# Proposition de structuration historique des concepts de la pensée informatique fondamentale

Yannis Delmas-Rigoutsos

Yannis Delmas-Rigoutsos

Laboratoire TECHNÉ EA 6316, Université de Poitiers, France
yannis.delmas@univ-poitiers.fr

# Proposition de structuration historique des concepts de la pensée informatique fondamentale[1]

**Résumé**

*Les programmes de 2015 de l'enseignement obligatoire français (MENESR, 2015b) placent celui-ci dans le cadre du* Socle commun de connaissances, de compétences et de culture *(MENESR, 2015a), définissant ainsi un objectif pédagogique large, relevant de la culture commune de base du pays. Pour l'informatique, comme pour de nombreuses autres matières, l'approche disciplinaire, savante, n'est guère adaptée à cet objectif, pas plus qu'une approche qui serait seulement opératoire ou instrumentale, limitée, comme par le passé, à des usages des technologies usuelles d'information et de communication (TUIC).*

Qu'est-ce que l'informatique ? *Des auteurs précédents (Papert, 1996 ; Denning, 2003 ; Wing, 2006, 2008 ; Dowek, 2011 ; Hartmann et al., 2012) ont suggéré plusieurs approches et structurations du champ, visant des objectifs différents, toutefois. Pour définir les besoins de l'honnête Homme, il convient de reprendre cette question sur de nouvelles bases.*

*En adaptant une définition de Schwill (1993), nous proposons une cartographie de ce que pourraient être les concepts les plus fondamentaux de la pensée informatique, susceptibles de contribuer à la formation générale de l'esprit humain, au même titre que les mathématiques ou les humanités. Pour cela, nous nous appuyons sur l'histoire de l'informatique et du numérique et soulignons également les liens que ces concepts peuvent entretenir avec d'autres savoirs. Ces liens ne sont pas seulement culturels, ils permettent aussi d'entrevoir les liens qui peuvent exister entre disciplines et placer ainsi l'informatique dans un cadre scientifique élargi.*

**Mots clés :** didactique de l'informatique, enseignement scolaire, histoire de l'informatique, pensée informatique

1 Recherche conduite dans le cadre du programme « didactique et apprentissage de l'informatique à l'école » (DALIE) soutenu par l'ANR (14-CE24-0012–01).

# 1 Introduction

Les technologies numériques sont partout, ambiantes, remodèlent les rapports humains et le rapport aux connaissances. Comme Lawrence Lessig l'a abondamment défendu avec son célèbre « Code is law » (2000), les artefacts numériques constituent des lois pour la société informatisée. Puisqu'il s'agit de vecteurs de contrôle social, de sédimentation de choix, de stratégies et d'idéologies, le Citoyen est face à une alternative simple : « coder ou être codé », autrement dit être éclairé sur le cadre technique qui l'entoure ou se rendre, renoncer à son autorité pour devenir leur sujet. Mais que signifie ce « coder » pour la formation de l'esprit du Citoyen ?

Avec le développement considérable de la science informatique, d'une part, et l'extension de l'enseignement de « l'informatique » en deçà de l'enseignement supérieur, d'autre part, se pose la question pratique de la définition de *curricula*. Comme l'argumente Peter Denning (2003), ancien président de l'Association for computing machinery (ACM)[2], l'approche par domaines de spécialisation universitaire n'est plus adéquate : ils sont une trentaine, chacun avec ses apports fondamentaux, ses idées-forces, etc. À l'extrême inverse, celui-ci ne se montre pas plus satisfait d'approches se limitant aux grandes idées (*great ideas*), trop restrictives, par construction. Il propose donc de poser différemment la question curriculaire, en distinguant quatre niveaux d'approche : 1) les domaines d'application, l'informatique étant largement une discipline ancillaire, 2) les domaines de l'informatique, *core technologies*, qui s'attachent à la conception des différents types de traitements, 3) les principes d'organisation humaine de l'acte de conception et 4) les grands principes du corpus théorique de l'informatique, qu'il nomme « *computing mechanics* » par parallélisme avec la physique. Son objet est bien de définir les contours de ce qu'est épistémologiquement le traitement de l'information (*computation*) – quelle est sa mécanique interne. C'est cette même question que nous posons également ici.

Chaque réponse à cette question, chaque découpage, chaque nomenclature, chaque cartographie est, en soi, légitime ; il reste à en déterminer la pertinence. Or, dire pertinence, c'est souligner que toute

2 L'ACM est la principale société savante de l'informatique, couvrant à la fois ses aspects scientifiques et pédagogiques.

catégorisation relève d'un arbitrage, d'une finalité. Peter Denning (2003) visait essentiellement à structurer les concepts de l'informatique savante. Gilles Dowek (2011), à l'occasion du colloque Didapro de 2011, fit, lui, une proposition visant le secondaire supérieur (en France, le lycée). Pour ce qui nous concerne ici, nous souhaitons viser ce que nous considérons comme le socle fondamental de formation de l'esprit et de culture que devrait maîtriser tout honnête Homme contemporain. Aux termes des instructions officielles pour l'enseignement obligatoire français, ce cadre est défini par le *Socle commun de connaissances, de compétences et de culture* (MENESR, 2015a) et règle les enseignements élémentaire et secondaire inférieur (collège) ; c'est donc ce que nous viserons. Cet horizon, beaucoup plus large, est celui de la culture commune de base du pays. Comme ces programmes, nous nous plaçons en rupture avec leurs prédécesseurs récents, qui ne concevaient le numérique qu'instrumental et opératoire, dans le champ restreint des technologies usuelles d'information et de communication (TUIC).

## 2 Méthode et travaux antérieurs

### *2.1 Propositions antérieures*

Avant de nous attacher à notre méthode, rappelons quelques propositions antérieures récentes[3].

Jeannette Wing s'attache, avant tout, à défendre l'intégration de la pensée computationnelle (*computational thinking*) au socle intellectuel fondamental (2006) et son insertion curriculaire systématique dans l'enseignement scolaire (2008). Par ce terme de « pensée computationnelle », elle désigne les outils intellectuels qui relèvent d'une compétence fondamentale, au-delà des questions d'artefacts et/ou de routines techniques. Ses principales caractéristiques sont les suivantes :

3 Nous laissons ici de côté l'ouvrage de Seymour Papert (1996), qui demanderait un développement plus conséquent.

- « *The essence of computational thinking is abstraction*. » Ceci inclut : reformuler un problème, p. ex. par réduction, plongement, transformation ou simulation ; utiliser l'abstraction et la décomposition ; s'attacher à l'encapsulation (*separation of concerns*) ; choisir la représentation la plus adaptée pour modéliser un problème ; utiliser des invariants pour décrire le comportement d'un système ; modulariser, pour anticiper différents usages possibles ; précharger et mettre en cache des données en vue d'une utilisation future.
- Ceci inclut également de grandes méthodes de conception de programmes : penser récursivement ; traiter en parallèle ; interpréter le code comme de la donnée ou la donnée comme du code ; penser la vérification de type de données comme une généralisation de l'analyse dimensionnelle [en physique] ; saisir les avantages et inconvénients de donner plusieurs noms à une entité ; mesurer les coûts et bénéfices de l'adressage indirect et de l'appel de procédures. C'est également être capable de percevoir l'esthétique d'un programme.
- « *Because our abstractions are ultimately implemented to work within the constraints of the physical world, we have to worry about edge cases and failure cases*. » Notamment : raisonner en termes de prévention, protection et récupération des pires scénarios au moyen de la redondance, du confinement des dégâts et de la correction d'erreurs ; « *call gridlock deadlock* »[4] ; définir des contrats d'interfaces ; apprendre à éviter les situations de concurrence (*race conditions*) en cas de synchronisation.
- Enfin, c'est utiliser un raisonnement heuristique pour parvenir à une solution ; planifier, apprendre et programmer (*schedule*) face à l'incertitude ; utiliser des techniques de recherche de données ; utiliser de grandes quantités de données pour accélérer un traitement ; arbitrer entre temps et espace, entre puissance de calcul et capacité de stockage.

