION défini ici est un modèle, c'est parce que nous sommes dans un méta-modèle.

17 En principe, toute information devant avoir une origine et une utilité, tout attribut devrait être prédicat de quelque chose, la chaîne s'achevant sur un effecteur (par exemple, un libellé modifié déclenchera l'affichage de son contenu sur l'écran de l'utilisateur…).

18 Voir 1ère partie : une singularité est un objet dont la population est limitée à un et un seul individu, qui n'a donc besoin d'aucun identifiant en dehors de son type.

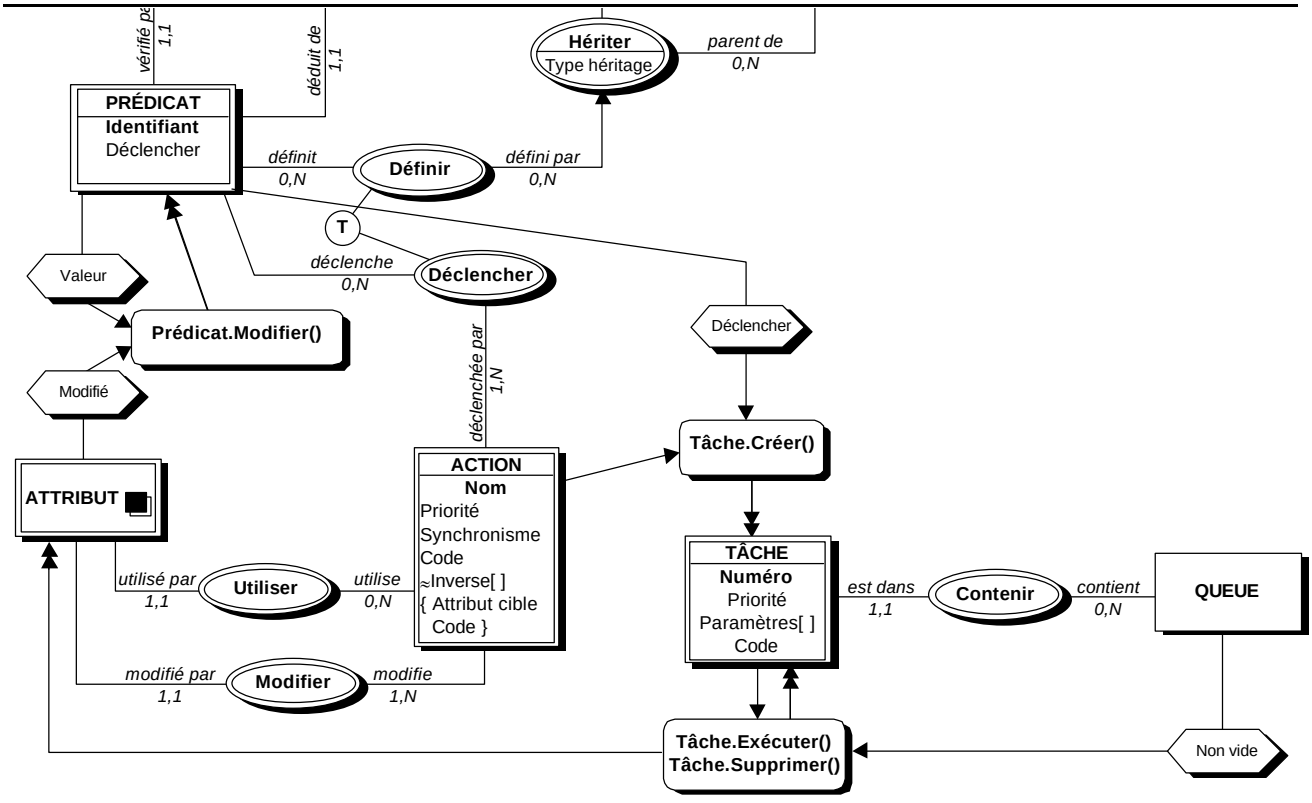
19 Tout le monde a évidemment reconnu le principe de fonctionnement d'un moteur d'inférence de Système à Base de Connaissances.

- À quoi servent les priorités ?

Essayons de répondre à ces questions :

- Une action n'a que peu d'attributs : son identifiant ; son code source[20] (qu'à la saisie une méta-action contrôle et utilise pour générer les liens avec les entités impliquées) ; éventuellement des codes source de fonctions inverses (pour permettre une recherche à partir du résultat voulu) ; une synchronisation (pour les déclenchements dépendant de plusieurs paramètres) ; une priorité ; et c'est à peu près tout, si on excepte la possibilité d'attributs internes (paramètres constants et/ou données intermédiaires utilisées lors de l'appel de méthodes et/ou routines).
- Une tâche reçoit un certain nombre d'attributs, qu'il faudrait conceptuellement définir comme des associations avec l'ensemble des informations nécessaires à son déroulement : attributs manipulés, code, etc. Pour ne pas compliquer le modèle, nous nous contenterons, en incorporant le schéma ci-dessus au métamodèle, de spécifier ces attributs.
- Un prédicat n'a qu'un identifiant relatif par rapport à l'attribut qu'il représente et un attribut booléen indiquant que la ou les actions associées doivent être déclenchées. Par contre, il est associé à un objet **VALEUR**, qui permet de connaître les conditions de déclenchement. Nous retrouvons effectivement les caractéristiques du prédicat d'héritage.
- Quant aux actions présentes dans le schéma ci-dessus, elles seront directement programmées.
- La queue, elle, n'a besoin d'aucun attribut.
- Les priorités permettent de gérer l'importance des actions, notamment dans un contexte temps réel : toutes les actions de priorité zéro seront exécutées avant toutes les autres, c'est-à-dire en principe immédiatement[21].

