RESET
Recherches en sciences sociales sur Internet

11 | 2022
Codes. L'informatique comme elle s'écrit

Codes. L'informatique comme elle s'écrit

Gabriel Alcaras et Antoine Larribeau

Édition électronique
URL : https://journals.openedition.org/reset/3914
DOI : 10.4000/reset.3914
ISSN : 2264-6221

Traduction(s) :
Writing code, making software - URL : https://journals.openedition.org/reset/3944 [en]

Éditeur
Association Recherches en sciences sociales sur Internet

Référence électronique
Gabriel Alcaras et Antoine Larribeau, « Codes. L'informatique comme elle s'écrit », *RESET* [En ligne], 11 | 2022, mis en ligne le 22 avril 2022, consulté le 23 avril 2022. URL : http://journals.openedition.org/reset/3914 ; DOI : https://doi.org/10.4000/reset.3914

Codes. L'informatique comme elle s'écrit

Gabriel Alcaras et Antoine Larribeau

1 Comment se créent les logiciels que nous utilisons chaque jour ? Qui construit les infrastructures informatiques de nos sociétés contemporaines ? De quoi le numérique est-il fait ? Ces questions appellent quotidiennement des réponses tantôt idéalisantes, tantôt désenchantées. Quel que soit le ton du discours, une seule chose semble sûre : l'informatique est une machinerie efficace, qui produit des effets sur le monde social. Les débats croissants sur les algorithmes, notamment concernant leur pouvoir et leur opacité, illustrent parfaitement cette logique particulière d'interprétation du monde informatique. Même si les études consacrées à l'appropriation des techniques mettent en garde contre une vision déterministe des technologies et documentent « les possibilités d'*autonomie* et d'*émancipation* pour les individus et les groupes » (Proulx, 2015), les infrastructures numériques conservent leur aura de dispositifs fascinants, à la fois incroyablement complexes et parfaitement ordonnés.

2 Pour interroger cette illusion d'ordre et d'efficacité, une stratégie possible est d'entrer au cœur du dispositif : « de l'extérieur, on est frappé par l'agencement merveilleux des éléments, bien alignés les uns par rapport aux autres, harmonieusement solidaires ; de l'intérieur on découvre les éléments retors, les froissements, les blocages, les aspérités » (Dodier, 1995, p. 5). Mais quel « intérieur » explorer quand les infrastructures sont si vastes, si diverses ? Doit-on s'occuper des bâtiments, des circuits imprimés, des serveurs, des câbles du réseau, des bases de données, des logiciels, des plateformes ou de quantité d'autres objets — sans parler des personnes qui les créent, les maintiennent et les utilisent ? Faire un choix n'est pas évident, surtout face à la polysémie du terme numérique, dont les usages et les définitions varient grandement selon les contextes (Moatti, 2012 ; Drot-Delange et Bruillard, 2012 ; Baron, 2018).

3 L'objectif de ce numéro spécial de RESET est d'encourager les sciences sociales à emprunter une porte d'entrée encore trop peu empruntée — l'informatique. Nous entendons par ce terme à la fois la discipline scientifique et le génie industriel qui participent à la production de nos infrastructures numériques et qui ont pour

ressource le code, c'est-à-dire un texte destiné à être exécuté par un ordinateur[1]. Étudier l'informatique comme elle s'écrit constitue donc une piste particulièrement engageante pour saisir la construction des infrastructures logicielles de l'intérieur.

1. Programmer ou coder ?

4 Récemment, le terme de « programme » a été employé pour aborder les logiciels dans toute leur globalité et leur ambivalence (Méadel et Sire, 2017), du code aux usages, des interfaces (Galloway, 2012) aux API[2] (Ermoshina, 2017), des infrastructures physiques aux bases de données. Ce numéro s'inscrit dans la continuité de ces recherches et fait le pari que partir d'un objet plus précis, le code, et d'une activité plus circonscrite, son écriture, peut non seulement documenter l'informatique en train de se faire, mais aussi contribuer à une compréhension plus globale des infrastructures logicielles.

1.1. Hacker, geek, entrepreneur : au-delà des figures mythiques

5 L'activité informatique reste encore largement dans l'ombre de grandes figures mythiques. Qu'il s'agisse des hackers (Hikkamen, 2001 ; Auray, 2013 ; Lallement, 2015), des geeks (Kelty, 2008) ou encore des entrepreneurs géniaux (Turner, 2006), ces représentations émiques[3] attirent souvent l'attention au détriment des pratiques concrètes de l'informatique. Ces dernières se trouvent alors régulièrement occultées des travaux universitaires, au profit des discours — parfois conflictuels — que tiennent les codeur·euses sur leurs propres mondes. Ces discours incluent par exemple la constatation que le code serait la loi de l'espace cybernétique (Lessig, 1999) ; la revendication de la programmation comme un instrument d'émancipation politique (Auray et Ouardi, 2014) ou économique (Stevens 2012 ; Vicente 2017) ; la promesse d'une réinvention du travail à travers le jeu et l'expérimentation de l'activité informatique (Berrebi-Hoffmann et al., 2018 ; Flichy, 2017) ; la remarque enthousiaste selon laquelle « le logiciel est en train de dévorer le monde »[4] et que tous les aspects de nos sociétés contemporaines pourraient être transformés, voire améliorés, par leur mise en code.

6 Rendre compte de ces représentations constitue incontestablement une force pour la recherche sociologique et anthropologique. Grâce à ces travaux, nous avons gagné une compréhension beaucoup plus fine d'éléments essentiels de la culture technique (Coleman, 2010), politique (Coleman, 2013 ; Auray, 2007 ; Broca, 2013) et économique (Rosental, 2017 ; Vincente 2017) des milieux informatiques. Ces discours nous aident à comprendre l'intérêt pour le code, les rapports qu'entretiennent les individus avec leur activité ou encore les stratégies de distinction et de légitimation déployées par certains groupes sociaux. Mais les discours, si opérants soient-ils, ne sont pas les pratiques. Si les premiers ont été remarquablement investis par la recherche, nous ne pouvons pas en dire autant des secondes. Lorsque des enquêtes sont menées sur l'activité informatique, elles l'abordent souvent par le prisme de ces discours, que ce soit pour les vérifier ou pour les infirmer. En dépit des réponses qu'apportent ces travaux, les questions restent, dans l'ensemble, inchangées — et ce sont ces questions qu'il s'agit aujourd'hui de renouveler.