Peter Denning (2003) distingue au sein de l'informatique (*computing mechanics*) cinq registres (*windows*), chacun centré sur un objectif (*concern*) et s'appuyant sur des histoires (*stories*, ce qu'on peut raconter sur la discipline et que l'on retrouve dans les domaines technologiques). Ces registres de l'informatique savante sont :

4 Cette formulation est intraduisible en français : un *gridlock* est un interblocage dans le contexte de la circulation routière et un *deadlock* un interblocage de plusieurs processus informatiques.

- Le calcul ou traitement, en anglais *computation*, centré sur la question de ce qui peut être calculé, dont : algorithme, structures de contrôle, structures de données, automates, langages, machines de Turing, machines universelles, complexités de Turing et de Chaitin, autoréférence, logique des prédicats, approximations, heuristiques, calculabilité, traductions, réalisations physiques ;
- La communication, transmission de messages d'un point à un autre, dont : transmission de données, entropie de Shannon, encodage sur un médium, capacité d'un canal, réduction du bruit, compression de fichier, cryptographie, réseaux à commutation de paquets reconfigurables, vérification d'erreur de bout en bout ;
- La coordination, coopération de multiples agents, dont : de personne à personne (boucles d'action, *workflows* […]), personne-machine (interface, entrée, sortie, temps de réponse), machine-machine (synchronisation, concurrence, interblocage, sérialisabilité, actions atomiques ;
- Automatisation (*automation*), dont : simulation de tâches cognitives, problématiques philosophiques liées à l'automatisation, ingénierie cognitive (*expertise*) et systèmes experts, amélioration (*enhancement*) de l'intelligence, tests de Turing, apprentissage automatique et récognition, bionique ;
- Mémorisation (*recollection*), dont : hiérarchies de stockage, localité de la référence, mise en cache, espace d'adressage, nommage, partage, emballement, recherche, accès par nom, accès par contenu.

Gilles Dowek (2011), qui s'intéresse au niveau secondaire supérieur (lycée), insiste sur les notions fondamentales suivantes, souvent utilisées de conserve dans toute conception informatique :

- Algorithme, dont : non exclusivité de l'informatique ; algorithmes déterministes ou non ; nécessité de la représentation symbolique ;
- Machines, pour exécuter les algorithmes, dont : leurs limites physiques ; ordinateur, réseau, machines parallèles et systèmes embarqués ; concept de machine universelle ;
- Langage, dont : langages pour écrire des algorithmes ; langages de programmation ; antériorité historique par rapport à l'ordinateur ; langage au sens général : langues naturelles et formelles ;
- Information, dont : représentation symbolique ; « numérisation » qui serait mieux désignée comme une mise en symboles ; codage, compression et chiffrement ; archivage et restitution.

Gilles Dowek « insiste sur l'importance de respecter l'équilibre entre ces différents concepts dans la conception d'un programme d'enseignement de l'informatique au lycée ».

En ne citant que ces seuls auteurs, nous constatons déjà une grande diversité de choix. Nous constatons également combien certaines notions semblent tout à fait inaccessibles à un niveau d'enseignement élémentaire ou secondaire inférieur, faute d'un arrière-plan intellectuel, et d'une capacité d'abstraction, suffisants. Nous allons maintenant nous efforcer de structurer un ensemble d'idées fondamentales, accessibles dès l'enseignement primaire ou secondaire inférieur (collège) et suffisamment articulées entre elles pour permettre aux élèves de se forger une représentation globale. Nous ne prétendons pas à l'exhaustivité, mais le cadre que nous proposons demandera déjà un effort de formation des enseignants concernés.

### *2.2 Ce qu'est une idée fondamentale*

Jerome Bruner propose dans son précis d'éducation (1960) une notion très générale de thèse fondamentale. Cette notion est systématisée et adaptée à l'informatique par Andreas Schwill (1993), pour qui une idée est fondamentale, si elle respecte quatre grands critères, d'horizontalité, de verticalité, de pérennité et de sens. Hartmann *et al*. (2012) ajoutent un critère pédagogique que nous intégrerons aux critères de verticalité et de sens, que nous élargirons. Nous définissons ici une *idée fondamentale* par trois critères :

- *Extension* : elle apparaît sous différentes formes dans au moins deux domaines (critère d'horizontalité) et peut être décrite à un niveau élémentaire comme avancé (critère de verticalité) ;
- *Généralité* : elle est pertinente pour s'appliquer ou se transférer à des situations réelles, matérielles ou intellectuelles hors du seul champ de l'informatique (élargissement du critère de sens) ;
- *Influence* : son histoire montre une influence durable sur les sciences du numérique (critère de pérennité) ; elle a été féconde dans un autre domaine du savoir ou présente un enjeu sociétal particulier.

Une idée fondamentale doit se retrouver, à un degré ou à un autre, dans la plupart des dispositifs informatiques. Aucune situation écologique n'est

donc pure. Ceci n'est pas nécessairement le cas des situations pédagogiques, qui peuvent aussi bien être pures, ou réalistes, ou relier telle idée avec tel autre champ du savoir, ou encore transférer une idée issue de l'informatique à une situation qui n'en relève pas.

### *2.3 Organisation historique*

Ces idées fondamentales ne sont pas toujours marquées historiquement. Elles prolongent parfois des réflexions, voire des théories, antérieures à l'informatique elle-même. Nous organiserons cependant notre propos en suivant trois grandes périodes de l'histoire de l'informatique en reprenant la périodisation de David Fayon (2010), dans le champ économique, que nous étendons à d'autres aspects (*cf.* Delmas-Rigoutsos, 2014). Nous souhaitons ainsi seulement faciliter la compréhension de leurs relations.

Tableau 1 : Grandes périodes de l'histoire de l'informatique.

| | **Années 1930–1940** | **Années 1940–1950** | **1947–1980** | **1974–2005** | **1999–** |
|---|---|---|---|---|---|
| **Mouvement dominant** | logique mathématique | pionniers | première industrie | loi de Moore | numérique ambiant |
| **Artefacts dominants** | modèles de calcul (dont la machine de Turing) | quelques *computers* | dizaines de milliers d'ordinateurs centralisés | dizaines de millions d'ordinateurs personnels | dizaine(s) de milliards d'objets connectés |
| **Usages cibles** | démonstration de théorèmes | calcul militaire | recherche, grands fichiers admin. | TIC : bureautique et communication | numérique embarqué, pervasif |
| **Principal développement** | théorie | ingénierie | machines | logiciels (dont IHM) | contenus (données) |
| ***Leader*** | universités | armée états-unienne | IBM | Microsoft | Alphabet (Google) |

Évoquons donc successivement trois idées fondamentales liées aux machines (section 3), trois liées aux algorithmes et programmes (section 4) et trois liées aux données (section 5).