Complétons le modèle obtenu et intégrons-le au métamodèle en cours :



Ce schéma appelle quelques commentaires :

- Pour commencer, certains objecteront que **ACTION** et **TÂCHE** ne sont pas tout-à-fait conceptuels. C'est vrai, mais entrer dans le détail d'une conceptualisation complète augmenterait fortement le volume du présent article, sans ajouter grand-chose à sa compréhension. En d'autres termes, l'imperfection est volontaire.

---

20 Pourquoi pas en Java ?

21 Ce qui est fondamental si on veut pouvoir faire du temps réel.

- Nous avons mis qu'un attribut ne peut être modifié que par une et une seule action. Ceci constitue une condition nécessaire pour éviter les risques de non-intégrité[22].
- Il a été dit qu'une action pouvait contenir des attributs de travail. Nous intégrerons cette association dans le métamodèle complet (voir à la fin de cet article).
- La contrainte de totalité entre Définir et Déclencher nous dit qu'un prédicat sert obligatoirement à quelque chose…
- Les actions semblent être constituées de méthodes ? Pourquoi pas ? Pour plus de détails sur l'implémentation des méthodes, voir ci-dessous.
- Que se passe-t-il si une action (déclenchée 2 fois) modifie deux fois un attribut avant qu'une action subséquente ait pu impacter la première modification ? Voyons avec un exemple : si l'attribut passe de 80 à 110, puis à 90 :
  * si l'action déclenchée devait déclencher une alarme pour dépassement de la valeur 100, il faudrait logiquement l'annuler ;
  * si elle est simplement chargée de totaliser le nombre de dépassements de la valeur 100, elle doit être maintenue ;

  Bref, tout dépend du contexte, et ce cas sera étudié dans un autre ouvrage à venir.

Une question posée page précédente et à laquelle nous n'avons pas encore répondu : comment modéliser une action qui utiliserait d'autres actions ? Eh bien, c'est simple : il faut d'une part définir des objets recevant les résultats intermédiaires, et d'autre part que les sous-actions soient des méthodes ou des routines.

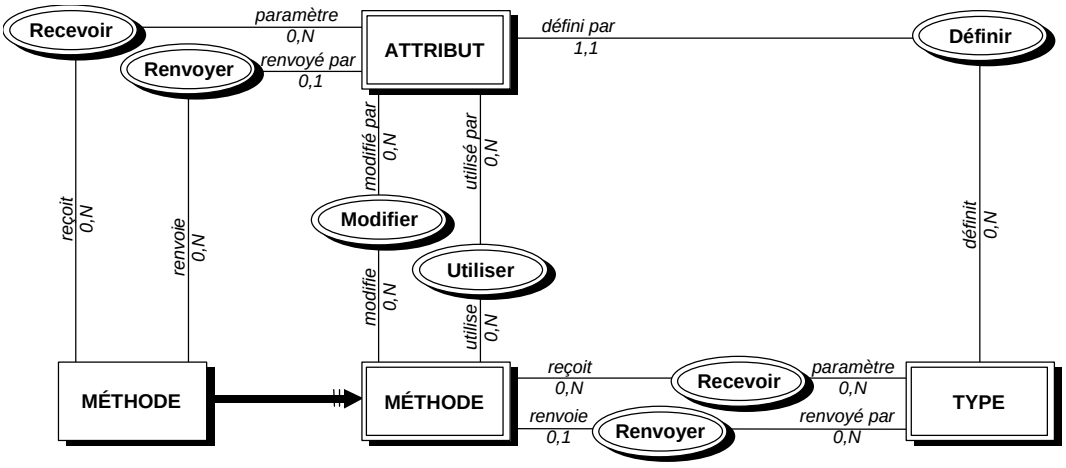
Les méthodes et routines

Maintenant que nous savons modéliser les actions, il est possible d'analyser les variantes que constituent une méthode ou une routine.

Qu'est-ce qui différencie une méthode ou une routine d'une action ? Simplement le fait que toutes ses entrées-sorties ne sont pas rattachées à des attributs précis lors de sa définition : on n'en connaît que le format. Ceci implique donc qu'on ne peut associer une méthode (partiellement) ou une routine (entièrement) à des attributs – via des prédicats –, mais seulement à des types.

Ce n'est que dans un deuxième temps, quand on définira une utilisation pour cette méthode ou routine, qu'on lui associera réellement des attributs. À la différence des actions, méthodes et routines sont donc prédéfinies comme modèles, les individus étant des occurrences.

Voyons ceci à l'aide d'une première modélisation :



Commentons ceci :

- Une méthode-type peut recevoir des paramètres et renvoyer une valeur. Elle peut également utiliser et/ou modifier des attributs dans un ou plusieurs objets (une routine, elle, n'a aucune association **Modifier** ni **Utiliser**).