1.2. Sortir de la réduction algorithmique

7 Quand elle ne s'efface pas derrière ces figures mythiques, l'informatique se trouve régulièrement réduite au problème des algorithmes. La centralité croissante du concept d'algorithme se comprend d'ailleurs aisément. D'une part, les algorithmes paraissent trouver toujours plus d'applications concrètes et plus de données à traiter, depuis la recommandation de biens culturels, comme la musique (Beuscart et al 2019), jusqu'à la diffusion d'informations (Benkler, 2018) en passant par la justice (Christin, 2017), la police prédictive (Benbouzid, 2017) et tant d'autres, sans oublier les sciences sociales elles-mêmes (Edelman et al. 2020). D'autre part, les algorithmes ne cessent, semble-t-il, de gagner en autonomie grâce à des systèmes tels que les réseaux de neurones (Cardon et al. 2018). La crainte est que ces technologies, présentées comme capables d'apprendre, absorbent les inégalités inhérentes aux données et reproduisent des dispositifs d'oppression (Noble, 2018). Les algorithmes s'inscrivent désormais dans une double mise en scène, qui les présente à la fois dans leur manque de transparence et dans leur toute-puissance. La force du « drame algorithmique » (Ziewitz, 2017) tient à ce jeu entre opacité et pouvoir qui se renforcent mutuellement. Même en partant du principe que les algorithmes ont du pouvoir, il n'est pas aisé de comprendre comment ils l'exercent, ce qui les rend d'autant plus mystérieux. Inversement, leur opacité peut être interprétée comme un signe supplémentaire de leur puissance. Bref, à défaut de savoir ce que sont et ce que font les algorithmes, tout le monde s'accorde sur leur statut d'objets de pouvoir.

8 Pourtant, les sociologues et les anthropologues qui suivent la piste des algorithmes rencontrent rarement de telles entités car, dans l'activité informatique, le mot désigne des objets très différents, aux contours parfois changeants. Par exemple, pour beaucoup de développeurs, "algorithme" fait souvent référence aux séquences d'instructions classiques qu'ils ont apprises sur les bancs des Universités ou de leur école d'ingénieurs. Dans ce sens, les algorithmes se manifestent lorsqu'une développeuse révise ses connaissances d'algorithmique en vue d'un entretien d'embauche ou qu'un ingénieur cherche sur un forum une implémentation précise du *quicksort*[5]. Si complexes ou impressionnants soient-ils, les algorithmes ne sont qu'un élément parmi d'autres au milieu d'une vaste boîte à outils — nous voici bien loin des mystérieux objets de pouvoir que nous mentionnions plus tôt. À l'inverse, si nous entendons par algorithme toute séquence d'instructions qui peut être exécutée par un ordinateur, l'acception devient si large qu'elle pourrait englober chaque aspect de la production informatique, au point que l'objet peine à être opérant pour les sciences sociales. Nick Seaver fait très justement remarquer qu'en informatique, le mot « algorithme » connaît une trajectoire semblable à celui de « culture » en anthropologie (Seaver, 2017), devenu si populaire à l'intérieur et à l'extérieur de sa discipline qu'il en est presque galvaudé. Dans ces circonstances, l'enquête sociologique peine à définir et à identifier ces fameuses boîtes noires. Et bien souvent, lorsqu'elle les trouve, les ouvrir s'avère extrêmement difficile, le tout pour un résultat déroutant, parfois décevant (Winner, 1993).

9 L'algorithme n'est donc pas toujours la bonne porte d'entrée pour comprendre l'activité informatique ; une concentration excessive sur les algorithmes peut même finir par occulter une grande partie des pratiques, des savoirs et surtout du sens de l'activité informatique. Notre propos n'est pas de contester l'algorithme en tant

qu'objet sociologique — plusieurs articles de ce numéro traitent d'ailleurs de cette question en l'abordant sous un angle différent — mais plutôt de le remettre à sa place parmi les nombreux autres objets de l'activité informatique.

1.3. S'emparer de l'écriture des codes

10 Notre proposition d'envisager l'activité informatique comme un travail d'écriture trouve de nombreux échos dans diverses traditions de recherche. Notons, au tournant des années 1990, la volonté de sortir l'histoire de l'informatique de ses préoccupations internalistes pour l'inscrire dans les questions contemporaines de l'histoire des techniques (Mahoney, 1988) et, plus précisément, des technologies de l'information (Kranakis, 1994 ; Aspray, 1994). C'est au même moment qu'un vent nouveau souffle sur plusieurs disciplines, du design à la sociologie, qui se demandent comment l'informatique assiste (ou peut assister) la coopération dans le travail : les études sur le *Computer Supported Cooperative Work* (CSCW) ont ouvert bon nombre de débats théoriques et pratiques, par exemple autour de la performativité[6] (Suchman, 1994 ; Winograd, 1994). Au début des années 2000, c'est au tour des *Software Studies* de réunir un groupe pluridisciplinaire, des *Media Studies* à l'ingénierie, pour appréhender les logiciels comme objets et pratiques culturelles[7]. Plus tard, les *Critical Code Studies* incitent à se concentrer plus spécifiquement sur le code source ; elles sont les premières à participer assidûment à la construction du code comme objet textuel, résultant d'une pratique d'écriture. Dans la littérature scientifique francophone, la sociologie du travail s'est également penchée sur l'organisation de la production logicielle dans le contexte du logiciel libre (Demazière et al., 2007), tandis que la sociologie de l'écriture s'est appropriée le code en prolongeant l'intuition des *Critical Code Studies* de le considérer comme un texte (Couture, 2012). Si bien d'autres avant nous ont ouvert la voie de l'écriture informatique, nous pensons que cette piste mérite d'être explorée jusqu'au bout et qu'elle réserve encore bien des trouvailles. Mais que signifie, au juste, étudier l'informatique comme elle s'écrit ?

11 S'intéresser à l'écriture des codes signifie tout d'abord faire passer l'activité au premier plan. Le but est de se détacher, ne serait-ce qu'un instant, des figures mythiques et des discours qui l'entourent afin de penser cet objet dans ses aspects ordinaires, routiniers, vulnérables. Écrire un logiciel ne se résume pas à produire du nouveau code. Bon nombre d'opérations relèvent par exemple d'un « *care* des choses » à l'égard d'infrastructures scripturales qu'il faut maintenir (Denis & Pontille, 2012 ; Denis & Pontille, 2020). D'autres consistent simplement à comprendre et à gérer le code existant (Couture, 2012). Comme dans toute étude anthropologique des infrastructures, l'enquête fait la lumière sur ces actions laborieuses, ennuyeuses ou invisibles pour mieux explorer ce jeu entre le transparent et l'opaque (Star, 1999). Comprendre l'écriture du code dans toutes ses nuances révèle comment ces différences servent de support à des distinctions, à des stratégies de légitimation, par exemple lorsqu'un ingénieur automatise ce qu'il considère comme du travail « manuel » ingrat (Alcaras, 2020), et plus largement à la construction d'un ethos professionnel (Zarca, 2009).