# 3 Les machines autonomes et leurs réseaux

## *3.1 Autonomie, automatique et automatisation*

**Généralité** On caractérise souvent l'Homme comme l'espèce qui utilise extensivement des outils. Les anthropologues, notamment André Leroy-Gourhan, ont montré combien l'Homme était lui-même modelé par ses outils, concevant généralement ceux-ci comme des prolongements de la main, d'abord au sens propre, puis au figuré. Avec le numérique, une nouvelle classe de machines se détache de la main pour être *autonome*, c'est-à-dire *fonctionner de façon réactive, mais non contrôlée* (parfois supervisée, cependant). Ce concept fondamental de fonctionnement autonome s'applique à de nombreux registres techniques : certains pièges de chasse préhistoriques, automates à eau de la civilisation arabe, automates à mécanisme d'horlogerie du 18$^{e}$ s. européen. Son application fut aussi non technique, ainsi Jacques de Vaucanson utilisa-t-il, au 18$^{e}$ s., des automates mécaniques pour faire progresser l'anatomie. En philosophie, l'influence de l'automatique fut considérable : « *À la vision médiévale d'un ordre hiérarchique d'êtres vivants créés et gouvernés par Dieu, la pensée du 17$^{e}$ s. opposait celle d'un ‹monde machine›, dénué de fin et de volonté, dont toutes les composantes, animaux et humains compris, étaient dirigées par les lois de la physique* » (Heudin, 2008).

Cette idée fondamentale n'est pas évoquée directement par les programmes (MENESR, 2015b). Elle mériterait pourtant notre attention, tant les représentations premières des élèves peuvent être erronées. Ainsi le fonctionnement autonome peut être confondu avec la vie (dont il partage quelques attributs), d'une façon qui peut rappeler les mots de Voltaire dans son cinquième discours en vers sur l'Homme : « *Le hardi Vaucanson, rival de Prométhée, | Semblait, de la nature imitant les ressorts, | Prendre le feu des cieux pour animer les corps* ». Il convient, de même, de bien différencier l'autonomie des notions de volonté propre ou d'intelligence.

**Influence et extension horizontale** Par définition, le fonctionnement automatique couvre toute l'informatique, puisque celle-ci se définit comme la science du traitement automatique de l'information. Cela fait partie de l'esprit même de la discipline, technique comme théorique, que de s'efforcer, dans une conception, de faire le moins possible appel à un contrôle

extérieur. Cette culture n'est pas sans alimenter d'ailleurs les peurs sociales du remplacement de l'Homme par « la machine » du fait de l'automatisation, qui prennent la suite des mêmes peurs de remplacement du fait de la mécanisation. Les œuvres de fiction d'anticipation abondent en réflexions sur ce thème. Dans le champ prospectif, le débat est actuellement intense de savoir si nos sociétés vont faire le choix de l'automatisation ou de l'instrumentation (*to automate or to informate*). Les avancées importantes actuelles de l'Intelligence artificielle, réelles ou publicitaires, posent cette question avec toujours plus d'acuité.

**Extension** L'idée fondamentale du fonctionnement autonome recouvre, en informatique, de nombreuses notions. Au sens strict, il peut s'agir d'un automate informatique, qui peut effectuer un traitement lui-même ou le faire effectuer par différents organes, comme dans une machine de Turing ou au cœur d'un processeur conçu selon l'architecture de von Neumann. À un haut niveau conceptuel, on peut aller jusqu'aux réflexions sur la nature même de ce qui constitue un traitement, un calcul, réflexions importantes en logique et informatique fondamentale depuis environ un siècle, dont les résultats les plus emblématiques étayent la thèse de Church et la correspondance de Curry-Howard (*cf.* ci-après, § 5.3).

À un niveau élémentaire, il nous semble important, au minimum, de faire comprendre les principes d'organisation des machines automatiques les plus courantes ainsi que leur dépendance à un programme de commande, afin d'écarter les interprétations magiques ou vitalistes et les lectures en termes de volonté ou d'intelligence. Les programmes de 2015 le prévoient explicitement pour le cycle 4 (C4[5]), mais ceci est accessible et pertinent dès la confrontation à des robots pédagogiques (C2/C3). Il est intéressant, à ce propos, d'observer la disjonction entre l'imaginaire, dans lequel les robots sont souvent anthropomorphes, images en miroir de l'Homme, de l'ouvrier ou de l'esclave, parfois de l'animal (*Electric dog*, 1912 ; *RUR*, 1922 ; *Metropolis*, 1927 ; etc.), et les réalisations techniques effectives. *Unimate* (1961), le premier robot industriel est seulement un bras : le corps se limite à l'instanciation de sa fonction utile (pensons de même aux automates téléphoniques, de paiement, etc.).

5 Nous abrégeons ici « C2 » le cycle des apprentissages fondamentaux (cycle 2 français : CP, CE1, CE2), « C3 » le cycle de consolidation (cycle 3 : CM1, CM2, 6e) et « C4 » le cycle des approfondissements (cycle 4 : 5e, 4e, 3e).

### *3.2 Cybernétique et interface : les flux d'information*

**Généralité et influence** On conçoit souvent le structuralisme comme un mouvement intellectuel des sciences humaines et sociales. Il a, en réalité, été également fécond en sciences avec, notamment, les mathématiques de Bourbaki et la cybernétique de Norbert Wiener, qui aura eu une influence sur plusieurs sciences, dont l'informatique, les sciences de l'information et de la communication et les sciences cognitives. L'idée fondamentale de la cybernétique est que le *comportement de communication*, c'est-à-dire son comportement en termes de *flux d'information* (C4), est plus utile pour comprendre un système que sa nature physique, qu'il s'agisse de sociétés, d'êtres vivants ou de machines. Cette théorie a permis de repenser complètement les relations de commande et de contrôle (C4), en particulier dans le cas des robots (C2/C3).

**Extension** Dans le domaine informatique, l'idée du comportement de communication se traduit principalement par la notion centrale d'interface : entrées/sorties d'un traitement, interfaces de programmation et, surtout, interfaces Homme-machine (IHM, C4). Présente dès la première industrie, la question des IHM, a pris une ampleur considérable depuis que les artefacts appelés commercialement « ordinateurs » ne constituent plus qu'une petite fraction des objets numériques. La plupart de ceux-ci sont embarqués et connectés ; leurs interfaces, d'une part avec l'Homme, d'autre part avec le réseau, sont donc primordiales pour leur fonction. L'ordinateur, *lato sensu*, est une machine très générique de contrôle, il a donc pris, avec les années, une place considérable dans le contrôle de très nombreux dispositifs, y compris non numériques, favorisant, de ce fait, un mouvement d'intégration des dispositifs de commande. Les questions sociétales deviennent prégnantes : faut-il penser les dispositifs portés comme des prothèses, des prolongements ou des augmentations de l'Homme (au sens de Douglas Engelbart) ? Quelle trajectoire technologique choisissons-nous pour notre société : celui d'interfaces « faciles » qui prennent de nombreuses décisions pour l'utilisateur sans l'en informer, ou celui d'interfaces riches qui demandent à l'Homme des efforts et le font coévoluer avec elles ?

### *3.3 Les systèmes distribués intégrateurs*

**Généralité** Norbert Wiener prédit très tôt que les ordinateurs allaient devenir des machines à communiquer et qu'ils joueraient un rôle important dans la société. Depuis cette année (2017), on estime qu'il y a plus d'objets connectés à Internet que d'être humains sur Terre. Ceux-ci confèrent désormais à de nombreux aspects de notre vie quotidienne une épaisseur numérique. Dans certains domaines, en particulier la communication, la médiation numérique est même devenue largement dominante : qui n'a pas constaté combien les adolescents pouvaient échanger par minimessages plutôt qu'oralement, y compris à quelques mètres de distance ? Le phénomène est également très présent dans certaines professions pourtant sédentaires. Certains analysent l'évolution de notre société en « société de communication ». Cette omniprésence de l'intercommunication soulève de nombreuses questions ; parmi celles-ci, retenons ici la notion fondamentale de *système distribué : un ensemble composé de nombreux agents*, pas nécessairement localisés au même endroit, *qui bénéficie d'un fonctionnement intégré*, c'est-à-dire que l'on peut percevoir comme celui d'un agent indivis. De nombreux systèmes non informatiques, tels que des entreprises, ou plus généralement des personnes morales, ont les caractéristiques de systèmes distribués. Nous nous intéresserons ici surtout aux agents informatiques en réseau, tels que les principaux outils en ligne (C3/C4) ou les objets connectés (C4).

