Les défaillances de l'informatique : une nouvelle menace ? l'exemple du bug de l'an 2000

Jean-François Colonna,
CMAP, École polytechnique *

Dire que l'informatique, les télécommunications et l'énergie constituent les infrastructures vitales de notre économie, de notre défense et de notre confort est d'une extrême banalité. Mais s'interroge-t-on suffisamment sur leurs interdépendances que personne ne maîtrise complètement, ainsi que sur leur robustesse et leur capacité à résister à des perturbations ? Et n'existerait-il pas aussi des risques internes, voire intrinsèques ? Ne seraient-elles que de gigantesques châteaux de cartes prêts à s'écrouler à la moindre occasion ? Finalement, ne construisons-nous pas sur du sable ? Nous allons montrer sur un exemple précis et d'apparence anodine, celui de la mauvaise gestion des dates dans les ordinateurs, qu'il en est malheureusement ainsi, et qu'un fâcheux grain de sable dans le silicium de nos microprocesseurs pourrait être à l'origine d'importantes défaillances, tout ceci sans faire preuve d'un catastrophisme excessif et en conservant présent à l'esprit que *mieux vaut prévenir que guérir.*

À L'APPROCHE DE L'AN 2000 a ressurgi un millénarisme qui contrairement à celui de l'an 1000 n'annonce point les splendeurs du Royaume de Dieu. Bien au contraire nos peurs sont aujourd'hui bien réelles, bien concrètes : le spectre de la pollution, les risques nucléaires..., planent au-dessus de nos têtes comme autant d'oiseaux de proie ; qui n'a pas tremblé à la vue de ces sous-marins nucléaires de l'ex-marine soviétique en train de pourrir dans les ports de la presqu'île de Kola ?

La science nous a appris au cours des siècles passés que le premier janvier de l'an 2000 serait un jour comme les autres dans l'histoire de l'univers. Malgré cela, ce jour pourrait rester inscrit dans l'histoire de l'humanité comme celui d'un désastre artificiel, aux conséquences *a priori* difficiles à imaginer.

Les phénomènes non linéaires naturels

Pour bien comprendre, il est bon de rappeler qu'il existe dans la nature des phénomènes caractérisés par le fait que leurs causes, généralement infimes, peuvent parfois s'amplifier démesurément et avoir alors des conséquences sans commune mesure avec ce qui leur a donné naissance.

La suite est immédiate : il est impossible théoriquement et pratiquement de faire à leur sujet des prévisions à long terme.

Les exemples de tels phénomènes sont nombreux, mais celui qui vient spontanément à l'esprit est celui qui a donné naissance à l'expression *faire boule de neige* : l'avalanche.

Les phénomènes non linéaires artificiels

Au cours des décennies passées, la science et la technologie nous ont offert des phénomènes non linéaires artificiels, par exemple, celui de l'échec du vol Ariane 501, le 4 juin 1996. Une petite anomalie dans un logiciel issu d'Ariane 4 et qui ne s'était jamais manifestée a contraint à l'autodestruction de la fusée après quelques dizaines de secondes de vol.

Ainsi, une infime anomalie s'est amplifiée jusqu'à avoir des conséquences ayant coûté un nombre appréciable de milliards de francs. Il convient, pour en terminer avec cet incident "exemplaire", de préciser que la compétence des ingénieurs concernés ne peut être mise en doute : c'est l'extrême complexité de ces systèmes qui rend impossible (à jamais ?) leur fiabilité absolue.

*Internet : http://www.lactamme.polytechnique.fr/an2000.html

La préhistoire de l'informatique

Né à la fin des années quarante, l'ordinateur avait à sa naissance vocation de machine à calculer et ses pères imaginaient alors que seuls quelques exemplaires de ces monstres nous seraient utiles. Pesant des dizaines de tonnes, occupant des centaines de mètres carrés au sol, ils étaient entourés d'armées de spécialistes. Puis, dans les années cinquante, les premières applications de gestion virent le jour. Au début des années soixante-dix, les performances et les capacités restaient encore très limitées : des mémoires centrales de quelques dizaines de kilo-octets et des disques de moins d'un méga-octet étaient la norme.