12 Étudier l'informatique comme elle s'écrit implique aussi de prendre en compte le rapport que les codeur·euses entretiennent avec cette activité. Ce parti pris s'inscrit résolument dans une démarche sociologique qui a le souci de la technique (Desrosières, 2013 ; Dagiral et Martin, 2017) et qui refuse de « laisser de côté les réalités vécues par

celles et ceux qui développent, distribuent et promeuvent les logiciels et technologies concernés » (Vinck & al., 2018). Au-delà de cette approche compréhensive, le rapport à l'écriture logicielle participe à la construction des identités au sens professionnel (Perrenoud et al, 2018) comme au sens large à travers, par exemple, la performance du genre (Faulkner, 2000) dans un monde du travail dominé par les hommes (Jorgenson, 2002 ; Collet, 2006). Enfin, puisque le travail ainsi que le rapport à l'activité informent la structure et les valeurs des professions (Abbott, 1988), s'intéresser au contenu et à l'expérience de l'écriture logicielle éclaire les mondes professionnels de l'informatique dans leur ensemble.

13 Prendre pour objet l'écriture des codes nous invite enfin à considérer cette activité dans toute sa matérialité, en commençant par les conditions concrètes de travail. Si l'informatique s'écrit partout, elle s'écrit différemment selon ces conditions de production. Les mondes informatiques témoignent en effet d'une vaste hétérogénéité de statuts (de la développeuse freelance à l'ingénieur salarié), de contextes (espaces militants, service technique dans une autre industrie, entreprise star de la Silicon Valley), d'échelles (des petits scripts personnels aux vastes projets des multinationales du numérique) et d'espaces (bureau traditionnel, espaces de co-working, télétravail). L'analyse de la matérialité de l'écriture logicielle se poursuit dans le rapport à la matière du code. Alors que l'algorithme permet de penser en termes de dispositif, le code incite à se pencher sur l'acte concret d'écriture des programmes, c'est-à-dire la production d'un texte qui sera interprété ou compilé par les machines, inséré dans des infrastructures matérielles et logicielles, lu et amendé par des collègues, copié-collé par des amateur·ices ou hobbyistes et ainsi de suite. En ancrant solidement notre analyse de l'informatique dans la matérialité du code, nous pouvons alors rendre compte de la diversité, des tensions ou des conflictualités qui traversent ces pratiques, c'est-à-dire comment s'écrivent les codes.

14 L'ambition de ce numéro spécial n'est pas de proclamer que tout est code ou que le code serait le seul objet valable ; cela reviendrait à dénoncer un réductionnisme pour le remplacer immédiatement par un autre. Séparer complètement le programme du code serait artificiel ; en revanche, cette distinction peut nous aider à penser l'informatique sous tous ses angles. Ainsi, si programmer se rapporte à la planification et à l'architecture informatique, alors coder désigne le rapport concret et matériel à la construction logicielle. En nous intéressant de près aux codes, nous ne documentons pas seulement une activité essentielle et quotidienne de l'informatique : nous pouvons également observer et analyser comment d'autres objets, algorithmes ou autres, se manifestent dans les pratiques matérielles. C'est pourquoi ce numéro spécial prend l'écriture informatique à la fois comme objet d'analyse et comme porte d'entrée dans « l'intérieur » des infrastructures numériques.

2. Poser d'anciennes questions à un nouvel objet

15 L'informatique a beau s'être démocratisée, massifiée puis banalisée durant les quatre dernières décennies, le code donne encore aujourd'hui l'impression d'être un nouvel objet pour la sociologie. Cette sensation de nouveauté s'explique probablement par le peu de recherches empiriques qui existent au regard de l'immensité du terrain informatique. Par conséquent, le code peut parfois apparaître comme un objet effrayant, demandant des efforts considérables de théorisation pour se laisser

apprivoiser. Selon nous, l'écriture informatique présente justement l'avantage de se trouver à l'intersection de nombreuses traditions de recherches. Autrement dit, cet objet permet de mobiliser un vaste éventail d'ancrages théoriques, de questions heuristiques et de stratégies empiriques. Dans l'esprit de la revue RESET, ce numéro spécial est donc une invitation à emprunter et à faire évoluer des questions sociologiques anciennes, voire classiques, afin de les appliquer aux codes.

2.1. Du pouvoir des algorithmes aux savoir-faire informatiques

16 Les discours sur les figures mythiques du numérique comme sur les algorithmes insistent, chacun à leur manière, sur la puissance de l'informatique. La représentation du hacker met l'accent sur le rôle émancipateur de cette activité pour l'individu, tandis que la réduction algorithmique se drape plus volontiers dans les plis du déterminisme technologique. Malgré leurs différences, ces discours posent d'abord la question du pouvoir. S'intéresser à l'écriture des codes, c'est mettre de côté — au moins temporairement — la question de ce que peut l'informatique pour enquêter sur ce que savent les informaticien·nes.

17 Comment une ingénieure trouve-t-elle la bonne ligne de code pour résoudre un bug ? Pourquoi un codeur choisit-il telle convention de style plutôt qu'une autre ? Comment un architecte système se sent-il contraint par des décisions prises plus de trente ans auparavant ? Comment un groupe de développeurs développe-t-il une représentation collective du code sur lequel ils travaillent ? Que veut dire cette personne quand elle déclare devoir réparer cette infrastructure « à la main » ? Nous proposons, en somme, de revenir aux premières questions de l'anthropologie cognitive et de les appliquer aux personnes qui écrivent les logiciels. Qu'ont-elles besoin de savoir pour faire ce qu'elles font ? Comment s'y prennent-elles pour savoir ce qu'elles savent (Hutchins, 1994) ? L'écriture informatique se révèle alors comme un vaste ensemble de pratiques, de savoirs et, surtout, de savoir-faire.

18 Étonnamment, ce déplacement du politique vers l'épistémique ne limite pas l'informatique à un simple exercice intellectuel. Au contraire, étudier l'informatique comme elle s'écrit nous invite à la considérer comme une activité concrète. Le code se présente comme un texte, c'est-à-dire comme une matière qui dispose de ses propres contraintes. Cette matière devient un objet de connaissance et d'action grâce à de nombreux outils qui composent un environnement numérique. En tant qu'inscription, le code se trouve *de facto* dès le départ étroitement inséré dans une vaste infrastructure scripturale (Denis, 2018) qui résulte elle-même d'une longue histoire de construction et de maintenance. Analyser l'écriture informatique, c'est enquêter sur des savoir-faire en prise avec la matérialité du texte, qui s'appuient sur des connaissances et sur des actions situées dans un environnement scriptural, qui héritent des contraintes techniques passées et qui se développent autour de normes et de représentations collectives.