**Extension** Dès le début les ordinateurs ont été conçus comme un concours de plusieurs éléments : l'architecture de von Neumann fait ainsi collaborer une unité de contrôle, une unité de mémoire et une unité arithmétique et logique. Les grands calculateurs qui ont précédé les ordinateurs, y compris la machine de Babbage, utilisaient aussi des transports de données entre sous-parties. En 1940, un congrès de l'American mathematical society à Hanover (NH, USA) connecta même plusieurs terminaux à un calculateur arithmétique localisé à New York (NY, USA). Dès le début, l'ordinateur fut donc un assemblage d'éléments centraux et d'éléments périphériques qui communiquaient entre eux. Au cours de la période de la loi de Moore, le nombre de processeurs centraux augmenta : le processeur principal se vit épaulé d'un coprocesseur, qui intégra ensuite la même puce, de processeurs de sons et de traitement de signaux analogiques, de processeurs spécialisés dans l'affichage vidéo, dans la gestion de mémoires de masse, etc.. Puis, les

machines deviennent parallèles ; le nombre de processeurs « principaux » augmenta, « le » microprocesseur intégrant plusieurs files de calcul, puis plusieurs cœurs. La période actuelle voit, elle, l'ordinateur se développer à travers le réseau, réseau local, puis Internet : d'abord certains périphériques, puis certains services de stockage puis de traitement. Aujourd'hui, à l'heure du « cloud », il n'est pas rare de voir certains services d'application fournis par une architecture N-tiers incluants parfois des dizaines de machines serveuses. Beaucoup de ces services (SaaS, PaaS, IaaS) sont conçus pour que l'infrastructure soit invisible à l'utilisateur, permettant ainsi une grande souplesse de mise en œuvre et, en particulier, l'extensibilité (*scalability*). Le principe de fonctionnement des systèmes distribués nécessite une forme de transparence au réseau ; pour le web, par exemple, ceci s'appuie sur un système d'adressage universel (*uniform*).

**Influence** Il nous semble essentiel de ne pas se limiter, comme les programmes actuels, au web et outils associés (C2), aux systèmes d'information distribués (C3) ou à la compréhension des arborescences (C4). Il nous semble important de comprendre qu'un système distribué est constitué de plusieurs agents logiciels qui interagissent, qu'il occasionne donc de nombreuses communications entre machines : des centaines d'échanges de données entre plusieurs dizaines de machines pour une simple consultation du web. En effet, les systèmes distribués ont vocation à prendre une ampleur dépassant de beaucoup le cas du web dans le cadre de l'*informatique pervasive*. Ce paradigme a été conçu par Mark Weiser, d'abord comme *informatique ubiquitaire* (1991), imaginant de nouveaux types d'appareils omniprésents mais invisibles, « disparaissant dans l'arrière-plan », s'intégrant de façon « transparente » à notre environnement. L'ordinateur serait alors remplacé par une multitude d'objets interconnectés. Cette informatique ambiante, ubiquitaire, qui est un tissu de systèmes distribués, est en train d'advenir, avec, comme interface principale le « téléphone mobile » individuel. Comme tout système distribué, comme le web avant elle, elle est, par nature, par conception, un *système intégrateur*, c'est-à-dire qui a vocation à incorporer de nombreuses fonctions auparavant assurées par d'autres dispositifs. C'est ce qui a fait la faveur du web, intégrant l'accès à de nombreux contenus ainsi qu'à quantité d'applications. Pour beaucoup, les systèmes qui sont ainsi intégrés s'étiolent, voire disparaissent, dans leur version non intégrée. Dans le cas du web, ceci a profondément transformé notre rapport à l'information d'actualité et au savoir. Il faut s'attendre à ce

que l'*Internet des objets* renouvelle aussi profondément le rapport à notre environnement, ou en tout cas à la part de celui-ci qui sera intégrée ou médiatisée par l'informatique pervasive.

## 4 De la conception des algorithmes à la rédaction des programmes

### *4.1 L'algorithme, procédure systématique, abstraite et finalisée*

**Influence** Les inventions les plus fondamentales sont invisibles, seule la confrontation à leur absence nous les rend présentes à l'esprit. Là est la difficulté de conception des enseignements les plus fondamentaux : produire une évolution dans l'esprit des élèves sur une notion que l'enseignant ne perçoit pas sans effort conscient. Difficulté supplémentaire pour nombre d'outils numériques : il s'agit de technologies cognitives, comme l'écriture ou l'imprimerie avant elles, et donc complètement intriquées dans la plupart des activités humaines. L'objet intellectuel algorithme en est certainement le plus parfait exemple, tant il est présent, silencieusement, dans nombre d'activités. Son histoire remonte au moins à l'Antiquité grecque. Sciences et philosophies y étaient alors fondées sur le discours, le logos (*λόγος*, toujours très présent dans notre héritage, comme l'atteste le nom en « -logie » de nombre de disciplines scientifiques ou prétendant l'être) ainsi que sur l'enquête descriptive, l'historia (*ἱστορία*, histoire politique, histoire naturelle, etc.) ; pour autant cette culture conçut également des procédés systématiques permettant de résoudre certains problèmes, tels que l'algorithme d'Euclide pour la détermination du PGCD. Les procédés de ce type se développèrent surtout, pour l'aire occidentale, dans le cadre de la science arabe médiévale. Le mot « algorithme » dérive d'ailleurs du nom d'Al-Khawârizmî, savant perse du XVIII$^{e}$–IX$^{e}$ siècle, qui rédigea plusieurs traités, dont des recueils de procédés de calcul. À cette époque, on est encore dans le domaine des outils intellectuels ; la confrontation systématique au monde, l'expérimentation, la sortie de la Caverne, ne se fera que plus tard, sous l'impulsion des Galilée, Newton et Bacon. L'*algorithme*, que nous pouvons définir comme une *procédure formalisée et systématique résolvant un problème posé dans un cadre formel*, outille l'Homme bien

avant le premier ordinateur. Il est présent aussi bien en mathématiques, en chimie ou en ingénierie, par exemple.

**Généralité** Pour ce qui est de l'adéquation aux niveaux élémentaires, adoptons une définition scolaire plus simple que la définition savante : *procédure systématique, abstraite et finalisée* (c. à d. visant un objectif donné). Sous cette formulation, c'est déjà une compétence à développer au C2 : l'élève doit apprendre à ne pas présupposer que l'autre sait déjà. Il doit ainsi être capable d'indiquer son chemin à un passant ; il doit pouvoir décrire une scène ou un dessin à un camarade de façon à ce que celui-ci puisse la reproduire ; etc. Cette compétence peut s'instancier dans de nombreuses situations de jeu de rôle pédagogique ou de la vie courante.

Moyennant notre définition adaptée, la recette de cuisine n'est plus seulement une métaphore d'algorithme, mais un algorithme à proprement parler. Sa mise en œuvre permet de faire le lien entre la procédure écrite, abstraite, et le processus effectif qui l'actualise. Cet exemple permet d'ailleurs de voir que l'algorithme de l'école primaire n'est pas nécessairement linéaire ou circulaire, à l'image des orgues de barbarie ou des métiers Jacquard, il peut également introduire des conditionnels, des options, des paramètres, des itérations. Ces aspects pourront être ensuite formalisés au C4.

**Extension** Pour compléter, notons avec Baron & Bruillard (2001) combien l'algorithmique, la science des algorithmes, est essentielle dans la construction de la science informatique.