---

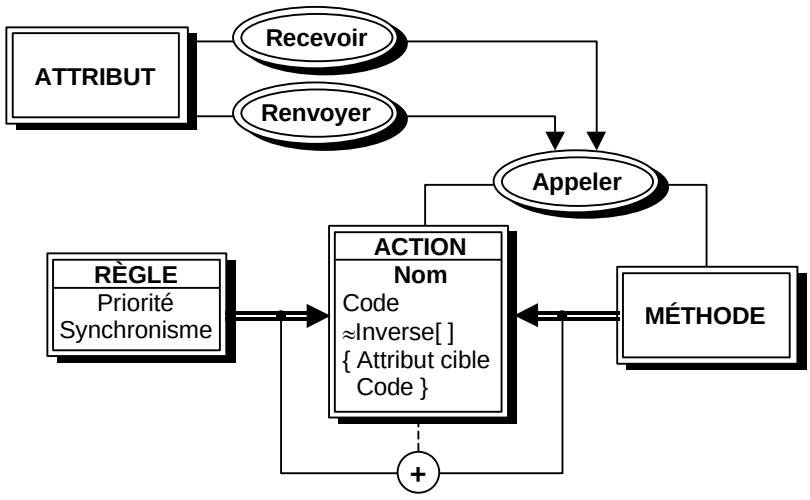
22 Condition loin d'être toujours respectée en informatique "traditionnelle"…

- Dans une occurrence de méthode, il faut remplacer les types par des attributs, qui doivent être de type (et de valeurs possibles) compatibles. Les occurrences de **Recevoir** et **Renvoyer**, pour chaque occurrence de méthode, doivent correspondre individu par individu avec celles de son modèle.
- Il faut également tenir compte (ce qui n'a pas été fait ici) que plusieurs paramètres peuvent être du même type, et qu'un même attribut peut servir pour plusieurs paramètres : ceci nécessite un identifiant complémentaire pour les associations **Recevoir**.
- Nous retrouvons les associations **Modifier/Utiliser ATTRIBUT** déjà définies pour les actions.

Le bilan de cette modélisation ? C'est qu'une méthode ou une routine peut être considérée comme une variété d'action pouvant posséder des paramètres et/ou renvoyer une valeur, et qui n'est déclenchée que lors de l'exécution d'une action/méthode/routine appelante[23]. Comme elle n'est déclenchée que par une action ou une méthode déjà en cours d'exécution, il est inutile de lui définir une priorité, puisqu'elle doit être exécutée immédiatement, sous peine d'interrompre l'action en cours.

De plus, une occurrence de méthode n'ayant d'existence que dans la mesure où elle est appelée par une action (le modèle définit la méthode, l'occurrence est son utilisation), on n'a pas affaire dans le métamodèle à un objet, mais bien à une association.

Nous allons donc compléter comme suit la définition d'un objet **ACTION**, une action ordinaire devenant une **RÈGLE** (de par son rôle en tant que règle de gestion ou règle d'un Système à Base de Connaissances), et les occurrences de méthodes étant tout simplement l'association **Appeler**, sur laquelle viendront se greffer **Recevoir/Renvoyer ATTRIBUT** :



Il s'agit bien ici d'héritage sémantique, sachant que toute action est soit une règle, soit une méthode, mais pas les deux, et que :

- Une règle a des prédicats et des attributs cibles, pas une méthode.
- Une méthode peut avoir des paramètres et/ou un résultat en retour, et est appelée par une règle ou une méthode.

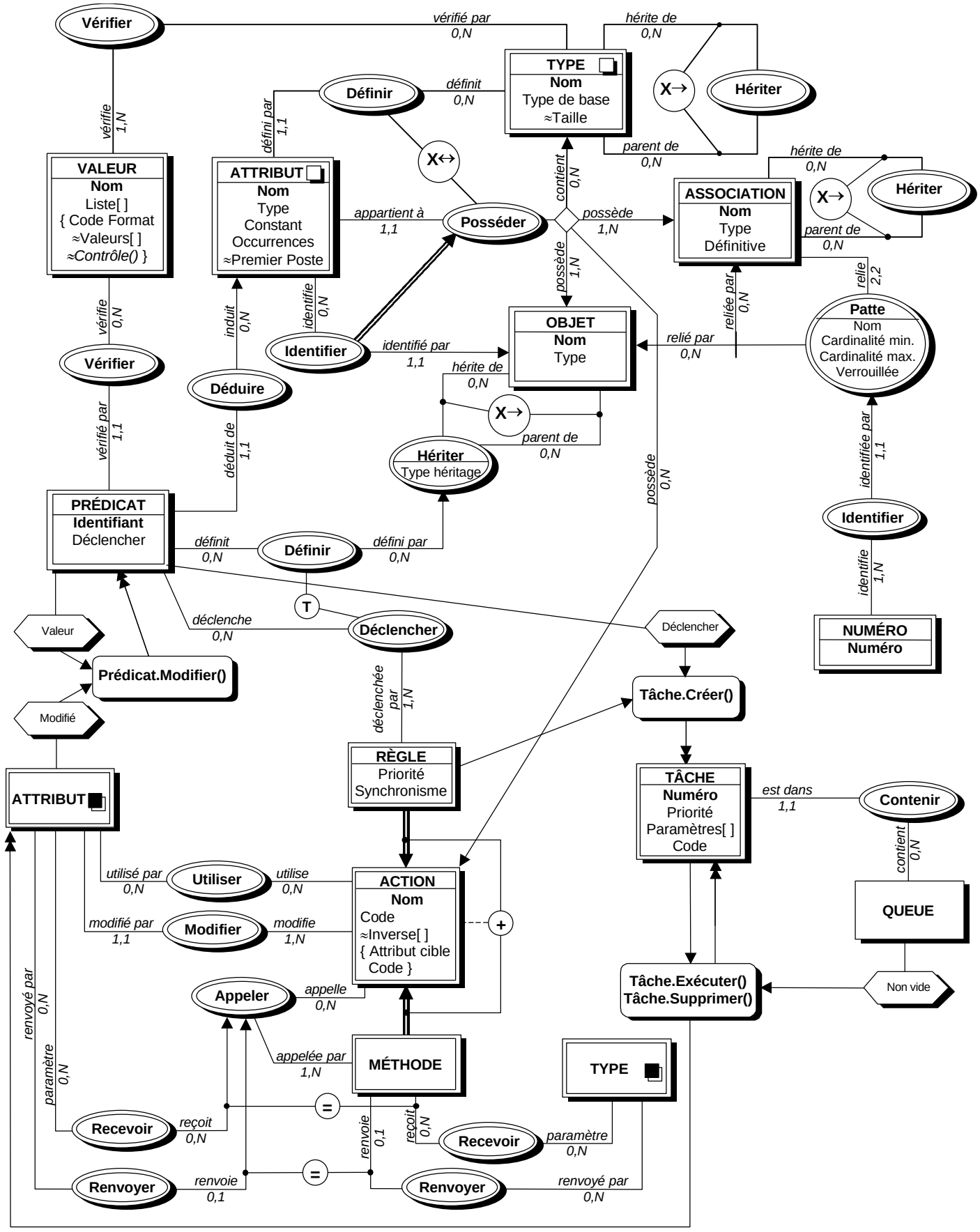
Finalement, c'est plutôt simple, non ?