19 En mettant la lumière sur ce que savent les travailleur·euses du code, nous nous apercevons aussi de tout ce qui échappe à leur connaissance. Écrire du code, c'est constamment se heurter à des zones d'ombre, à des imprévus, ne serait-ce que dans le cas anodin d'un bug qui se manifeste sans cause apparente. L'ignorance peut aussi être plus radicale et pourtant ne poser aucun problème, par exemple quand des ingénieurs utilisent un algorithme sans comprendre comment il fonctionne. La sociologie de

l'écriture informatique peut alors dessiner les contours des savoirs informatiques en action, expliquer pourquoi certaines zones d'ombre passent inaperçues tandis que d'autres deviennent des territoires à explorer, et souligner les stratégies mises en œuvre à la fois pour contourner l'obscurité ou pour y faire la lumière.

2.2. L'acte d'écriture pour interroger le pouvoir du code

20 Petit à petit, interroger ces savoir-faire de l'écriture informatique précise ce que peuvent faire les codeur·euses et, *in fine*, ce que peut véritablement le code. L'analyse fine de ces pratiques évite le piège du « fétichisme du code » dénoncé par Wendy Chun, une pionnière des études du code (Chun, 2008). Cet écueil consiste, dit-elle, à confondre le code source et son exécution par la machine. Cette confusion s'explique par l'histoire militaire et genrée de l'informatique : « dans l'armée, il n'est pas censé y avoir de différence entre un ordre donné et un ordre exécuté » (p. 304). Cette approche critique du pouvoir du code l'amène à contester la célèbre expression « code is law » de Lawrence Lessig. Selon la chercheuse, ce précepte exprime un fantasme de ce que devrait être la loi, fondé sur une image embellie de l'exécution du code. Elle invite plutôt à s'étonner du fait qu'un logiciel puisse résulter d'un code et à se demander comment le code peut devenir exécutable.

21 Si l'objet algorithmique permet de réfléchir en termes de dispositifs, il favorise une vision intellectuelle de la programmation, détachée des moyens avec lesquels le code est implémenté. Rappelons que la programmation n'est pas nécessairement liée au numérique, ni même aux ordinateurs (Aspray, 1990). Avant l'apparition des premiers langages alphanumériques dans les années 1950, les algorithmes étaient directement programmés à l'aide d'artefacts matériels, comme les fameuses cartes perforées du métier Jacquard ou les machines tabulatrices (Gardey 2008, p. 263–267). Alan Turing (1937) emploie d'ailleurs le terme *computer* pour désigner un « calculateur », c'est-à-dire non pas une machine, mais un être humain qui calcule (Mélès, 2015). De même, les dernières évolutions du *machine learning* rendent possible l'exécution d'instructions qui n'ont pas été explicitement codées. Compris comme méthodes systématiques pour résoudre un problème, les algorithmes existent même en dehors de la sphère numérique, par exemple dans la résolution de casse-têtes comme le Rubik's Cube. En somme, pour emprunter les termes de l'anthropologie de l'écriture, le code et les algorithmes désignent des technologies intellectuelles et des cultures matérielles différentes (Goody, 1979) qui, si elles se recoupent parfois, ne sont pas équivalentes.

22 En revanche, penser l'exécution du code comme un acte d'écriture (Fraenkel & Pontille, 2003 ; Fraenkel, 2007) met l'accent sur l'ensemble de conditions matérielles, techniques et sociales qui permettent au code de prendre effet. L'exécution est un long chemin, qui n'aboutit pas toujours. Au-delà des contraintes scripturales qui pèsent sur le code en tant que texte (syntaxe, style, intégration), nous pensons également aux phénomènes sociaux qui permettent par exemple de s'accorder sur quelle version du code est la « source » officielle. Enfin, suivre l'écriture informatique donne à voir la dualité entre le code et son exécution. Une ligne de code qui semble parfaitement limpide, qui s'exécute sans pépins, peut pourtant donner un résultat complètement imprévu — un fait que certains utilisent à leur avantage lors de concours de code « sournois »[8]. Les ingénieurs peuvent d'ailleurs jouer sur cette confusion entre code et exécution, par

exemple lorsqu'ils mettent en scène une intelligence qui n'est pas présente dans le code (voir dans ce numéro, Ionescu, 2022).

23 En même temps, nous pouvons aussi considérer comment l'acte d'écriture, une fois qu'il a pris effet, fait autorité dans les milieux informatiques. Cette autorité explique pourquoi les ingénieurs préfèrent recourir à des logiciels plutôt qu'à de simples guides de conduite pour faire respecter des normes techniques. Elle explique également pourquoi certains codes peuvent devenir un enjeu de luttes pour des groupes aux intérêts divergents. Penser l'informatique comme une forme d'écriture interroge ainsi la construction technique et sociale d'un pouvoir scriptural, tout en prenant au sérieux la réalité des effets que produit l'écriture (Denis, 2018).

2.3. Discours de codification, pratiques du code

24 Ainsi l'activité d'écriture du code n'échappe-t-elle pas au processus de « rationalisation intellectualiste que nous devons à la science et à la technique scientifique » (Weber, 1959). En effet, certains discours insistent sur le caractère « maîtrisé », « efficace » et « prévisible » de la pratique. Plutôt que de renforcer cette illusion d'un domaine entièrement normé et rationalisé, à l'instar des « discours de vérité » des promoteurs du Web 2.0 (Bouquillion et Matthews, 2010), nous appelons à « montrer comment les acteurs sociaux négocient la signification de chacun de ces mots avant de les imposer aux autres comme des vérités premières » (Callon, 2013). Pour répondre à ces discours, bien des recherches montrent que les usages d'une technique peuvent échapper à ses producteur·ices (Jouët, 2000 ; Proulx, 2015), par exemple quand les utilisateur·ices s'approprient, détournent ou réinventent les scripts (Akrich 1987 ; 2010). Les contributions de ce numéro empruntent le pas à ces travaux et les prolongent en examinant les usages des personnes qui produisent les logiciels. Autrement dit, coder n'est pas seulement créer ; c'est aussi utiliser des outils et des techniques.

25 Cette perspective nous conduit à distinguer les pratiques du code et les discours sur le code, qui sont souvent des discours de codification. Dans un texte intitulé « Habitus, code et codification » (1986), Pierre Bourdieu s'intéresse aux processus sociaux de formalisation des normes, en particulier dans les domaines juridique et linguistique. Selon lui, « codifier, c'est à la fois *mettre en forme* et *mettre des formes* » : les règles censées formaliser la pratique deviennent un enjeu de lutte et de contrôle en elles-mêmes. L'écriture informatique est traversée par des phénomènes similaires. Nous y observons des processus de codification aussi bien formels, par exemple la mise en place et la négociation de standards du web au sein du W3C, qu'un jeu sur des pratiques informelles, comme l'usage du copier-coller à partir de la plateforme de questions/réponses StackOverflow. La maîtrise du code s'accompagne donc d'une maîtrise des codes, qu'ils soient explicites ou implicites.