### *4.2 Le programme, implémentation concrète d'un algorithme*

L'algorithme est la compréhension d'un processus, tandis que le codage sous forme d'un programme est la rédaction d'un texte qui permet de lever certaines ambiguïtés de la conception (toutes, à l'idéal) et de faire exécuter les commandes par un agent (humain ou artefact). Cette rédaction comporte de nombreuses dimensions ; quelle(s) notion(s) de programmation retenir comme fondamentale(s) ?

**Influence** Le premier concept, qui nous semble le plus fondamental après celui de l'algorithme, est celui de *programme enregistré*. Pour la plupart des historiens de l'informatique, c'est souvent la présence d'un programme enregistré qui sert à poser l'acte de naissance de l'ordinateur – tous ne

s'accordent pas, d'ailleurs, sur les autres critères[6]. On peut faire remonter cette idée, au moins, aux automaticiens Jaquet-Droz du 18e s. (*cf.* Heudin, 2008). Ceux-ci, parmi les plus remarquables fabricants d'automates à mécanismes d'horlogerie de leur époque, créèrent un automate écrivain permettant de produire « n'importe quel texte comportant un maximum de 40 lettres ou signes », de même qu'un dessinateur et une musicienne. Ces trois automates étaient programmables au sens où les mouvements d'horlogerie étaient codés par des ergots sur un cylindre amovible. Cela peut sembler anecdotique rétrospectivement, mais le fait que ce cylindre soit amovible constitue une rupture majeure. Cela change toute la perspective : toutes les lettres, y compris celles qui ne sont pas dans le premier texte, doivent être codées. La programmation par cylindre et ergots, qui auparavant activait directement des actionneurs, à la façon d'un langage-machine, devenait un langage de plus haut niveau, un véritable langage de communication. Le célèbre métier à tisser Jacquard, au début du 19e s., utilisa la même idée pour enregistrer un programme de tissage sur une bande de carton. Ce métier inspira plus tard les machines à calculer à mécanisme d'horlogerie de Charles Babbage, dont l'analyse par Ada Lovelace conduisit à la première formalisation du concept d'algorithme et au premier programme de calcul jamais écrit.

**Extension** Du point de vue pédagogique, l'idée fondamentale de programme enregistré permet de développer plusieurs idées, à partir du cycle 3. Tout d'abord, cela permet de bien distinguer entre ce qu'est un algorithme et ses *implémentations* sous la forme de programmes. Les programmes eux-mêmes, qui sont des textes, sont, comme tous les textes, dans tous les langages, encore des virtualités qui attendent d'être actualisées sous la forme d'une interprétation, d'une exécution. Ceci permet d'ancrer les notions d'*instruction* et de *programmation impérative*, très tôt, montrant par l'exemple que « la mécanisation est possible du fait de […] la précision de la notation » (Wing, 2008). À un niveau plus avancé, C4, cela peut être l'occasion de faire toucher du doigt les dimensions documentaire et effective des langages informatiques ; citons (Lévy, 1992) :

> Comme le dit Daniel Bougnoux : « Le poète ne se contente pas de donner à entendre ou à voir, il fait exister. Le poème ne signifie pas, il est. » […] De là une confusion

6 Certains voient ainsi le *Zuse 3* comme premier ordinateur, tandis que d'autres attribuent ce mérite aux premiers ordinateurs utilisant l'architecture de von Neumann.

> féconde entre le signe et la chose : « En poésie, les mots fonctionnent comme choses parmi les choses plutôt que comme signes. » | On voit ce qui rapproche le poème du logiciel. Au fur et à mesure que l'on s'éloigne des interfaces et langages tournés vers l'utilisateur humain et qu'on s'approche des rouages de la machine, le logiciel abandonne le caractère fantomatique et subordonné des signes pour acquérir la persistance et la densité des choses. | Plus on descend vers le langage-machine et plus la fonction référentielle du langage semble s'effacer […] | Telle est l'ambivalence fondamentale du logiciel : il peut être tour à tour symbole et agent, effectif et descriptif.

**Généralité** Autour du concept fondamental de programme enregistré, nous avons choisi de décliner la séparation entre code et mécanisme, la notion de langage de programmation impératif, la démarche d'implémentation et une première approche de la dimension documentaire[7]. Ce choix permet de développer une réflexion plus générale sur le codage, des langues naturelles comme des langages informatiques, tant pour ses dimensions lexicales, syntaxiques que, dans une certaine mesure, conatives, notions essentielles des cycles 2 à 4, indépendamment de l'informatique. La dimension sémantique semble, en revanche, difficile à aborder dans le cadre de l'enseignement obligatoire, à ce jour.

Cette dimension linguistique est très ancrée dans l'esprit humain, ou dans la culture contemporaine ; c'est peut-être une des sources d'un phénomène qui ne laisse pas d'émerveiller tout enseignant d'informatique : les cris de joie très spontanés des élèves quand s'exécute, et fonctionne, leur premier programme. Ce phénomène est tellement courant qu'il a conduit à une forme de rite d'initiation : écrire un premier programme affichant « Bonjour, tout le monde ! » (« Hello, world », en anglais).

### *4.3 La modularité, élégance conceptuelle et pragmatique des programmes*

**Généralité et influence** Terminons ce second volet par une troisième notion fondamentale, la *modularité*, qui n'apparaît dans les programmes de 2015 que sous la forme de la décomposition d'un problème en sous-problèmes (C4). Elle est pourtant, à nos yeux, la plus importante, car elle apporte à

7 D'autres formes de programmation existent, au-delà de la programmation impérative ; nous les écartons ici (cependant, *cf.* § 5.3), dans la mesure où elles font appel à des conceptions, mathématiques notamment, qui ne sont pas en place au niveau auquel nous nous plaçons dans cet article.

la fois une dimension esthétique (certes peu accessible aux profanes) et un outillage intellectuel très général, dont l'intérêt dépasse très largement l'informatique : philosophie, ingénierie, gestion de projets, mathématiques, etc.

**Extension** Historiquement, le premier grand pas dans la théorisation de la modularité, en 1968, est l'article « Go To Statement Considered Harmful », où Edsger Dijkstra défend la nécessité de la programmation structurée. Le principe de modularité comportera par la suite de nombreux autres avatars, notamment l'encapsulation (*separation of concerns*) et la programmation orientée objet. C'est encore la modularité qui fait que dans une page web on s'efforce de sépare le contenu (HTML), sa mise en forme (CSS) et l'interaction (ECMAScript). À un niveau plus élémentaire, c'est le fait d'utiliser systématiquement des feuilles de style dans un traitement de texte. La *modularité* c'est une *démarche d'abstraction de situations concrètes afin d'en percevoir la généricité et*, celle-ci identifiée, *d'affecter des traitements spécifiques à des procédures spécifiques*.

On le voit, l'enseignement de démarches relevant de la modularité a tout à fait sa place dans l'enseignement obligatoire. Il est même possible de l'aborder dès le C3 à l'aide de logiciels comme *Scratch*, ou même *ScratchJr*, qui permettent de travailler avec plusieurs agents simultanés (les lutins), attribuant à chacun un programme spécifique réagissant à certains sémaphores ou messages. Cela peut également être un moyen de montrer l'importance de la dimension documentaire que nous évoquions précédemment. Comme pour les langues naturelles, plus la maîtrise de la langue est limitée, ou plus le sujet est complexe, et plus il convient de bien structurer un texte, de bien l'expliciter.