Il est cependant recommandé, dans ce style de conception, de limiter au maximum l'arborescence des appels à des méthodes et/ou routines en définissant plusieurs règles simples de préférence à une règle complexe: plus une action s'exécute rapidement, mieux c'est.

23 Sachant qu'une action et une méthode peuvent appeler des méthodes et/ou des routines, mais qu'une routine ne peut appeler que des routines.

Synthèse

Après avoir intégré les méthodes, nous avons donc obtenu un métamodèle – incomplet, on le sait, mais cependant significatif – qui est celui-ci :



Et voici la liste des attributs :

Nom	Description	Format
ACTION.Code	Descriptif du traitement à effectuer.	Code source compilable et/ou interprétable (Java ?)
ACTION.Inverse[ ]	Tableau des traitements inverses possible (retrouver une valeur origine en connaissant le résultat). Autant de postes possibles que d'attributs en entrée.	Tableau d'agrégats
ACTION.Inverse[ ].Attribut cible	Identifie l'attribut source devenu cible dans le traitement inverse (pour une méthode, la définition cible un type).	Pointeur
ACTION.Inverse[ ].Code	Descriptif du traitement inverse à effectuer.	Code source compilable et/ou interprétable (Java ?)
ACTION.Nom	Nom de la règle de gestion.	Texte
ASSOCIATION.Définitive	Si toute la population vs objet commun doit être créée en une fois.	Booléen
ASSOCIATION.Nom	Nom "Merise" de l'association.	Texte
ASSOCIATION.Type	Modèle, occurrence, particularité.	Code
ATTRIBUT.Constant	Si attribut non modifiable.	Booléen
ATTRIBUT.Nom	Nom de la donnée dans l'objet ou agrégat qui la contient.	Texte
ATTRIBUT.Occurrences	Nombre de postes si l'attribut est un tableau (min. 1 = pas un tableau).	Numérique
ATTRIBUT.Premier Poste	Numéro d'ordre du 1er poste dans un tableau (peut être différent de 0 ou 1).	Numérique
ATTRIBUT.Type	Attribut obligatoirement renseigné ou pas.	Booléen
Hériter.Type héritage	Héritage sémantique ou génétique.	Booléen
OBJET.Nom	Nom de l'objet.	Texte
OBJET.Type	Modèle, occurrence, singularité.	Code
Patte.Cardinalité max.	Nombre maximal d'occurrences possibles pour que le système soit cohérent.	Numérique
Patte.Cardinalité min.	Nombre minimal d'occurrences possibles pour que le système soit cohérent.	Numérique
Patte.Nom	Nom de la patte, utilisé quand on vient de l'objet ou association connecté.	Texte
Patte.Numéro	Différencie les pattes dans le cas d'une association réflexive (dans le modèle : NUMÉRO.Numéro).	2 valeurs
Patte.Verrouillée	Si l'association ne peut être supprimée sans que l'objet ou association rattaché soit supprimé également.	Booléen
PRÉDICAT.Déclencher	Indique si la ou les actions dépendantes de ce prédicat devraient être déclenchées ou si l'héritage spécialisé est valide pour l'objet.	Booléen
PRÉDICAT.Identifiant	Permet de différencier les différents prédicats concernant un même attribut.	Texte
RÈGLE.Priorité	Urgence d'exécution.	Numérique

Nom	Description	Format
RÈGLE.Synchronisme	Combinaison des prédicats pour permettre un déclenchement.	Formule booléenne
TÂCHE.Code	Descriptif du traitement à effectuer.	Code compilé et/ou pré-interprété
TÂCHE.Numéro	Identifiant de la fonction dans la queue.	Quelconque
TÂCHE.Paramètres	Valeurs en entrée et destination du résultat.	Tableau/Agrégat
TÂCHE.Priorité	Urgence d'exécution (hérité de l'action).	Numérique
TYPE.Nom	Nom du type.	Texte
TYPE.Taille	Taille de la donnée (selon type de base)	Numérique
TYPE.Type de base	Format de base de la donnée : (caractère - chaîne de caractères fixe ou variable - entier court, standard, long, signé ou non – nombre avec décimales fixes – flottant – complexe – booléen - texte formaté – image - etc.)	Code
VALEUR.Liste[ ]	Liste des valeurs possibles pour un attribut	Tableau d'agrégats
VALEUR.Liste[ ].Code Format	Usage en contrôle des valeurs associées (valeur(s) autorisée(s) - valeur(s) interdite(s) - valeur minimale et/ou maximale, masque, routine…)	Code
VALEUR.Liste[ ].Contrôle()	Routine permettant si nécessaire des contrôles complexes	Routine
VALEUR.Liste[ ].Valeurs	Liste des valeurs associées au type de contrôle.	Selon TYPE
VALEUR.Nom	Nom du contrôle	Texte