26 Le respect des normes (« se mettre en règle »), leur subversion (« jouer avec les règles ») ou leur abrogation (« briser les règles ») n'auront pas la même valeur symbolique selon le contexte : rationalisation et optimisation des processus de production dans l'ingénierie, valeurs prônées par l'éthique hacker, appréciation esthétique du code comme forme d'art (Knuth, 1974), proximité avec la « matrice disciplinaire » académique (Millet, 2003). En étudiant les rapports entre activité et discours, entre code et codification, entre le « faire » et le « dire sur le faire » (Lahire,

1998), les écrits informatiques se placent dans la continuité des écrits littéraires, scolaires et ordinaires.

27 Pour ce faire, les propos des codeurs qui donnent une cohérence et un sens aux pratiques *a posteriori* doivent être considérés comme tels. Voilà pourquoi étudier l'informatique comme elle s'écrit ne peut faire l'économie d'une observation empirique de l'écriture, afin d'éclairer un ensemble de pratiques informelles, de micro-pratiques et de pratiques invisibles. La sociologie de l'écriture informatique se trouve alors confrontée à deux défis. Le premier défi est de surmonter une double difficulté méthodologique : que signifie « observer » dans le contexte d'un terrain numérique ? Comment comprendre ce qui se joue dans ces pratiques sans s'appuyer systématiquement sur les discours de codification, mais sans non plus oublier l'importance de ces derniers ? Le deuxième défi pose, quant à lui, le problème de l'écriture scientifique. Comment décrire une activité foisonnante sans perdre le fil de l'analyse ? Comment rendre compte du désordre sans faire désordre ? Comment mettre en forme ces compte-rendus sans reprendre à notre compte les mêmes discours de codification ?

28 De toute évidence, ces interrogations traversent toute recherche en sciences sociales. Si la tâche paraît encore difficile en ce qui concerne l'informatique, c'est sans doute que nous ne disposons encore que d'un petit nombre de travaux qui documentent les embûches spécifiques à ces terrains (Mahoney, 2008). Nous sommes donc d'autant plus heureux de regrouper, dans ce numéro spécial de RESET, sept recherches qui proposent chacune une manière originale d'aborder l'écriture informatique, chacune répondant par l'exemple et à sa manière aux questions que nous venons de soulever.

3. Enquêter sur l'écriture informatique

29 Ce numéro s'ouvre sur une enquête dans le monde des développeurs de la Silicon Valley, qui situe l'informatique dans ses conditions matérielles d'écriture. Olivier Alexandre y explore le quotidien et les spécificités de carrières professionnelles caractérisées par un fort degré d'incertitude. À travers des observations, des entretiens et un suivi d'activité en ligne, l'auteur présente les multiples stratégies mobilisées par ces travailleurs pour répondre aux différentes manifestations de ces incertitudes, que ce soit dans les opérations de conception et de maintenance, dans l'orientation au sein des projets ou dans leurs trajectoires professionnelles. Loin des représentations d'un travail maîtrisé et prévisible, la carrière de développeur informatique est dépeinte comme une course d'orientation au sein d'une pluralité de médiations et d'outils. En faisant apparaître les « prises » individuelles susceptibles d'orienter l'action et de réguler l'incertitude, par exemple l'obsolescence rapide de certaines compétences, cette contribution nous renseigne plus finement sur les processus à l'œuvre au sein d'un marché du travail traversé par des injonctions récurrentes à « l'évolution, l'agilité et la fluidité ».

3.1. Savoir et ignorance dans la production logicielle

30 Après cette analyse des carrières de l'ingénierie logicielle, deux ethnographies nous plongent dans le détail de l'écriture informatique, en particulier dans ses aspects épistémiques et pratiques.

31 Tout d'abord, Tudor Ionescu consacre une étude ethnographique d'une grande richesse à une pratique courante et pourtant considérée comme mauvaise, le *hardcoding*. Cette pratique consiste à spécifier un mécanisme ou une donnée de la manière la plus explicite possible dans le code lui-même, sans se donner la peine de réfléchir à une solution plus générale à un problème précis. À travers plusieurs vignettes, l'auteur suit les péripéties d'une équipe de développement qui doit prochainement faire la démonstration d'un robot intelligent sur une chaîne d'assemblage. Or les ingénieurs n'ont ni le temps, ni les moyens de développer cette technologie. Ils ont donc recours au *hardcoding* et dictent de façon extrêmement précise et — littéralement — millimitrée les mouvements que la machine doit effectuer, plutôt que d'inventer une intelligence qui déciderait des mouvements par elle-même. Alors que le code reflète l'étendue de ce que la machine ne sait pas faire, son exécution aboutit à une performance convaincante de la part du robot, qui donne l'impression que cette intelligence artificielle est à portée de main. En plus de documenter un grand nombre de problèmes typiques de l'écriture informatique industrielle (dette technique, *integration hell*), cette enquête propose une réflexion passionnante sur la dualité entre code et exécution ainsi que sur la question de la performativité. L'auteur montre également comment l'écriture du code est directement liée aux contraintes de l'économie de la promesse, au fonctionnement par projets et aux exigences de production.

32 L'article suivant ouvre une fenêtre sur une séquence particulièrement ordinaire de l'écriture informatique : un épisode de débuggage. Dans une étude méticuleuse, Florian Jaton choisit de décortiquer étape par étape une scène de quelques minutes à peine afin de mieux entrer dans la profondeur — parfois vertigineuse — de chaque moment d'écriture. Au sein d'un laboratoire de traitement de l'image et au côté d'une chercheuse qui rencontre un bug, l'auteur montre comment la codeuse adopte temporairement une attitude d'enquêtrice et cherche à comprendre d'où vient le problème. Pendant quelques instants, son poste de travail se transforme en une sorte de laboratoire qui sera le théâtre de multiples petites expérimentations visant à isoler la ligne responsable du bug. L'auteur observe et décrit de près ces diverses expériences, qui vont de l'utilisation de documentation, de forums en ligne et d'instruments techniques comme un débuggueur. Au gré de cette description dense, au formalisme original, l'activité gagne progressivement en consistance. La chercheuse mobilise différents savoir-faire pour renseigner l'état « d'entités lointaines » mal connues, comme l'interpréteur, à travers une série d'inscriptions (code, documentation, sortie de l'interpréteur) qu'il faut parvenir à aligner correctement. Ces petits moments de production de connaissances, assez proches de la démarche scientifique, font la lumière sur ces instants anodins et éphémères de l'écriture du code.

3.2. La codification du code : du style aux standards

33 Les pratiques du code, comme le *hardcoding* ou le débuggage, se trouvent souvent en prise à des processus de codification. Ces interactions entre code et codification font l'objet des deux contributions suivantes de ce numéro. La première revient sur la standardisation dans le domaine du web ; la seconde s'intéresse à la codification du style dans l'industrie informatique.