En ces temps reculés, point d'écran couleur et de souris servant à l'interaction avec les machines, mais uniquement les 80 colonnes des cartes perforées. Tout cela contraignait les programmeurs à beaucoup d'habileté et à certaines pratiques dont quelques-unes se sont perpétuées jusqu'à aujourd'hui. L'une d'entre elles fut la transposition d'une habitude bien antérieure à l'informatique, consistant à omettre les deux premiers chiffres des années. Mais si l'homme peut parler sans ambiguïté de *Mai 68* et de *la guerre de 70*, il en va malheureusement différemment pour nos chers ordinateurs...

L'informatique aujourd'hui

En l'espace de quelques années, le paysage informatique a bien changé. Cela est dû principalement à *l'intégration à très grande échelle (VLSI)*, technique qui permet de réaliser de façon fiable, automatique et économique, sur une surface de l'ordre du centimètre carré, des systèmes qui hier demandaient des dizaines de mètres cubes. Aujourd'hui, à côté des machines qui portent le nom d'ordinateur, rares sont les dispositifs, même parmi les plus quotidiens, qui n'intègrent pas l'une de ces "puces" (microprocesseurs, mémoires...). À cela s'ajoute la facilité avec laquelle tous ces dispositifs peuvent communiquer entre eux et échanger de l'information. Ainsi l'informatique actuelle est omniprésente et communicante.

La plupart des ordinateurs de la planète, qu'ils soient visibles (un PC sur un bureau) ou invisibles (un dispositif de contrôle de processus industriel) communiquent donc entre eux. Il est alors possible d'affirmer qu'ils forment un système non linéaire dont la complexité dépasse l'entendement et qui peut, dans certaines circonstances, devenir imprévisible et chaotique.

Quelques remarques concernant les développements informatiques

Avant de présenter le *bug de l'an 2000*, il semble utile de faire quelques remarques très générales permettant de mieux appréhender celui-ci, tout à la fois dans sa magnifique simplicité et son effroyable complexité...

L'expérience montre que la durée de vie des applications informatiques dépasse en général largement celle qui est envisagée par leurs concepteurs : il n'est pas rare de voir aujourd'hui des chaînes de traitement conçues il y a plus de vingt ans et encore opérationnelles. Malheureusement, leurs développements eurent lieu bien souvent sans que des méthodes strictes et bien documentées ne soient utilisées. Et cela n'a fait ensuite que s'aggraver au cours de leurs années d'exploitation car elles ont dû évoluer pour intégrer de nouvelles fonctionnalités et pour corriger les inévitables anomalies (*bugs* ou bogues en français) introduites par les concepteurs et les programmeurs.

Le travail de celui qui doit s'introduire dans de tels logiciels est semblable à celui du paléontologue qui cherche à déchiffrer le passé à partir des quelques traces qu'il a laissées. Ainsi, un programme peut être vu bien souvent comme un empilement instable d'un certain nombre de couches déposées de façon plus ou moins anarchique et précaire au cours des années. Les informaticiens préfèrent l'ajout à la suppression parce que cela est plus simple et moins risqué. La complexité et l'instabilité chronique qui en résultent ne peuvent qu'aller en s'amplifiant : l'informatique au quotidien n'est ni un art, ni une science, ce n'est que du bricolage...

Lorsqu'en 1876 à Boston Graham Bell inventa le téléphone, il n'imaginait absolument pas la portée de son invention. Cette absence de prévisibilité à long terme (voire à moyen ou à court terme), de même que l'inévitable existence d'anomalies et de risques, se retrouve tout naturellement lors de tout développement et de toute innovation informatiques. Ainsi, l'écran et le clavier facilitent l'interaction avec l'ordinateur, introduisent une nouvelle forme d'ergonomie et de confort ; mais inversement, nos vertèbres cervicales et lombaires ne vont-elles pas en souffrir, et ce temps réel qu'ils permettent ne va-t-il pas nous pousser à agir avec plus de précipitation, moins de réflexion et de discernement qu'en ces temps reculés où la réponse de la machine nous arrivait après de nombreuses heures d'attente fébrile ?

Lors de développements informatiques, les concepteurs sont amenés à faire des choix bien souvent arbitraires et certains d'entre eux s'avèrent être par la suite irréversibles. Ils créent ainsi de fortes relations de dépendance qu'il faut ensuite absolument respecter : c'est la notion de compatibilité. C'est le cas des codes utilisés pour représenter les caractères d'imprimerie ou encore les 80 colonnes des cartes perforées encore gravées dans de nombreux logiciels d'aujourd'hui... Dans tous les cas, l'ensemble des spécialistes reconnaissent unanimement qu'il serait plus que souhaitable de faire quelque chose : oui, mais quoi et surtout comment ?