Conclusion de la 2ème partie

Nous avons donc vu qu'il était possible de définir, sous la forme – et avec les techniques – d'un MCD Merise amélioré – un métamodèle sémantique universel, qu'il suffit de renseigner pour pouvoir générer une modélisation de n'importe quelle application, qu'elle soit de gestion, objet, temps réel, système expert...

Bien sûr, ce qui a été élaboré tout au long du présent article est loin d'être absolument complet, et prête à de nombreuses critiques. Cet état de choses est délibéré, pour plusieurs raisons :

- Une modélisation complète nécessite une étude beaucoup plus approfondie – une LETTRE entière n'y suffirait pas.
- J'ai conçu, pour ce genre de modélisation, un certain nombre d'éléments dont je tiens à la paternité et que j'estime donc prématuré de présenter au public. Cette restriction sautera dès lors qu'existera un produit opérationnel mettant en œuvre ce concept. Que les frustrés prennent leur mal en patience ou viennent participer au projet.

Dans la suite – et fin – de ce dossier, à paraître dans LA LETTRE n°28, nous verrons justement comment mettre en œuvre ce métamodèle[24] pour réaliser un véritable Système d'Information intégré, permettant tout à la fois de concevoir, modéliser, prototyper (en RAD, s'il vous plaît !) et mettre en service n'importe quelle application informatisable.▲

(à suivre)

Jean-Luc Blary

[24] ...À quelques astuces près – pour les mêmes raisons que ci-dessus, ce qui frustrera encore une fois plusieurs d'entre vous.

Au sommaire de la 3ème partie

- ✓ Mise en pratique du modèle sémantique universel sous forme d'un SGBD
- ✓ Perspectives d'avenir

Sources documentaires

MERISE, support de cours
ABOUHAIR G.
IBSI, Paris 1988, 1991, 1992

EPHIBASE, SGBD entité-relation orienté objet, dossiers de conception et prototype
BLARY J-L.
EPHITEQ, Caëstre 1989-1990, 1993, 1995, 1996

ODESYS, Outil d'aide à l'évolution du Système d'Information, dossiers de conception
BLARY J-L., THELLIEZ Ph.
EPHITEQ, Caëstre 1993-1996

MERISE & OBJETS, support de cours
BLARY J-L.
EPHITEQ, Caëstre 1995

MERISE, support de cours
BLARY J-L.
EPHITEQ, Caëstre 1996

Le Réseau Sémantique Universel (1ère partie)
BLARY J-L.
La Lettre de l'ADELI n°26, janvier 1997

De la modélisation systémique au réseau sémantique
BRES P-A., ROCHFELD A., TABOURIER Y., SIBERTIN-BLANC C.
Conférence-débat de l'AFCET, Paris 15/11/1990

SGBD avancés : bases de données objet, déductives, réparties
GARDARIN G. VALDURIEZ P.
Eyrolles, Paris 1990

Merise vers une modélisation orientée objet
MOREJON J.
Les Éditions d'Organisation, Paris 1994

Réseaux de neurones
NADAL J-P.
Armand Colin, Paris 1993

Le langage C++
STROUSTRUP B.
Addison-Wesley, Paris 1992

Moteurs de systèmes experts
VOYER R.
Eyrolles, Paris 1987

☛*Intéressés, critiques, puristes, adversaires, partisans... pour en savoir plus ou participer :*

☎ *0.660.602.702*
0.328.402.702
jlblary@nordnet.fr
✉ *EPHITEQ - Château Vallée - 59190 CAËSTRE*

Formel et informel

Chaos, planète bleue, économie et simulations par ordinateur de faits sociaux.

Laplace convoqué par Napoléon et interrogé sur Dieu dans tout cela, répondit à l'empereur qu'il ne voyait pas l'intérêt de cette hypothèse.

Il est d'usage de placer le développement de l'ingénierie dans le contexte laplacien où grâce à du papier et des crayons, (beaucoup, et quelques gommes pour les erreurs) on peut calculer l'évolution future et les situations rétroactives de l'univers. Le progrès technologique aidant, on se sert maintenant d'ordinateurs pour faire tout cela, c'est chic. Mais les limites de la prévisibilité apparaissent aussi bien dans la météorologie, voir Ivar Ekeland[1], que dans l'économie[2]. En ce qui concerne l'économie, certains annoncent le retour de la grande dépression[3] et d'autres une nouvelle économie mondiale[4].

La théorie générale du chaos montre que certains phénomènes sont difficilement intégrables, donc imprévisibles, comme le pressentait Poincaré dans l'étude du problème des trois corps. Il faut donc se résigner à un univers où le calcul préalable ne résout pas tout. Certains modernes n'hésitent pas à dire que le calcul ne résout ou ne démontre qu'une très faible part des choses de cet univers. En fait le calcul laplacien ne peut résoudre que les abstractions les plus simplifiées nous permettant effectivement une manipulation symbolique sous une forme accessible à nos faibles neurones. Les psychologues démontrent que le commun des mortels ne peut retenir en simultané que dix éléments mentaux à la fois. Nos théories scientifiques seraient-elles limitées par la réalité biologique de notre cortex cérébral ?