34 En s'intéressant aux arènes de standardisation que sont le W3C et l'IETF, Julien Rossi décrit les discussions, les controverses et les formes de consensus qui s'y développent. Dans ces espaces où la communication a lieu au croisement de rencontres en présence et d'outils numériques de coordination, l'auteur documente les enjeux méthodologiques à multiplier les types de matériaux. Entretiens, observations, analyses de compte-rendus de réunions et de listes mail ouvrent ainsi des pistes pertinentes, à condition d'en identifier les limites respectives et de resituer la genèse de ces sources. Dans ces espaces où « l'injonction au consensus » est omniprésente, les acteurs privilégient le régime de justification technique pour exprimer leurs désaccords et leurs critiques. Cette norme est renforcée et régulée par les formes de sanction et de (re)cadrages qui ont lieu dans les débats. Le croisement des matériaux donne alors à voir les postures critiques et politiques des intervenants et révèle les stratégies discursives qui se déploient dans ces espaces de négociation.

35 Pierre Depaz aborde, quant à lui, une autre sphère de la codification de l'écriture informatique : la régulation et la négociation du style. L'auteur y analyse la tension entre la dimension subjective du style, comme une préférence personnelle dans l'utilisation d'un point-virgule à la fin d'une ligne de code, et l'émergence de normes collectives. Ces dernières visent à maintenir la cohérence du style sur l'ensemble d'un projet, même quand plusieurs personnes y participent. Ces normes sont souvent codifiées dans des guides de style, qui constituent l'objet central de l'article. En comparant trois guides de style populaires qui s'appliquent au langage Javascript, l'auteur cherche à comprendre comment le contenu de ces guides est négocié, quels arguments sont mobilisés et quels registres de justification finissent par l'emporter. Si le répertoire argumentatif est commun aux trois guides, chacun présente ses spécificités, en particulier quand le guide s'accompagne d'un logiciel qui relève, voire corrige automatiquement les passages qui dérogent à ses normes. L'article explore ainsi, parmi d'autres tensions entre code et codification, la force de l'argument d'autorité technique.

3.3. L'autonomie de l'écriture informatique en question

36 Ce numéro s'achève par deux contributions qui étudient comment l'écriture informatique s'insère dans un ensemble de pratiques industrielles. Elles examinent ainsi l'autonomie relative de l'écriture informatique, tout d'abord face à des processus d'insécurisation dans le cadre du développement d'un protocole de chiffrement ; ensuite face à un ensemble de pratiques de guidage dans le contexte de l'élaboration d'algorithmes de recommandation.

37 Dans une enquête sur des développeurs qui travaillent sur un protocole cryptographique, Sylvain Besençon explore les enjeux autour de la sécurité du code informatique. En s'intéressant à la fois aux moments critiques et aux instants routiniers de maintenance et de soin apportés au code, il explicite les efforts constants pour le rendre moins vulnérable. Loin de la représentation d'un univers de la sécurité régulièrement perturbé par des failles et des coups d'éclat, il insiste sur la dimension collective et cyclique de sécurisation et d'insécurisation du code. Chaque nouvelle inscription risque de rendre le code plus fragile, même lorsque l'écriture vise à résoudre un problème de sécurité. En renseignant des pratiques peu documentées à l'aide d'un matériau varié (conférences publiques, entretiens, observations, discussions

en ligne, plateformes de partage du code), l'auteur invite à déplacer le regard sensationnel porté sur les enjeux de sécurité vers l'examen de pratiques routinières de clarification, d'amendements ponctuels et de maintenance quotidienne. L'(in)sécurisation est ainsi analysée comme un processus de continuité plutôt que de rupture, inscrit dans des dynamiques de collaboration au croisement de différentes sphères d'acteurs (organisations, entreprises, monde académique).

38 Le dernier article montre comment un certain nombre d'outils, notamment algorithmiques, équipent l'écriture du code. Camille Roth et Jérémie Poiroux examinent de nombreux entretiens avec des personnes responsables du développement d'algorithmes de recommandation. Les auteurs montrent que cette écriture s'appuie sur un grand nombre d'infrastructures et d'outils algorithmiques afin de réparer, d'ajuster ou de faire évoluer le code de leurs propres algorithmes. Ils s'intéressent en particulier à la pratique fort répandue, mais jusqu'ici peu documentée, de l'*A/B testing*. Cette dernière consiste à proposer différentes versions d'une même fonctionnalité afin de quantifier l'effet de chaque variation sur les usages d'une plateforme donnée. L'exploration de ces pratiques souligne que, contrairement à l'image d'une ingénierie logicielle complètement rationalisée et planifiée, nombre de développeurs bricolent. Ils ne cherchent pas vraiment à ouvrir les boites noires algorithmiques qu'ils déploient ; la plupart se contentent de multiplier les tests afin de retenir les variations qui semblent donner le meilleur résultat. Les auteurs discutent des conséquences théoriques et méthodologiques de ce résultat puisqu'*in fine*, cette forme d'écriture informatique semble moins traversée par des problèmes algorithmiques que par des enjeux de quantification et de benchmarking.

4. Conclusion. L'informatique comme pratique et comme croyance

39 Dans leur fameux article *L'informatique comme pratique et comme croyance*, Michel Gollac et Francis Kramarz (2000) constatent « l'impossibilité d'un "champ informatique" autonome » en parlant de toute pratique liée à l'utilisation d'un ordinateur, de la bureautique à l'écriture du code. Sans négliger les recherches qui considèrent les programmes et leurs usages numériques dans leur globalité, ce numéro montre l'intérêt d'adopter une définition plus précise de l'informatique, saisie par les pratiques du code. Si la piste de l'écriture informatique nous semble particulièrement prometteuse, elle n'est qu'un point de départ parmi d'autres pour explorer « l'intérieur » des infrastructures numériques.

40 Poser de nouvelles questions, construire de nouveaux objets : ce numéro est avant tout une invitation à explorer l'informatique sous toutes ses coutures, avec une curiosité et un enthousiasme renouvelés. Au-delà des questions théoriques et de leurs éclairages sur l'activité informatique, nous espérons surtout que les contributions de ce numéro contribueront à fournir des exemples concrets d'enquêtes et de compte-rendus originaux sur ce type de terrain, susceptibles d'encourager l'exploration des pratiques et des croyances des mondes informatiques.

BIBLIOGRAPHIE

ABBOTT A., 1998, *The System of Professions. An Essay on the Division of Expert Labor*, University of Chicago Press, 452 p.

AKRICH M., 1987, « Comment décrire les objets techniques ? », *Techniques & Culture. Revue semestrielle d'anthropologie des techniques*, 9.

AKRICH M., 2010, « Retour sur « Comment décrire les objets techniques ? » », *Techniques & Culture. Revue semestrielle d'anthropologie des techniques*, 54-55, p. 202-204.

ALCARAS G., 2020, « Des biens industriels publics. Genèse de l'insertion des logiciels libres dans la Silicon Valley », *Sociologie du travail*, *62*, 3.

ASPRAY W., (dir.) 1990, *Computing before computers*, Iowa State University Press, 266 p.