Les systèmes informatiques sont généralement d'une complexité qui dépasse nos capacités de maîtrise. L'attitude la plus répandue consiste donc à éviter de toucher à ce qui semble fonctionner correctement, sauf cas de force majeure (cas du *bug de l'an 2000*). En effet, l'expérience montre que toute modification, même mineure, dans un logiciel fait courir des risques graves à celui-ci, ainsi qu'à l'intégrité de la mission qu'il remplit.

Enfin, contrairement à une idée très répandue, les ordinateurs possèdent des limites qui peuvent être soit intrinsèques

(la continuité – au sens mathématique du terme – n'existe pas dans un ordinateur), soit imposées par les concepteurs et les programmeurs, souvent de façon tout à fait arbitraire. Celles-ci peuvent être atteintes avec plus ou moins de facilité, mais cette situation entraîne un grand danger car généralement elle est ignorée et donc non prévue...

Le bug de l'an 2000

L'exemple bien trop réel que nous allons maintenant décrire va nous permettre d'illustrer concrètement ces différents points. Avant toute chose, précisons que, pour simplifier, nous supposerons les années écrites à l'aide de quatre chiffres décimaux et nous appellerons *pseudo-numéro de siècle* les deux premiers chiffres de cette représentation (par exemple 19 pour 1999). Précisons que ce pseudo-numéro de siècle ne correspond au véritable numéro de siècle que pour les années dont l'écriture se termine par 00 ; ceci implique en particulier que l'entrée dans le vingt et unième siècle et dans le troisième millénaire n'aura lieu que le premier janvier de l'an 2001 (l'an 2000 est la dernière année du vingtième siècle).

Le pseudo-numéro de siècle maltraité : déjà évoqué, ce premier problème est de loin le plus conséquent. La pratique ancienne de n'utiliser que les deux derniers chiffres des années, comme cela se retrouve dans le numéro INSEE des Français, fut donc transposée dans le monde de l'informatique. En effet, les dates, et donc les années, constituent des données omniprésentes dans les ordinateurs et en particulier dans les systèmes de gestion. Cela constitua donc un artifice capable de faire économiser 50 % des octets nécessaires à la mémorisation des années, tout en permettant de faire correctement les opérations arithmétiques utiles et en particulier les calculs de durées ; par exemple, un enfant né en [19]90 aura [19]99 – [19]90 = 9 ans en 1999. Malheureusement, cette propriété ne se conservera pas au passage de la Saint-Sylvestre 1999 ; ce même enfant aura [20]00 – [19]90 = –90 ans en 2000. Il nous est évident qu'une telle valeur est incorrecte, puisqu'un âge ne peut être négatif, mais comment se comportera un programme en sa présence ? Il est impossible de répondre en toute généralité à cette question, mais il est certain que ces circonstances n'ayant en général pas été prévues, le résultat sera *a priori* imprévisible...

Ce choix n'était en fait qu'un moyen provisoire destiné à compenser la faible capacité des machines préhistoriques. Les programmeurs d'alors n'imaginaient pas que cette façon de représenter les années se perpétuerait au cours des décennies suivantes ; il est donc logique qu'ils n'en aient pas prévu les funestes conséquences.

Cette pratique est extrêmement répandue, même sur des systèmes informatiques donnant l'illusion d'une gestion des années sur quatre chiffres ; l'édition de " 1998 " sur un relevé bancaire ne prouve rien : les deux caractères " 19 " peuvant avoir été ajoutés *a posteriori* lors de l'impression. De plus, les formats de la date varient d'un pays à l'autre ; parmi trois valeurs donnant chacune sur deux chiffres le jour, le mois et l'année, pour retrouver cette dernière, l'algorithme fréquemment utilisé consiste à rechercher la valeur strictement supérieure à 31. Il est évident qu'à partir du premier janvier [20]00 cette pratique sera défaillante, tout au moins jusqu'en [20]32. Ainsi, ces deux exemples montrent que la représentation et la manipulation des dates dans les ordinateurs ne sont ni aussi simples, ni aussi logiques que le bon sens le laisserait supposer...