## 5 L'information et son codage

La troisième grande époque du développement de l'informatique, en cours, est celle des contenus, des données, de l'information. L'informatique est la science du « traitement rationnel de l'information, notamment au moyen de l'ordinateur » ; le bit est l'« unité fondamentale de quantité d'information » ; notre époque serait celle de la « société de l'information » ; mais qu'est-ce que l'*information* ? En science informatique ou en sciences de

l'information et de la communication, le sens est clair, défini par de nombreux théoriciens à la suite de Claude Shannon, depuis le milieu du 20e s. En revanche, la scolarité obligatoire n'introduit aucune de ces théories, nous laissant, au mieux, avec une notion protéiforme dont les différents aspects ne sont pas conceptuellement reliés et, au pire, avec un mot très polysémique. Qui plus est, l'« information », au sens de l'informatique, serait certainement occultée par l'« information » au sens de l'éducation aux médias et à l'information (EMI), qui, elle, occupe une place importante dans les programmes. Nous proposons de ne pas conserver ce mot pour l'enseignement obligatoire, et de lui préférer la notion plus accessible de « donnée » : on parlera ainsi de « traitement de données » et de « quantité de données ». Sur le fond, il nous semble essentiel d'aborder les trois idées fondamentales suivantes, du codage, de la structuration et du typage.

### *5.1 Principe de codage : « un bit est un bit »*

**Généralité** Notre première idée fondamentale de ce domaine est ce que nous appellerons le *principe de codage : une donnée est un signe plus un code* (au sens d'encodage). Son principe, l'arbitraire du signe, remonte aux origines de l'écriture alphabétique elle-même : le signe ou l'assemblage de signes (le signifiant) qui désigne un sens (le signifié) n'a pas de nécessité intrinsèque ; il relève, pour l'essentiel de choix et de déterminismes historiques. Même si ce principe n'a été étudié théoriquement qu'à la suite des travaux de Ferdinand de Saussure (19e s.), le phénomène est connu depuis l'Antiquité, celui-ci permettant de composer des codes secrets, dits de substitution, un signe remplaçant une lettre, un groupe de lettre ou un mot. Francis Bacon (16e–17e s.) inventa, sur cette base, le premier code binaire en représentant chaque lettre par une suite de ce que nous appelons aujourd'hui des bits. Le grand pas suivant est réalisé en 1936 par Alan Turing, qui conçoit, pour résoudre une question mathématique, un modèle de calcul, que nous appelons aujourd'hui machine de Turing, en codant tout calcul par une suite de symboles tirés d'un alphabet fini.

**Influence** Ce modèle se traduira en artefacts matériels grâce à l'architecture de von Neumann, dont relèvent encore tous les ordinateurs aujourd'hui (certes sous une forme très évoluée). L'idée de Turing, reprise par von Neumann, et développée ensuite, est qu'une mémoire informatique stocke

toutes natures de données dans les mêmes bits. Une même série de cases mémoires peut coder une partie de programme, une partie de texte, une partie de vidéo ou toute autre donnée. Conséquence conceptuelle, les bits ne portent pas d'information en eux-mêmes, il est nécessaire de spécifier un codage pour que les données prennent sens. Conséquence pratique : une même mémoire informatique, un même fichier, un même canal de transmission, un même logiciel peuvent stocker, contenir, véhiculer et traiter de façon indifférente tous types de données. Dès le début, le multimédia, les documents interactifs et la convergence numérique étaient déjà en germe. Nicholas Negroponte résume cela par sa célèbre formule « un bit est un bit ».

**Extension** L'extension horizontale ne faisant guère de doute, évoquons seulement l'extension verticale. Le programme de 2015 (MENESR, 2015b) couvre bien les éléments pratiques du codage, évoquant la notion de signal (C3/C4), la représentation binaire (C3) et le bit (C4), les mémoires informatiques (C2) et leurs ordres de grandeur de capacité (C3/C4), la quantité de données (« d'information », C3/C4), les documents multimédias (C3). Les aspects conceptuels eux, relèvent plutôt de l'enseignement de français : arbitraire du signe linguistique (C2), chiffrement par substitution (C4). Il nous semble important de pouvoir faire le lien entre les deux. Le principe de codage, qui est fondamentalement un principe logique, dit que *toute information enregistrée numériquement est symbolique*, qu'il s'agisse de micro-symboles, comme des pixels, ou de macro-symboles, comme des mots. Ceci permet également de mettre en perspective les technologies cognitives, des tablettes argiles sumériennes aux fichiers numériques actuels. C'est pour cette raison que nous préférons dire « numérique », même si les signes ne sont pas seulement des nombres, plutôt qu'« informatique » ou, pire, « *computer science* ».

### *5.2 « Algorithmes + structures de données = programmes »*

**Extension** Deuxième avancée majeure : la compréhension de l'importance de la représentation des données dans la conception des logiciels. Dans ce domaine, le célèbre ouvrage de Niklaus Wirth, *Algorithms + Data Structures = Programs*, publié en 1976, fait figure de *credo*. Il ne s'agit pas là seulement d'une question pratique de codage des contenus

(C2) ou de fichiers (C2, C4), dans le prolongement du principe de codage (§ 5.1), mais d'une seconde idée fondamentale : *toute donnée nécessite un référentiel* (techniquement un *type*). Il s'agit d'un cadre conceptuel qui permet de lui donner une signification, opératoire notamment. Prenons un exemple, le plus courant : l'immense majorité des données (sons, images fixes, vidéos, etc.) sont acquises par numérisation, en combinant un *échantillonnage* et une *quantification* qui produisent des données brutes, qui seront ensuite structurées de façon pertinente en données élaborées, lesquelles, enfin, seront codées à proprement parler pour être stockées. Échantillonner demande une représentation de l'espace et/ou du temps et la quantification une représentation de la grandeur à enregistrer, notions qui peuvent être reliées au champ du corps et de l'espace (position relative, latéralisation, etc.), avec les mathématiques (repère cartésien), avec les sciences de la nature (données expérimentales). De même, l'élaboration des données sonores ou visuelles, par exemple, s'appuie sur une représentation de l'ouïe ou la vue humaines. Cette réflexion, omniprésente en informatique fondamentale, a créé, dans le domaine appliqué, nombre de professions : spécialistes de bases de données, de systèmes experts, d'ontologies, *data scientists*, architectes de l'information, etc., toutes spécialités dont l'activité essentielle consiste à trouver la représentation la plus pertinente d'un corpus de données. La place considérable prise par les données et leur exploitation dans la période actuelle fait de cette question un enjeu économique considérable.

**Influence** Pour mesurer l'influence de cette idée fondamentale, évoquons seulement le cas des sciences humaines et sociales, dont certaines approches ont été profondément renouvelées depuis les années 1990 par l'emploi de bases de données et de traitement de corpus étendus, construisant peu à peu le champ maintenant désigné par le terme « humanités numériques ». Il ne s'agit pas que d'outils technologiques : il s'agit aussi d'outils conceptuels fondés précisément sur une réflexion sur la structuration des données. Laissons la parole à Peter Denley, cofondateur et premier Secrétaire général de l'Association for history and computing, qui explique (1994, p. 34) que comprendre le principe d'organisation d'une base de données relationnelle serait, pour tous les historiens, « *a considerable service to their understanding of structured systems of any kind* ».

**Généralité** Un tel niveau informatique n'est pas aujourd'hui accessible à l'enseignement obligatoire. En revanche, ce principe est loin de se limiter à l'informatique. Citons, en mathématiques, le repère cartésien et l'organisation des données (C3), ou, plus généralement, la question du repérage du corps (latéralisation, p.ex.) ou dans l'espace (C2). En physique, la question de la relativité galiléenne peut être abordée dès le C4. En arts graphiques, la pratique du *pixel art* ou de la mosaïque à la façon de l'artiste français Invader, outre leur intérêt artistique, permettent de faire le lien entre histoire de l'art et pratiques artistiques explicitement numériques. Elle permet, de plus, d'aborder conceptuellement la différence entre un document nativement numérique et un document issu d'une numérisation.