Dans ce contexte où le pragmatisme revient en force, la première bonne nouvelle est que la conduite et le contrôle de projets reviennent à l'honneur, étant donné les divergences exponentielles accélérées de tout système aux conditions initiales instables (voir Thom et Prigogine[5]).

Il faut donc plus que jamais pour les projets un tant soit peu complexes, vérifier au pas-à-pas leur bon déroulement, remettre au premier plan le comité de pilotage merisien, prendre des mesures, publier, communiquer, diffuser, discuter, gérer, et décider.
Il me paraît d'ailleurs évident que cela force ou avantage un contexte "démocratique" de prise de décision, quoique le mot démocratie soit lui aussi sémantiquement instable dans son histoire.
Mais si l'instabilité est de règle et facteur d'insécurité psychologique, c'est une explication de la chute de tous les systèmes autoritaristes et déterministes, bonne nouvelle. En particulier la fixation a priori des prix de marché apparaît comme une erreur fondamentalement contre-nature. Prigogine et compagnie seraient-ils influencés par l'école économique dite de Chicago, purs chantres de thèses ultra-libérales ?

---

1 Le Calcul, l'Imprévu. Ivar Ekeland, Seuil, Paris.1984.

2 Les économistes en procès. J.P. Dupuy, G.Granger, M.Henry, A.d'Autume, E.Malinvaud, A.Orléan, M.Allais. Le Monde des Débats. N° 10. Décembre 1993. Le Monde. 15, rue Falguière. Paris.

3 Le retour de la très grande dépression. J.L. Gombeaud, M.Décaillot. Economica. Paris.

4 Economic growth in Europe since 1945. N.Crafts, G. Toniolo. Cambrige University Press.

5 Les lois du chaos. Ilya Prigogine. Flammarion. Paris. 1994.

Pourtant, un ouvrage stimulant, adepte du calcul et de la simulation algorithmique, apparaît chez les anglo-saxons, et pose quelques bons problèmes, par exemple sur la détermination des prix.
Dans ce cadre, un oligopole, (dans le cas discuté : deux fournisseurs), est discuté et simulé par un processus neutre homme-machine.
Cet excellent ouvrage[6] se situe donc dans la lignée laplacienne et tendrait à faire croire à un avenir lumineux pour les sciences dites sociales. Essayant ainsi de faire un pas vers les sciences exactes, vieux rêve de Marcel Mauss[7], et de paraître crédible pour récolter quelques subventions gouvernementales. Est-ce là le fait de la jeunesse relative des sciences humaines ou tout simplement faut-il que l'homme soit de par sa nature le fruit du hasard et la cause du brouhaha généralisé ? On peut également invoquer l'hubris, cette ambiance mystérieuse et antique propre aux inspirations sacrées.

Les météorologistes eux, font un pas en arrière. Ayant publié il y a une dizaine d'années des scores de prévisions exacts sur une dizaine de jours, les voilà qui après quelques mesures de satisfaction peut-être plus rationnelles, pensent atteindre 80 % de satisfaction pour 5 jours de prévisions, et avouent tomber à 70 % au bout du sixième. Pour la semaine prochaine, il suffit de tirer à pile ou face.

Se repose alors le problème fondamental de la nature de cet univers. Dieu s'adonnerait-il aux jeux de hasard ? Einstein pensait que non, la nature ne pouvait être le résultat d'un hasard massif. Catastrophe nouvelle, la terre, cette boule bleue d'eau et d'oxygène, ne devrait sa stabilité axiale qu'au seul hasard.

Cette stabilité axiale, c'est-à-dire la régularité de rotation de cette sphère autour d'un axe, conditionne son climat tempéré, la température moyenne de la planète oscillant autour de 15° centigrades depuis quelque milliers d'années même en période glaciaire.
Cette stabilité qui détermine notre apparition, évolution et survie ne serait due qu'à la présence fortuite d'une lune, suffisamment grosse pour stabiliser la terre dans son orbite, mais suffisamment petite pour éviter des désagréments graves comme de trop fortes marées, par exemple, qui ravageraient systématiquement les littoraux. Rappelons que 80 % de la population terrestre vit en bord de mer.

Or la seule explication de la présence d'une seule lune autour de la planète, et donc son influence bienfaisante, serait due à la pénétration d'une météorite suffisamment grosse pour arracher la lune à notre planète et suffisamment petite pour éviter de la désintégrer.

Sans ces concours de circonstance, pas de planète bleue, ou pour peu de temps.

Comment calculer alors le nombre probable de planètes bleues de l'univers ?

Il faudra s'y atteler avec des machines plus précises que les nôtres, mais les conditions initiales de ces phénomènes fleurant l'instabilité, il faudra peut être y ajouter un sens divinatoire aigu.

Alors la nécessité de modèles formels systématiques prend un coup de vieux. La boucle classique de la science, modèles, prévisions, constations, écarts, synthèse et validation semble bien battue en brèche par un monde qui ne l'entend pas de cette oreille.

---

6 Simulating societies. The computer simulation of social phenomena. F.Bousquet; J.C. Castro Caldas, C.Cambier, C.Castelfranchi, H.Coelho, R.Conte, N.Dalton, J.Doran, A.Drogoul, J.Ferber, N.Gilbert, B.Latané, P.Mellars, S.Mithen, P.Morand, C.Mullon, A.Nowak, M.Palmer, A.Penn, J.Quensière, R.G. Reynolds, A.C. Séror, K.G.Troizsch. UCL Press, Londres.1994.