Ainsi, une gestion des années sur deux chiffres pourra conduire à des calculs de durée erronés, à des relations d'ordre temporel inversées ou encore à des actions automatiques déclenchées au mauvais moment ou pas du tout. Malheureusement, le problème du codage des années sur deux chiffres n'est pas le seul : quatre autres l'accompagnent...

Les dates inaccessibles : pour représenter dans les programmes certaines conditions particulières telle une erreur ou la fin d'une liste d'objets contenant des dates, une pratique très répandue a consisté à utiliser une date (9/9/99...) ou une année (99...) particulières, les rendant par là même inaccessibles. Ces valeurs furent choisies parce que d'une part elles demandaient la frappe d'une touche unique sur un clavier et que d'autre part elles semblaient appartenir à un futur bien lointain. Cette anomalie n'a provoqué que quelques dysfonctionnements mineurs : par exemple, elle a rendu impossible la délivrance de passeports provisoires en Suède dans les premiers jours de 1999.

L'arithmétique sur les dates : dans les systèmes informatiques de gestion, certaines données numériques, comme les dates, sont préférentiellement codées sous forme de chaînes de caractères. Ainsi, l'année " 1998 " sera représentée par la suite {" 1 ", " 9 ", " 9 ", " 8 "}. Lorsque des calculs doivent être effectués sur ces représentations, le programmeur doit faire accomplir explicitement à la machine tout ce qui est fait lors d'un calcul manuel et en particulier la propagation des retenues. Oublier cela lors de la détermination de l'année suivante peut donner fréquemment des résultats corrects (" 1998 " suivie de " 1999 ") et de temps en temps des surprises (" 1999 " suivie de " 199: "). De telles anomalies ont déjà été rencontrées en Italie, lors d'opérations portant sur des cartes bancaires, à la fin des années quatre-vingt.

La capacité limitée : il ne suffit pas à un ordinateur d'archiver des dates " statiques ". Pour accomplir ses missions, il doit aussi être capable de déterminer en permanence la date et l'heure courantes. Cela est réalisé à l'aide d'un instant de référence arbitraire et d'un compteur incrémenté périodiquement d'une unité. Que se passe-t-il si ce dernier est sous-dimensionné par rapport à l'usage prévu ? Le système GPS *(Global Positioning System)* donne un exemple d'une telle anomalie. Celui-ci utilise une constellation de satellites et nécessite, au niveau des mobiles qui l'utilisent, des récepteurs. Ces derniers ont besoin de la date et ils la déterminent, en particulier, à l'aide d'un compteur de semaines qui ne peut compter que de 0 à 1 023. Ainsi, tous les récepteurs GPS sont repassés à 0 le 21 août 1999 à 23 : 59 : 47 UTC en ne provoquant aucune catastrophe, mais seulement quelques anomalies dans des systèmes grand public de navigation (maritime et automobile). Malgré cela, le choix d'une telle limite (de l'ordre de vingt ans) dans un dispositif d'origine militaire semble tout à fait arbitraire et mystérieux !

De telles anomalies se retrouvent potentiellement dans la plupart des ordinateurs. Remarquons enfin que cette non-extensibilité des ordinateurs concerne aussi de nombreuses autres informations : par exemple le stockage des numéros de sécurité sociale ou encore de téléphone.

Les années bissextiles : les unités de mesure du temps reposent historiquement sur des considérations astronomiques. C'est ainsi que l'année correspond à la durée de la révolution de la Terre autour du Soleil, soit environ 365,2422 jours, ce qui n'est pas un nombre entier. Afin que l'année astronomique reste en phase avec le calendrier, la seule solution est de donner à cette dernière une longueur variable de 365 ou de 366 jours. Ainsi, périodiquement, des années dites bissextiles sont introduites. Jusqu'à la réforme imposée par Grégoire XIII en 1582, les années bissextiles revenaient tous les quatre ans, ce qui donnait une année moyenne de 365,25 jours donc légèrement trop longue. Le calendrier, dit grégorien, fut donc destiné à compenser cette anomalie au prix d'une définition plus complexe de la bissextilité. Une année sera bissextile si elle est divisible par 4 sauf si elle est séculaire (c'est-à-dire divisible par 100) et non divisible par 400. Ainsi, 2000 et 1996 sont bissextiles, alors que 1900 et 1999 ne le sont pas.