ASPRAY W., 1994, « The history of computing within the history of information technology », *History and Technology*, *11*, 1, p. 7-19.

AURAY N., 2007, "Le modèle souverainiste des communautés en ligne : impératif participatif et désacralisation du vote", Hermès, n°47, p.137-145.

AURAY N., OUARDI S., 2014, « Numérique et émancipation », *Mouvements*, 3, p. 13-27.

AURAY N., VÉTEL B., 2013, « L'exploration comme modalité d'ouverture attentionnelle », *Réseaux*, 6, p. 153-186.

BARON G.-L., 2018, « Informatique et numérique comme objets d'enseignement scolaire en France: entre concepts, techniques, outils et culture »,.

BENBOUZID B., 2017, « Des crimes et des séismes », *Reseaux*, *206*, 6, p. 95-123.

BENKLER Y., FARIS R., ROBERTS H., 2018, *Network Propaganda: Manipulation, Disinformation, and Radicalization in American Politics*, New York, Oxford University Press, 472 p.

BERREBI-HOFFMANN I., BUREAU M.-C., LALLEMENT M., 2018, *Makers-Enquête sur les laboratoires du changement social*, Média Diffusion.

BEUSCART J.-S., COAVOUX S., MAILLARD S., 2019, « Les algorithmes de recommandation musicale et l'autonomie de l'auditeur », *Reseaux*, *213*, 1, p. 17-47.

BOUQUILLION P., MATTHEWS J.T., 2010, *Le Web collaboratif: mutations des industries de la culture et de la communication*, Presses Universitaires de Grenoble.

BOURDIEU P., 1986, « Habitus, code et codification », *Actes de la Recherche en Sciences Sociales*, *64*, 1, p. 40-44.

BROCA S., s. d., *Utopie du logiciel libre*, Passager clandestin (Le).

CALLON M., 2013, « Pour une sociologie des controverses technologiques », dans AKRICH M., LATOUR B. (dirs.), *Sociologie de la traduction : Textes fondateurs*, Paris, Presses des Mines (Sciences sociales), p. 135-157.

CARDON D., COINTET J.-P., MAZIÈRES A., 2018, « La revanche des neurones », *Reseaux*, *211*, 5, p. 173-220.

CHRISTIN A., 2017, « Algorithms in practice: Comparing web journalism and criminal justice », *Big Data & Society*, *4*, 2, p. 2053951717718855.

CHUN W.H.K., 2008, « On "Sourcery," or Code as Fetish », *Configurations*, *16*, 3, p. 299-324.

COLEMAN E.G., 2010, « The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld », *Anthropological Quarterly*, *83*, p. 42-76.

COLLET I., 2006, *L'informatique a-t-elle un sexe?*, Editions L'Harmattan.

COUTURE S., 2012, « L'écriture collective du code source informatique », *Revue d'anthropologie des connaissances*, *6*, 1.

DAGIRAL É., MARTIN O., 2017, « Liens sociaux numériques », *Sociologie*, N° 1, vol. 8.

DEMAZIÈRE D., HORN F., ZUNE M., 2007, « Des relations de travail sans règles ? L'énigme de la production des logiciels libres », *Societes contemporaines*, *66*, 2, p. 101-125.

DENIS J., PONTILLE D., 2012, « Travailleurs de l'écrit, matières de l'information », *Revue d'anthropologie des connaissances*, *6*, 1, p. 1-20.

DENIS J., 2018, *Le travail invisible des données : Éléments pour une sociologie des infrastructures scripturales*, Paris, Presses des Mines (Sciences sociales), 206 p.

DENIS J., PONTILLE D., 2020, « Maintenance et attention à la fragilité », *SociologieS*.

DESROSIÈRES A., 2013, *Pour une sociologie historique de la quantification: L'Argument statistique I*, Presses des Mines via OpenEdition, 331 p.

DODIER N., 1995, *Les hommes et les machines: la conscience collective dans les sociétés technicisées*, FeniXX.

DROT-DELANGE B., BRUILLARD E., 2012, « Éducation aux TIC, cultures informatique et du numérique: quelques repères historiques », *Études de Communication. Langages, information, médiations*, 38, p. 69-80.

EDELMANN A., WOLFF T., MONTAGNE D., BAIL C.A., 2020, « Computational Social Science and Sociology », *Annual Review of Sociology*, *46*, 1, p. 61-81.

ERMOSHINA K., 2017, « Le code peut-il réparer les routes? », *Reseaux*, 6, p. 155-189.

FAULKNER W., 2000, « Dualisms, Hierarchies and Gender in Engineering », *Social Studies of Science*, *30*, 5, p. 759-792.

FLICHY P., 2017, *Les nouvelles frontières du travail à l'ère numérique*, Média Diffusion.

FRAENKEL B., 2007, « Actes d'écriture : quand écrire c'est faire », *Langage et société*, *121-122*.

FRAENKEL B., PONTILLE D., 2003, « L'écrit juridique à l'épreuve de la signature électronique, approche pragmatique », *Langage et societe*, *104*, 2, p. 83-122.

GALLOWAY A.R., 2012, *The interface effect*, Polity.

GARDEY D., 2008, *Écrire, calculer, classer: Comment une révolution de papier a transformé les sociétés contemporaines (1800-1940)*, Paris: La Découverte.

GOLLAC M., KRAMARZ F., 2000, « L'informatique comme pratique et comme croyance », *Actes de la Recherche en Sciences Sociales*, *134*, 1, p. 4-21.

GOODY J., 1975, *La raison graphique*, Editions de Minuit, Paris

HIMANEN P., LEBLANC C., 2001, *L'éthique hacker et l'esprit de l'ère de l'information*, Exils Paris.

HUTCHINS E., 1994, *Cognition in the wild*. Cambridge, MIT Press.

JORGENSON J., 2002, « Engineering Selves: Negotiating Gender and Identity in Technical Work », *Management Communication Quarterly*, *15*, 3, p. 350-380.

JOUËT J., 2000, « Retour critique sur la sociologie des usages », *Réseaux. Communication - Technologie - Société*, *18*, 100, p. 487-521.

KRANAKIS E., 1994, Introduction, *History and Technology,* 11, pp. 1-5

KELTY C.M., 2008, *Two bits: The cultural significance of free software*, Duke University Press.

KNUTH D.E., 1974, « Computer programming as an art », *Communications of the ACM*, *17*, 12, p. 667-673.

LAHIRE B., 1998, « Logiques pratiques : le « faire » et le « dire sur le faire » », *Recherche & formation*, *27*, 1, p. 15-28.

LALLEMENT M., 2015, *L'Âge du Faire. Hacking, travail, anarchie: Hacking, travail, anarchie*, Média Diffusion.

LESSIG L., 1999, *Code: And other laws of cyberspace*, Basic Books.