7 Essais de sociologie. Marcel Mauss. Coll. Points. Editions de Minuit. Seuil. 1968.

Si les pratiques divinatoires plongées dans le fouillis des superstitions et mysticismes ne semblent pas d'un grand recours, le constat d'un mur des connaissances et méthodes d'actions scientifiques doit nous conduire à plus de modestie dans notre volonté de dominer le monde, et revenir à des formes respectueuses de vie sur cette planète et au-delà. Nos petites excursions astronomiques récentes ne nous permettent pas de penser que cette planète est à refiler toute pourrie à quelque futur acquéreur. On ne pourra filer ailleurs, parce que il n'y a peut-être pas d'ailleurs. L'histoire de nos civilisations, toutes mortelles, est de surcroît bornée par une première catastrophe à venir, une ère glaciaire qui commence dans un millier d'années, et d'autre part par l'anéantissement probable d'une bonne partie des espèces biologiques sur cette terre dans une centaine de millions d'années, qui semble être la demi-valeur moyenne de règne biologique avant la collision avec le météore monstre successeur de celui qui peut-être fût la cause de l'extinction des dinosaures du passé et volatilisa le golfe du Mexique en cendres il y a 60 millions d'années.

Serons-nous les prochains dinosaures à disparaître ?

Qui récoltera le fruit de nos agitations ?

Vive la gestion de projets. ▲

Michel Demonfaucon

L'infobésité

Dans des temps qui ne paraissent pas si lointains, nous manquions cruellement d'une denrée rarissime : l'information.

La création d'un message sur la paroi d'une caverne exigeait la patience d'un artiste et il fallait venir sur place pour consulter un message qui ne pouvait être mis à jour. Au Moyen Âge, la création du moindre ouvrage littéraire occupait une équipe de moines copistes.

Puis l'imprimerie, le télégraphe, le phonographe, le téléphone, le cinématographe, la radiophonie, la télévision, l'informatique, la télématique, Internet sont venus étancher notre soif d'informations. Ces moyens techniques ont permis de reproduire l'information écrite, sonore, imagée, animée et de la transmettre, instantanément, loin du lieu où elle a été produite.

Aujourd'hui, de l'information, en voulez-vous, en voilà !

- Il suffit de tourner un bouton d'un récepteur de télévision pour pouvoir consulter des centaines de chaînes.

- Il suffit de lancer un mot-clé sur Internet pour ramener des tombereaux de documents écrits, imagés, animés, sonores sur le thème interpellé.

Après des siècles de pénurie, n'est-on pas en train de périr d'abondance ?

Telle l'oie, dont le foie dès sa naissance est destiné au palais des gourmets, l'homme est voué au gavage d'informations.

Bien sûr, l'information prend des formes très variées, beaucoup plus agréables à ingurgiter que la bouillie insipide dont on nourrit les oies ; l'entonnoir informationnel n'est pas encore fiché dans notre cerveau. Mais, si l'on y réfléchit bien, l'effet de la surconsommation imposée reste analogue.

Nous passons désormais une grande partie de notre temps à attendre, à guetter, à consommer de l'information. L'information a envahi tous les rouages de notre société. Une nouvelle maladie insidieuse qui provoque déjà des ravages dans nos organisations.

La recherche de la bonne information

Les internautes consacrent :

- 70 % de leur temps à chercher l'emplacement de l'information ;
- 25 % à isoler l'information utile de la gangue des informations accolées ;
- 4 % à consulter des documents relatifs au thème de la recherche ;
- et seulement 1 % à comprendre ce qu'ils sont venus chercher.

Les techniciens déclarent que l'insuffisance des débits de transmissions freine le développement de la recherche documentaire. Augmentons encore les diamètres des tuyaux pour faire passer encore plus d'informations.

L'accumulation

L'information enregistrée n'est jamais détruite car elle constitue une trace indispensable aux archéologues du présent. Ainsi, dans les entreprises, on crée en permanence de nouvelles bases. Mais personne n'est chargé de faire le ménage en éliminant les informations périmées, dupliquées, erronées.

La pollution

Il est bien tentant pour certaines firmes de parasiter l'information recherchée par un consommateur potentiel en y instillant, d'une façon plus ou moins subliminale, une information promotionnelle pour tenter de vendre leur propre production.
On arrive à ce paradoxe des journaux gratuits payés par les annonceurs en fonction du nombre de lecteurs potentiels de pauvres articles rédactionnels.

Pouce !

Assez de bases documentaires où l'on retrouve les mêmes textes fondateurs coupés collés de façon différente ! Assez de bases de données bourrées de chiffres que l'on introduit dans d'autres bases de données pour les associer à d'autres chiffres issus d'autres bases de données !

Pour une écologie de l'information

Avant de créer la moindre information, demandons-nous comment nous pourrions ne pas contribuer à la prolifération et au foisonnement informationnel.
Demandons-nous à quoi peut servir cette information et comment nous pouvons la mettre à disposition de ceux qui en ont besoin sans pour autant polluer les autres.

Réfléchissons ensemble

Pourrions-nous donner à l'une de nos futures commissions le thème de réflexion suivant :

Comment mettre à la disposition de chacun son information nécessaire et suffisante. ▲

Alain Coulon

Oui, j'ai gardé l'accent...

Faut-il accentuer les majuscules ? Voici une réponse à cette question d'une importance "capitale".

Tout le monde l'a constaté : dans certains écrits, les majuscules sont accentuées, dans d'autres non.