L'ensemble de ces critères est généralement méconnu du public, même cultivé, et aussi, bien souvent, absent des ouvrages de référence. C'est ainsi que *Le Petit Robert*, dans son édition de 1976, donne à la page 179 la définition *Bissextile : se dit de l'année qui revient tous les quatre ans et dont le mois de février comporte 29 jours!* Cela a donné naissance à quelques logiciels et matériels présentant des anomalies graves leur faisant considérer que l'an 2000 n'était pas bissextile.

Ainsi, ce qui est appelé communément le *bug de l'an 2000* est en fait l'union d'un certain nombre d'anomalies de gestion des dates dans la plupart de nos ordinateurs, tant au niveau des matériels que des logiciels. Ces anomalies, qui sont plus le résultat de choix anciens que d'erreurs, sont particulièrement élémentaires à spécifier. Cette constatation a deux conséquences immédiates : la première concerne l'incrédulité de ceux qui découvrent ces problèmes, alors que la seconde fait croire à ces derniers que la solution est, elle aussi, élémentaire. De par la nature non linéaire de ces "phénomènes", il n'en est rien, tout au contraire.

Les responsables

Mais avant de décrire les difficultés correspondantes, il est bon de rechercher les responsabilités et d'essayer de comprendre comment il est possible que la plupart des acteurs concernés aient attendu la dernière minute pour agir, alors que ces problèmes sont connus depuis plus de vingt ans, dans le milieu bancaire en particulier.

Les informaticiens semblent bien évidemment être les coupables tout désignés. En effet, ils n'ont pas su voir à long terme les conséquences de leurs choix et ils n'ont pas été capables d'écrire des programmes sans erreurs. Ils n'ont pas eu ensuite la franchise d'avouer leurs fautes, le courage d'expliquer les conséquences néfastes de ces dernières et la force de demander les énormes moyens indispensables pour effectuer les réparations utiles... *Rien ne sert de courir, il faut partir à point*, regrettons donc que le nécessaire ne fût point fait lorsqu'il en était encore temps ; fait donc posément, correctement et non point dans la précipitation, elle-même source évidente de nouvelles anomalies. Mais arrêtons là de les accabler : en effet, l'absence de prévisibilité et de fiabilité est une constante dans toutes les activités humaines.

Malgré cela, pourquoi les grandes associations professionnelles que sont l'ACM *(The Association for Computing Machinery)* ou encore l'IEEE *(The Institute of Electrical & Electronics Engineers)* n'ont-elles pas tiré le signal d'alarme plus tôt ? Pourquoi a-t-il fallu attendre ces derniers mois pour voir publier de trop brefs articles sur ce sujet ? Est-ce parce que la maintenance n'est pas une activité digne d'intérêt ou pire parce que qu'elle ne fait pas partie de l'univers de l'informatique ? Mais les ordinateurs ne sont pas des objets abstraits et simples ; ce sont des machines complexes, composées de nombreuses entités coopérantes présentant toutes, de façon plus ou moins chronique, des anomalies. La maintenance fait donc partie de la vie de l'informaticien et doit donc être traitée avec tout le respect dû à ce qui finalement conditionne le bon fonctionnement quotidien de nos machines. Enfin, l'ampleur des difficultés auxquelles nous sommes aujourd'hui confrontés aurait pu susciter des vocations, par exemple dans le monde de l'intelligence artificielle. Malheureusement, il est trop tard !

Ainsi, les informaticiens (des temps héroïques...) ont des circonstances atténuantes. Mais qu'en est-il des utilisateurs eux-mêmes ? En effet, nombreux sont ceux qui connaissaient ces problèmes depuis longtemps : les banquiers, par exemple, ne font-ils pas des prêts à vingt ou trente ans ? Ils savaient donc dès les années soixante-dix que l'absence du pseudo-numéro de siècle était nuisible. Il est certain qu'alors seul le strict nécessaire a été fait ; n'aurait-ce pas été faire preuve de clairvoyance que d'extrapoler vers les domaines non encore critiques ? Notons de plus qu'alors l'informatique était beaucoup moins omniprésente, ce qui rendait ces problèmes plus faciles à résoudre. Pourquoi enfin n'ont-ils pas fortement fait pression sur les grands constructeurs dont ils sont parmi les plus gros clients ? Cela aurait certainement accéléré l'arrivée sur le marché de systèmes "compatibles" et peut-être étouffé l'incendie avant qu'il ne se propage ! Là aussi, il est malheureusement trop tard !