### *5.3 Types de données : démontrer, c'est programmer*

**Extension** La troisième idée fondamentale du domaine des données est, selon nous, la *correspondance de Curry-Howard*, ainsi que ses développements dans les années 1980 et suivantes et leurs applications dans le domaine de la preuve de programmes en génie logiciel. Cette correspondance fait le lien entre démonstrations logiques et exécution de programmes informatiques, *via* les types de données, et permet de les considérer, d'une certaine manière, comme deux faces d'une même pièce, deux façons de présenter le même phénomène de calcul. Cette correspondance permet aussi de fonder théoriquement la dimension documentaire d'un programme, le fait qu'un programme ne doive pas être considéré seulement comme une suite d'instructions, mais aussi comme un texte qui a des lecteurs humains. Il doit donc avoir un sens. Un programme correctement rédigé doit comporter, outre les instructions, un commentaire décrivant, voire expliquant, les algorithmes et structures de données utilisés ainsi qu'une spécification, un typage, pour chaque traitement de ses entrées, sorties et variables, et, enfin, un guide ou manuel d'usage du logiciel. L'exemple le plus abouti, en la matière, est probablement le programme *Tεχ* de Donald Knuth, qui intègre à la fois des instructions permettant de produire un logiciel exécutable et la documentation de ce même logiciel. Qui plus est, cette documentation peut être mise en forme par le logiciel *Tεχ* lui-même. Une tradition des années 1990, de façon peut-être un peu désobligeante, prétendait que ce programme serait le seul dénué de bugs.

**Généralité et influence** À un niveau plus élémentaire encore, l'idée fondamentale de la correspondance de Curry-Howard nous enseigne que tout calcul devrait se concevoir comme typé. Ne voyons pas là seulement une norme opératoire ; ce principe a également un intérêt didactique et heuristique. Dans le domaine de la programmation, il est aisé de le mettre en œuvre de façon élémentaire. Ainsi le langage visuel *Scratch* code-t-il les (nombres) par des ovales, les [chaînes de caractères] par des rectangles et les <booléens> par des hexagones. En physique, ce même principe correspond à l'analyse dimensionnelle (équations aux dimensions), très féconde historiquement. À l'école élémentaire, sans même parler du secondaire (voire de certains étudiants du supérieur), nous sommes toujours frappés de constater combien d'élèves s'évertuent à combiner des grandeurs incommensurables telles que cargaison et âge du capitaine : ils se lancent dans des calculs sans réflexion sur leur sens, et exécutent le calcul avant d'avoir pris la peine de l'écrire. En mathématiques comme en informatique, ce principe fondamental montre qu'il est important de faire précéder l'exécution par une spécification, et de bien distinguer les deux. Faut-il enseigner une arithmétique non typée au C2 ? Nous ne le croyons pas ; les enfants découvriront la beauté des corps quand ils seront plus grands ! À cet âge, il convient plutôt de poser les bases d'un raisonnement structuré. Nous sommes ici au croisement de la logique, de la physique, de la philosophie et de l'informatique. Ainsi, au C2, il conviendrait d'expliquer que 8 lignes-de-chocolat × 4 carrés-par-ligne = 32 carrés-de-chocolat ; que 60 billes ÷ 4 personnes = 15 billes-par-personne ; que l'on ne peut pas additionner 3 pommes et 4 bananes, mais que 3 fruits + 4 fruits est possible ; etc. Plus tard, au C3 ou C4, on pourra aller plus loin et considérer que 3 pommes + 4 bananes donnent (mais pas « égalent ») 7 fruits.

## 6 Conclusion

À titre de conclusion, listons ici les principales idées relevées ainsi que quelques repères suggérés pour chacune (pour une suggestion de développement curriculaire de ces idées, *cf.* l'annexe en fin d'article) :

- Fonctionnement autonome : définition : « fonctionnement réactif mais non contrôlé » ; différence avec la vie, la volonté propre et l'intelligence ; automatique ; peur du remplacement ; débat entre automatisation et instrumentation ; programme de commande ; robot.
- Comportement de communication : flux d'information pour comprendre un système (société, être vivant, machine) ; interface ; interface Homme-machine ; débat prothèse/prolongement/augmentation.
- Système distribué : définition : « ensemble composé de nombreux agents qui bénéficie d'un fonctionnement intégré » ; web ; système intégrateur ; transparence au réseau ; interactions entre logiciels, entre machines ; informatique ubiquitaire, pervasive.
- Algorithme : définition : « procédure formalisée et systématique résolvant un problème posé dans un cadre formel » ; définition simplifiée : « procédure systématique, abstraite et finalisée » ; apprendre à ne pas présupposer qu'un interlocuteur sait déjà ; limites des algorithmes linéaires (ou circulaires).
- Programme enregistré : distinction entre langage de programmation et langage-machine ; distinction entre algorithme et implémentation ; exécution d'un programme ; instruction ; programmation impérative ; importance de la précision de la notation ; codage et langages informatiques.
- Modularité : définition : « démarche d'abstraction de situations concrètes afin d'en percevoir la généricité et d'affecter des traitements spécifiques à des procédures spécifiques » ; décomposition d'un problème en sous-problèmes ; utiliser systématiquement des feuilles de style dans un traitement de texte.
- Principe de codage : une donnée est un signe plus un encodage ; les bits ne portent pas d'information en eux-mêmes : « un bit est un bit » ; une mémoire, un fichier, un canal de transmission, un logiciel peuvent stocker, contenir, véhiculer et traiter de façon indifférente tous types de données ; toute information enregistrée numériquement est symbolique.
- « Algorithmes + structures de données = programmes » : toute donnée nécessite un référentiel conceptuel ; échantillonnage ; quantification ; différence entre document numérisé et document nativement numérique.
- Correspondance preuve-programme : dimension documentaire d'un programme ; typer tout calcul ; analyse dimensionnelle.

## Références

Baron, G., & Bruillard, E. (2001). Une didactique de l'informatique ?, *Revue fr. de pédagogie*, 163–172.

Bruner, J. S. (1960), *The Process of Education*, Harvard University Press.

Delmas-Rigoutsos, Y. (2014), *Histoire de l'informatique, d'Internet et du web*, notes de cours, Université de Poitiers, < https://delmas-rigoutsos.nom.fr/documents/YDelmas-histoire_informatique.pdf>.

Denley, P. (1994), Models, Sources and Users : Historical Database Design in the 1990s, *History and Computing* 6 (1), 33–43.

Denning, P. J. (2003), Great principles of computing, *Communications of the ACM* 46 (11), 15–20.

Dowek, G. (2011), Les quatre concepts de l'informatique. In *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques*, actes Didapro 4, 24–26 oct. 2011, Université de Patras, < https://edutice.archives-ouvertes.fr/edutice-00676169/fr/>.

Fayon, D. (2010), *Web 2.0 et au-delà*, Economica, 2e éd.

Heudin, J.-Cl. (2008), *Les créatures artificielles, des automates aux mondes virtuels*, Odile Jacob, Paris.

Lessig, L. (2000), Code is Law—On Liberty in Cyberspace, *Harvard Magazine*, jan. 2000, trad. française : < https://framablog.org/2010/05/22/code-is-law-lessig/>. Ces idées sont développées dans : *Code and Other Laws of Cyberspace*, Basic Books, 1999, et *Code : Version 2.0*, < http://codev2.cc/>, 2006.

Lévy, P. (1992), *De la programmation considérée comme un des beaux-arts*, éd. La découverte.

Ministère de l'éducation nationale de l'ens. sup. et de la recherche (2015a), Socle commun de connaissances, de compétences et de culture, *Bull. off. éducation nationale*, décret n° 2015–372 du 31/03/2015, France.

Ministère de l'éducation nationale de l'ens. sup. et de la recherche (2015b), Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4), *Bull. off. éducation nationale*, bulletin officiel spécial n°10 du 19/11/2015.