Tout le monde a un avis :

- certains mettent les accents parce qu'ils estiment que c'est plus lisible ;
- d'autres ne les mettent pas par habitude, ou parce qu'ils ne savent pas comment faire avec leur clavier ;
- d'autres encore ne les mettent pas parce qu'ils ont appris (souvent par un professeur) que les majuscules - tout comme le I n'a pas de point - ne prenaient jamais d'accent.

Bref, c'est le chaos. Pour mettre un peu d'ordre là-dedans, un petit tour d'horizon s'impose.

Le dictionnaire

Si on veut se donner la peine de jeter un coup d'œil dans le Petit Larousse, que constate-t-on ? Les différentes entrées sont en majuscules... avec les accents. Le mot "créer" apparaîtra non pas comme **CREER**, mais bien **CRÉER**. Serait-il possible que depuis un siècle ce monument de référence de la langue française se fourvoie ?

☒ Score : pour 1 - contre 0.

La Presse

Quelle est la pratique dans la presse. Dans leur grande majorité, les quotidiens accentuent les majuscules... la plupart du temps - et toujours quand il est nécessaire de lever une ambiguï té : si une manchette annonce **PATRICK SEBASTIEN ARRETE**[1], faut-il entendre par là qu'il renonce à la scène ou bien qu'il a été mis en examen ?

Dans la presse en général, les pratiques sont partagées, donc un point partout.

☒ Score : pour 2 - contre 1.

La dactylo classique

Si nos braves secrétaires travaillent encore avec une machine à écrire, la question ne se pose pas : impossible de taper un É, un È, un À, un Ù - ni même un Ç, dont la cédille partage ici le sort des accents.

Le sort des Â et autres Ï est à peine plus enviable, puisque généralement la touche permettant de mettre circonflexe ou tréma le fait à hauteur des caractères minuscules, le résultat avec les majuscules étant catastrophique.

☒ Le choix étant ici de nécessité, pas de score.

---

[1] *Il faut bien sûr lire **PATRICK SÉBASTIEN ARRÊTE** (il a recommencé depuis, mais c'est une autre histoire).*

La dactylo moderne

Les observateurs auront remarqué que les tant controversées majuscules accentuées sont bel et bien présentes dans le corps de cet article. En effet, toute personne utilisant un traitement de texte moderne n'a aucune excuse pour ne pas accentuer les majuscules, puisque celles-ci sont présentes dans toutes les polices de caractères standardisées (ANSI). Il est vrai que si c'est un jeu d'enfant sous Windows 95 (à 2 caractères près), c'est un peu moins simple sous Windows 3.x (il faut généralement passer par la table de caractères), et limité sous DOS si on utilise la page de code 437 (qui ne contient pas tous les caractères accentués français).

Alors, si "ça ne se fait pas" d'accentuer les majuscules, pourquoi le standard ANSI (pourtant américain, donc non utilisateur des accents) les a-t-il implémentées ?

☒ Score : pour 3 - contre 1.

L'histoire

Avant l'invention de la machine à écrire, il était d'usage d'accentuer les majuscules.

Avec l'apparition de celle-ci, c'est un modèle de clavier américain qui s'est imposé (Remington). Nous avons hérité d'une version "francisée", mais qui ne comportait pas suffisamment de touches pour prévoir toutes les combinaisons d'accents (et nous avons dans la foulée été privés du Ç déjà cité, ainsi que du œet du æ, tant minuscules que majuscules).

Bref, les majuscules accentuées ont disparu - par force - des textes dactylographiés pendant pratiquement un siècle. C'est une habitude dure à perdre... mais une mauvaise habitude, puisque si ce qui se conçoit bien s'exprime clairement, la remarque vaut aussi pour la typographie.

☒ Score : pour 4 - contre 1.

Conclusion

Le score est sans équivoque : il faut accentuer les majuscules. Si on peut pardonner aux dactylographes en herbe, et - partiellement - à ceux et celles qui n'ont pas encore la chance d'avoir Windows 95 ou NT - les autres - y compris les adeptes du manuscrit - n'ont aucune excuse.

Si vous avez Windows 95 ou NT, voici comment vous exprimer clairement :

- tous caractères avec accent grave (à À è È ì Ì ò Ò ù Ù) : AltGr-7, puis le caractère[2] ;
- tous caractères avec tréma ou circonflexe : faites comme pour les minuscules (¨ ou ^, puis le caractère) ;
- pour le Ç : Alt-128 ;
- pour le É : Alt-144 ;
- autres caractères avec accent aigu : pas de solution simple, mais heureusement pas d'usage en français ;
- si vous avez besoin d'un Ã, d'un ñ... : AltGr-2, puis le caractère ;
- pour œet Œ, Word (6 et +) connaît les principaux termes concernés et peut faire la correction automatique (si la fonction est active), ce qui permet de taper "oeil" et de récupérer "œil".

Un truc, justement : si votre traitement de texte dispose d'une fonction de correction automatique, enrichissez son dictionnaire avec les mots ayant des accents ou des collages peu aisés à faire manuellement ; vous n'aurez plus à vous préoccuper de l'accent, il sera ajouté automatiquement (on tape "canyon" - pas "canon", il y aurait ambiguï té -, on récupère "cañon", etc.).

Jean-Luc BLARY

P.S. Je vérifierai la présence des accents dans la prochaine LETTRE.

---

[2] *Mais attention : "A priori" est une expression latine, le A ne prend pas d'accent ; n'en profitez pas pour créer une faute !*