Que dire maintenant de la responsabilité des médias ? Elle semble considérable puisqu'un événement n'existe que s'il est relaté ; son importance est alors proportionnelle au nombre de lignes et de minutes qui lui est consacré. Il est donc regrettable de constater le trop long silence de la presse, en particulier informatique, tant nationale qu'internationale.

Enfin, le monde politique a lui aussi fait preuve d'une absence, mais encore faut-il que l'information remonte au sommet de la pyramide... En France, il a fallu attendre le 20 février 1998 pour qu'une mission présidée par Gérard Théry soit constituée. L'ampleur du problème est telle, qu'elle aurait demandé d'une part une coordina-

tion internationale au plus haut niveau des États et d'autre part que des informations soient communiquées aux citoyens afin qu'ils sachent l'état réel dans lequel se trouvent les administrations, les services publics et de santé, la distribution des énergies, les systèmes de transport, la défense nationale... Les plans de secours mis en place fonctionneront-ils (le malheureux accident survenu dans la nuit du 25 au 26 septembre 1998 à l'hôpital Édouard-Herriot de Lyon est particulièrement instructif) ? En cas d'incidents, le retour à la normale sera-t-il garanti rapidement ? Est-on capable de revenir provisoirement au papier et au crayon ? Autant de questions que tous se posent naturellement ; l'absence de réponses ne peut qu'engendrer plus d'inquiétude et déclencher des mouvements de panique dont les conséquences pourraient être bien plus graves que celles des incidents réels, peut-être inexistants d'ailleurs... Ces réactions possibles pourraient être déclenchées par des pénuries artificielles portant, par exemple, sur l'argent liquide ou encore les biens de première nécessité (dans les premiers jours de janvier 1999, quelques bureaux de poste français ont été pris d'assaut à la suite de l'indisponibilité du système informatique gérant les comptes d'épargne).

À cela, il convient d'ajouter le calendrier de la mise en place de l'euro qui a été défini, sans apparemment prendre en compte le fait évident qu'il est difficile, pour ne pas dire impossible, de mener à bien et correctement deux projets informatiques d'une telle ampleur simultanément. Enfin, sachant le déficit généralisé en informaticiens, la loi française sur les 35 heures n'est-elle pas incompatible, provisoirement, avec l'urgence de la situation (notons au passage qu'il ne s'agit pas ici de critiques envers l'euro et les mesures pour l'emploi, mais simplement de remarques de bon sens : serait-il raisonnable de demander à des pompiers qui éteignent un incendie, d'abandonner leur tâche, sous prétexte qu'il est dix-sept heures trente ?).

Finalement, ne serions-nous tous pas à la fois responsables et coupables ? En tout cas, quelle que soit la réponse donnée à cette interrogation, il paraît évident que nous devons tous nous sentir concernés, tant à l'intérieur des structures professionnelles auxquelles nous appartenons, qu'en tant que citoyen.

Les solutions

Tous les problèmes que nous venons de décrire sont élémentaires et ne demandent aucune compétence particulière pour être compris. Alors où est la difficulté ? Les systèmes informatiques sont semblables aux icebergs : 10 % de réel émergé (le matériel) et 90 % de virtuel immergé (le logiciel) ; or autant il est facile d'appréhender la complexité matérielle car visible, autant cela est difficile pour la complexité logicielle puisque intangible. En fait, les informaticiens sont en face d'un problème assez similaire à celui qui consisterait à leur demander de compter les grains de sable sur une plage.

En ce qui concerne la gestion des dates dans nos ordinateurs, la situation est identique : les opérations individuelles de correction sont élémentaires ; la difficulté réside uniquement dans leur nombre. Il y aurait actuellement, de par le monde, environ cent milliards de lignes de programmes concernées. Un mythe répandu consiste à affirmer l'existence d'une solution miracle. Malheureusement, celle-ci n'existe pas ; en effet, pour qu'un outil universel de mise à jour puisse être conçu, il aurait fallu qu'au cours des décennies passées, des méthodes strictes de développement aient été utilisées. Il n'en est rien et par exemple, le nombre de façons de représenter une date dans un ordinateur est élevé... Il convient d'ajouter que même si cette solution existait elle ne dispenserait pas de la première étape incontournable : l'inventaire.