MAHONEY M.S., 1988, « The History of Computing in the History of Technology », *Annals of the History of Computing*, *10*, 2, p. 113-125.

MAHONEY M.S., 2008, « What Makes the History of Software Hard », *IEEE Annals of the History of Computing*, *30*, 3, p. 8-18.

MÉADEL C., SIRE G., 2017, « Les sciences sociales orientées programmes », *Reseaux*, 6, p. 9-34.

MÉLÈS B., 2015, « L'informatique sans ordinateur », Site *CNRS : Images des mathématiques*.

MILLET M., 2021, *Les Étudiants et le travail universitaire : Étude sociologique*, Lyon, Presses universitaires de Lyon (Hors collection), 256 p.

MOATTI A., 2012, « Le numérique, adjectif substantivé », *Le débat*, 3, p. 133-137.

NOBLE S.U., 2018, *Algorithms of Oppression: How Search Engines Reinforce Racism*, New York University Press.

PERRENOUD M., SAINSAULIEU I., 2018, « Pour ne pas en finir avec l'identité au travail », *SociologieS*.

PROULX S., 2015, « La sociologie des usages, et après? », *Revue française des sciences de l'information et de la communication*, 6.

ROSENTAL C., 2017, « Les conditions sociales des échanges dans la Silicon Valley », *Zilsel*, *1*, 1, p. 55-81.

SEAVER N., 2017, « Algorithms as culture: Some tactics for the ethnography of algorithmic systems », *Big Data & Society*, *4*, 2, p. 2053951717738104.

STAR S.L., 1999, « The Ethnography of Infrastructure », *American Behavioral Scientist*, *43*, 3, p. 377-391.

STEVENS H., 2012, « Autonomie récusée, autonomie fabriquée. Informaticiens à l'épreuve de l'Entreprise de Soi », *Genèses*, *87*, 2, p. 90-112.

SUCHMAN L., 1994, « Do categories have politics? The language/action perspective reconsidered », *Computer Supported Cooperative Work (CSCW),* 2, p. 177-190.

TURING A.M., 1937, « On Computable Numbers, with an Application to the Entscheidungsproblem », *Proceedings of the London Mathematical Society*, *s2-42*, 1, p. 230-265.

TURNER F., 2006, *From counterculture to cyberculture*, University of Chicago Press.

VICENTE M., 2017, « Apprentissage du code informatique et entrepreneuriat: de la création d'entreprise à l'esprit d'entreprendre », *Formation emploi. Revue française de sciences sociales*, 140, p. 87-106.

VINCK D., CAMUS A., JATON F., OBERHAUSER P.-N., 2018, « Localités distribuées, globalités localisées: actions, actants et me·diations au service de l'ethnographie du nume·rique », *Symposium*, *22*, 1, p. 41-60.

WEBER M., 1959, *Le savant et le politique*, Paris, Plon.

WINNER L., 1993, « Upon Opening the Black Box and Finding It Empty: Social Constructivism and the Philosophy of Technology », *Science, Technology, & Human Values*, *18*, 3, p. 362-378.

WINOGRAD T., 1993, « Categories, disciplines, and social coordination », *Computer Supported Cooperative Work (CSCW)*, *2*, 3, p. 191-197.

ZARCA B., 2009, « L'ethos professionnel des mathématiciens, The professional ethos of mathematicians. », *Revue française de sociologie*, *50*, 2, p. 351-384.

ZIEWITZ M., 2016, « Governing Algorithms: Myth, Mess, and Methods », *Science, Technology, & Human Values*, *41*, 1, p. 3-16.

NOTES

1. Le terme français « informatique » présente ainsi selon nous l'avantage de considérer ensemble ce qui est souvent séparé dans la langue anglaise, entre *computer science* (la discipline scientifique) et *software engineering* (le génie industriel).
2. Une API, acronyme de *Application Programming Interface*, désigne communément un système qui permet à plusieurs applications de dialoguer entre elles.
3. Nous empruntons le terme « émique » à la tradition anthropologique (Sardan, 1998). Il fait référence aux discours et aux représentations qui ont du sens à l'intérieur du monde social étudié, en l'occurence ici l'informatique.
4. Voir la déclaration de l'investisseur Mark Andreesen, « Why Software Is Eating The World », *Wall Street Journal,* 20 août 2011 : https://www.wsj.com/articles/SB10001424053111903480904576512250915629460 (consulté le 15 janvier 2022). Toutes les traductions de l'anglais ont été réalisées par nos soins.
5. Le *quicksort*, ou « tri rapide », appartient à la famille des algorithmes de tri, qui visent à ordonner les éléments d'une structure de données, comme une liste ou un tableau. Réputé pour sa rapidité dans la plupart des situations, cet algorithme utilise un système de pivot : il sélectionne un élément autour duquel les autres éléments sont permutés, de sorte que ceux qui lui sont inférieurs se trouvent à sa gauche et ceux qui lui sont supérieurs se trouvent à sa droite. L'opération est répétée récursivement dans chaque partition.
6. Dans ce texte, nous utilisons le vocabulaire de la « performance » non pas au sens d'exécution efficace d'une tâche, mais au sens de la « performativité » théorisée par Austin (1962), développée par Butler comme « la pratique réitérative et citationnelle par laquelle le discours produits les effets qu'il nomme » (2009, p. 2).
7. Pour des exemples de travaux qui s'inscrivent dans les *Software Studies*, voir par exemple la revue *Computational Cultures*, fondée en 2011.
8. Voir par exemple http://underhanded-c.org/ : « Le *Underhanded C Contest* est un concours annuel visant à écrire du code C apparemment innocent, mais au comportement malveillant ».

RÉSUMÉS

Les infrastructures numériques se présentent comme des dispositifs fascinants, à la fois incroyablement complexes et parfaitement ordonnés. Alors que ces infrastructures font l'objet de nombreux discours, nous ne savons que peu de choses sur leurs pratiques de production et de maintenance, surtout au regard de leur étendue et de leur diversité. Comment se créent les logiciels que nous utilisons chaque jour ? Qui construit les infrastructures informatiques de nos sociétés contemporaines ? De quoi le numérique est-il fait ? Ce texte tisse une toile de fond théorique pour introduire les contributions du numéro spécial. Afin de dépasser les figures mythiques (hacker, geek, entrepreneur) et d'échapper à la réduction algorithmique, s'emparer de l'écriture des codes offre de multiples prises aux enquêtes empiriques. Cet objet se trouve également à l'intersection de plusieurs traditions de recherche. Elles ouvrent ainsi des pistes prometteuses sur les savoir-faire informatiques, la performativité ou encore les processus de codification.

INDEX

Mots-clés : Code, infrastructure, numérique, écriture, pratiques

AUTEURS

GABRIEL ALCARAS

Centre Maurice Halbwachs, i3

ANTOINE LARRIBEAU

EXPERICE (Université Sorbonne Paris Nord)