Hartmann, W., Näf, M., Reichert, R. (2012), *Enseigner l'informatique*, Springer, éd. orig. 2006.
Papert, S. (1996), An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning*, 95–123.
Schwill, A. (1993), Fundamentale Ideen der Informatik, *Zentralblatt für Didaktik der Mathematik*, 20–31.
Weiser, M. (1991), The Computer for the 21st Century, *Scientific American* 265 (3), sept. 1991 ; repris en français : Les réseaux informatiques de l'an 2000, *Pour la science* 169, nov. 1991.
Wing, J. (2006), Computational thinking, *Communications of the ACM*, 33–35.
Wing, J. (2008), Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society* A 366, 3717–3725.

# A Annexe : trame de curriculum

Nous synthétisons ici les éléments curriculaires proposés ci-dessus. Ces éléments ne visent que la pensée informatique et ne concernent donc ni les usages du numérique ni l'éducation aux médias et à l'information, sauf dans les quelques cas où il y a convergence entre ces objectifs. Par ailleurs, ces éléments se réfèrent à des notions scolaires qui transposent des concepts savants, et non à ces derniers. Nous suivons le découpage français utilisé dans l'article : C2, le cycle 2 des apprentissages fondamentaux (CP, CE1, CE2, 6–9 ans) ; C3, le cycle 3 de consolidation (CM1, CM2, 6e, 9–12 ans) ; C4, le cycle 4 des approfondissements (5e, 4e, 3e, 12–15 ans).

## *A.1 Les machines autonomes et leurs réseaux*

### A.1.1 Autonomie, automatique et automatisation

*Notion centrale.* Machine autonome ou automatique : fonctionnant de façon réactive, mais non contrôlée.

*Points de repère.* C2 : Structure des machines autonomes courantes, en particulier l'ordinateur, désignation de ses parties usuelles. C2 : L'autonomie n'est pas la vie (même s'il y a des attributs communs). C2/C3 : Dès leur première utilisation, structure des robots pédagogiques ; notion de programme de commande. C2 : L'autonomie n'implique pas forcément de volonté propre ni d'intelligence.
*Question transversale possible.* C4 : Question de société : le remplacement de l'Homme par la machine, qui peut être abordée sous l'angle de l'opposition automatisation/instrumentation.

### A.1.2 Les flux d'information

*Notion centrale.* Comportement de communication, c'est-à-dire comportement en termes de flux d'information, d'une machine, d'un être vivant, d'une personne ou d'une société[8].
*Points de repère.* C2/C3 : Comportement de communication de robots pédagogiques. C4 : Notion de flux d'information. C4 : Interface personne-machine. C4 : Communications entre machines.
*Question transversale possible.* C4 : Les machines comme prothèses (réparation), prolongements (outillage) ou augmentations (potentialisation) de l'Homme.

### A.1.3 Les systèmes distribués intégrateurs

*Notion centrale.* Système distribué : ensemble composé de nombreux agents, qui bénéficie d'un fonctionnement intégré bien que les agents ne soient pas nécessairement localisés au même endroit.
*Points de repère.* C2 : Structure du web, adresse universelle. C3/C4 : Systèmes d'information distribués, structure des outils en ligne usuels, arborescence. C4 : Fonctionnement des objets connectés ; transparence au réseau.
*Question transversale possible.* C4 : Informatique ambiante/ubiquitaire/pervasive, Internet des objets.

8 La notion d'information, au sens de l'informatique ou au sens des sciences de l'information et de la communication, est à réserver pour la suite de la scolarité. Le « flux d'information » est ici essentiellement un échange de signaux, de données ou de messages.

## *A.2 Les algorithmes et les programmes*

### A.2.1 L'algorithme

*Notion centrale.* Algorithme : procédure systématique, abstraite et finalisée.
*Points de repère.* C2 : Dans une transmission d'instructions, apprendre à ne pas présupposer qu'un interlocuteur sait déjà. C2/C3 : Exemples courants d'algorithmes (calcul posé, recette de cuisine, etc.), notamment qui ne soient ni linéaires ni circulaires. C2/C3 : Des activités de programmation simple pourront aborder des exemples de conditions, de paramètres et d'itérations. Ces trois notions seront formalisées au C4. C4 : Limites des algorithmes linéaires et circulaires.
*Questions transversales possibles.* Les programmes offrent déjà de nombreux points de contact entre la notion d'algorithme et l'enseignement des procédures (C2–C3) et des mathématiques (C3–C4).

### A.2.2 Le programme

*Notion centrale.* Programme enregistré : texte composé d'une suite organisée d'instructions.
*Points de repère.* C3 : Implémentation d'un algorithme simple ; faire dire « Bonjour, tout le monde ! » ; importance de la précision de la notation. C3 : Distinction entre programme et exécution, notion de programmation impérative. C4 : Distinction entre langage de programmation et langage-machine.
*Question transversale possible.* C2–C4 : Codage des langues naturelles et des langages informatiques, dimensions lexicales, syntaxiques et, dans une certaine mesure, conatives.

### A.2.3 La modularité

*Notion centrale.* Modularité : soucis d'affecter des traitements spécifiques à des procédures spécifiques.
*Points de repère.* C3 : Décomposition d'un problème en sous-problèmes, par exemple en faisant agir plusieurs agents simultanément (p. ex. lutins de *Scratch*). C3/C4 : Structuration des programmes. C4 : Structures de contrôles simples. C4 : Démarche d'abstraction de situations concrètes afin d'en percevoir la généricité ; cas pratique : utiliser systématiquement des feuilles de style dans un traitement de texte.

*Question transversale possible.* La démarche de recherche de généricité peut être conduite dans de nombreuses situations, pas nécessairement reliées explicitement à l'informatique.

### *A.3 Les données et leur codage*

#### A.3.1 Principe du codage

*Notion centrale.* Donnée : un signe, à interpréter à l'aide d'un codage.
*Points de repère.* C2 : Les ordinateurs traitent des données, qui sont stockées dans des mémoires informatiques ; mémoires de masse usuelles. C3 : Toute information enregistrée numériquement est symbolique ; un même signe peut représenter de nombreuses données ; ex. de la représentation binaire. C3/C4 : Notion de signal. C4 : Quantité de données, le bit, l'octet, mesures usuelles. C4 : Notions générales sur le chiffrement des communications.
*Question transversale possible.* C2–C4 : Arbitraire du signe linguistique, p. ex. avec des jeux de chiffrement.

#### A.3.2 Structures de données

*Notion centrale.* Toute donnée nécessite un référentiel pour l'interpréter.
*Points de repère.* C2 : Exemples de données rapportées à un référentiel, p. ex. latéralisation de mouvements relatifs d'un robot. C3 : La numérisation, comme combinaison d'un échantillonnage et d'une quantification, ex. des pixels. C3 : Différence entre document numérisé et document nativement numérique. C3 : Exemples de référentiel de données, p. ex. mouvement d'un lutin ou d'un robot codé en coordonnées cartésiennes ou en déplacement relatif. C4 : Formats usuels de fichiers.
*Questions transversales possibles.* C2 : Repérage du corps ou dans l'espace. C3 : Repères cartésiens, organisation des données. C4 : Données expérimentales. C4 : Relativité galiléenne. C2–C4 : *pixel art* en arts plastiques.

#### A.3.3 Typage des calculs et des traitements de données

*Notion centrale.* Toute donnée a un type, qui détermine les traitements qu'elle peut subir.

*Points de repère.* C2 : Tout calcul, en mathématiques, en sciences et en informatique, doit être typé. C3 : Analyse dimensionnelle élémentaire. C4 : Tout programme doit être commenté et documenté : type des entrées, type des sorties, effets.
*Question transversale possible.* C2 : L'apprentissage du calcul est au carrefour des mathématiques, des sciences et de cette notion fondamentale du typage de données.