En effet, avant de corriger les matériels et les logiciels, il est nécessaire de disposer de leur liste exhaustive. Or, aussi paradoxal que cela puisse paraître, ces inventaires, en général, n'existent pas ; il faut donc commencer par les établir. L'expérience montre que, pour une banque, plusieurs dizaines de milliers de logiciels sont utilisés et ce dénombrement demande parfois plus d'une année pour être réalisé proprement ! Malheureusement, l'informatique concernée par le *bug* n'est pas faite, loin s'en faut, d'ordinateurs posés sur nos bureaux et contenant quelques logiciels bien rangés... Non, elle est partout : aussi bien dans les ascenseurs, les climatiseurs, les robots de fabrication... Et c'est certainement là que le risque est le plus grand : dans cette informatique qualifiée d'*enfouie* qui assure en permanence le bon fonctionnement de tout notre environnement et dont nous ignorons presque tout !

Il n'y a donc pas d'autre solution que de passer au peigne fin ces milliards de lignes de programmes et ces millions de systèmes spécialisés : nous sommes à la recherche d'aiguilles (les dates) aux formes et en nombre inconnus dans une meule de foin. Si la prise de conscience avait eu lieu en temps utile, il aurait été envisageable de remplacer l'ancien par du nouveau. Mais un point de non-retour a été franchi et donc seules les corrections restent envisageables. En réalité, la situation est en général plus précaire que cela : lorsque sont pris en compte les ressources humaines disponibles, le temps restant et la quantité de travail à accomplir, il apparaît que, bien souvent, même ce " rafistolage " exhaustif est impossible ! S'impose alors un tri permettant de fixer les priorités : quels sont les logiciels et les matériels qui sont vitaux tant pour l'homme que pour l'entreprise et quels sont ceux qui le sont moins ? Tout ceci doit être évidemment associé à une analyse des risques encourus, ainsi qu'à la mise en place de plans de secours.

Mais en quoi consistent donc ces " rafistolages " ? La bonne solution au problème du codage des années sur deux chiffres consisterait à étendre les zones de mémoire destinées à stocker et à manipuler des années. Combien de chiffres seraient nécessaires ? Quatre paraissent *a priori* raisonnables, mais ce problème ne se reposera-t-il pas alors en 9999 ? Cette remarque en forme de boutade est destinée à nous rappeler deux points essentiels : les limites arbitraires (et souvent irréversibles) imposées par les programmeurs et le manque de visibilité à long terme. Quatre chiffres sont évidemment suffisants ; malheu-

reusement, la complexité des opérations correspondantes est rédhibitoire. D'autres solutions doivent donc être mises en œuvre. L'une des plus répandues consiste à choisir, dans un certain contexte, une année dite *pivot* codée sur quatre chiffres, par exemple 1960 ; ensuite toute année AA sera étendue sur quatre chiffres uniquement là où cela est utile. Cette extension se fera à l'aide d'un test simple : si AA est supérieure à [19]60, il lui est substitué 19AA et 20AA dans le cas contraire. Cette solution présente un certain nombre de dangers. Le premier concerne la cohérence ; en effet, il est important que la même date pivot soit utilisée par tous les systèmes en relation : mais est-ce possible ? Non en toute généralité, car cela signifierait à la limite que l'ensemble de la planète utilise la même. Le second danger est plus lointain ; en effet, que se passera-t-il lorsque tous les obstacles qui s'annoncent devant nous auront été franchis avec plus ou moins de succès ? La réponse est déjà connue : nous commettrons les mêmes erreurs que par le passé et nous oublierons nos rafistolages. Or la notion même d'année pivot contient sa propre limite, de l'ordre de quelques dizaines d'années. Ainsi, sans résoudre les vrais problèmes, nous sommes en train de mettre en place des bombes à retardement qui donneront bien des soucis à nos successeurs !

Dans tout projet informatique, il est essentiel de vérifier la conformité de la réalisation par rapport au cahier des charges et ces tests occupent environ 50 % du temps global de développement. Pour le *bug*, deux types de tests doivent être accomplis ; d'abord des tests de non-régression qui permettent de vérifier que les corrections apportées ont maintenu l'intégralité des fonctionnalités antérieures. Au niveau d'une banque, par exemple, cela se fait en mettant en place un site indépendant du site opérationnel, puis en comparant leurs résultats ; cette approche est malheureusement plus statistique qu'exhaustive. Ensuite, des tests de compatibilité doivent être conduits afin de vérifier que les problèmes de date ont été résolus. Certaines dates particulièrement critiques doivent être examinées à la loupe (le jeudi 09/09/1999, le lundi 03/01/2000, le mardi 29/02/2000, le mercredi 01/03/2000, le mardi 10/10/2000...). Cette série de tests est particulièrement délicate, personne ne sachant en toute généralité ni ce qu'il faut tester, ni comment vieillir les jeux de tests sans introduire de biais pervers.

Évalué très grossièrement, le coût des corrections est au niveau de la planète de l'ordre de mille milliards de dollars ! Mais il y a deux raisons poussant à augmenter cette évaluation : d'une part l'absence d'exemple de grands projets informatiques pour lesquels les budgets initiaux ont été respectés et d'autre part les dédommagements dus aux inévitables dégâts et catastrophes : ils sont actuellement évalués à deux mille milliards de dollars ! Les assureurs ont triplement à s'inquiéter : ils sont tout à la fois parmi les plus gros utilisateurs de l'informatique, les plus sensibles à ces problèmes de gestion de dates et les plus concernés de par leur fonction ; ils n'ont d'ailleurs pas manqué de rappeler (fort justement d'ailleurs !) que le *bug* ne comportait aucun caractère aléatoire, l'arrivée du 01/01/2000 étant connue depuis quelques siècles...

Pour la première fois dans l'histoire de l'informatique, des projets ont des dates d'achèvement immuables. Or rares sont les projets qui se sont achevés à temps, tout en respectant les budgets et les fonctionnalités.

Enfin, nous sommes en présence d'un magnifique exemple d'*effet domino* : de par l'interconnectivité généralisée, il ne suffit pas qu'une entreprise soit prête ; il importe que l'intégralité de ses clients et de ses fournisseurs avec lesquels elle entretient en permanence des relations informatiques soient eux aussi prêts, sous peine de risques graves de "pollution". Ainsi, en plus du délicat travail de mise à niveau interne, chacun doit procéder à un colossal travail externe de sensibilisation... Le monde de la banque et de la finance est particulièrement sensible à ce phénomène ; un scénario catastrophe, indépendant du *bug* et des plus réalistes malheureusement, consiste à imaginer sur une place boursière, n'importe où dans le monde, un ordinateur défaillant qui provoquerait une anomalie qui se propagerait ensuite, de façon quasiment irréversible, à une vitesse proche de celle de la lumière, causant des ventes massives, un krach, une récession...

Conclusion

Les problèmes de gestion des dates dans les ordinateurs, même s'ils sont élémentaires à spécifier, constituent une réelle menace pour l'intégrité de nos économies, voire même pour la sécurité de nos États. Il n'y a *a priori* aucune activité, ni personne qui en soit réellement à l'abri, ne serait-ce qu'à cause des interdépendances. Malheureusement il ne s'agit là que d'un exemple concret d'une catastrophe possible parmi d'autres, qui gisent potentiellement dans nos technologies, prêtes à éclater à tout moment.

Nos réalisations reposent aujourd'hui toutes sans exception sur l'informatique. Pratiquée au quotidien, celle-ci est à des années-lumière de la pureté des 0 et des 1 de l'algèbre de Boole et de ses règles absolues : elle n'est ni Art, ni Science, mais bricolage. Bricolage de génie parfois, ses réussites sont là pour en témoigner : l'ordinateur est en particulier un outil fantastique pour amplifier nos facultés intellectuelles et nous permettre des progrès sans précédent dans le domaine de la connaissance. Mais bricolage approximatif le plus souvent, qui pourra nous conduire vers d'autres catastrophes aux conséquences incalculables dans un avenir plus ou moins proche.

Le *bug* est une donc une opportunité à saisir : le temps n'est-il pas venu de s'interroger sur cette complexité dont nous sommes à l'origine, que parfois nous ne maîtrisons plus et qui nous échappe alors ? Soyons en tout cas conscients de notre responsabilité aujourd'hui et œuvrons chacun à notre niveau. ■

Bibliographie :

Le Bug de l'an 2000 – Comprendre l'informatique jusqu'à ses dysfonctionnements, Bernard Aumont et Jean-François Colonna, Flammarion, Paris, mars 1999.