Design *in* Translation

Software

Kim Sacks

---

1. Définition du terme importé en français

Le terme de « Sofware », parfois traduit en français par « logiciel », demeure dans certains contextes employé tel quel en français. La définition qui suit permet de s'en faire une idée générale de ce que le terme englobe.

> *« A simple definition of software is that it is the set of instructions that direct a computer to do a specific task. Every machine has it*[1]*. »*

Paul E., CERUZZI, *A History of Modern Computing* – 2nd ed., Cambridge, MA, The MIT Press, 2003, p. 80.

Pour approfondir cette définition succincte, rappelons que l'informatique, les *computer sciences* et tous autres domaines associés aux machines computationnelles, séparent les termes de *software* et de *hardware.* Ces deux notions sont intrinsèquement liées et pensées simultanément tout en donnant lieu à deux définitions autonomes et complémentaires. Cette scission permet de percevoir les machines en distinguant symboliquement les entités concrètes (microcontrôleurs, processeurs, RAM etc.) des entités abstraites (programmes, code sources, algorithmes etc.) Le terme software qualifie l'ensemble de ces entités abstraites. Il se compose des méthodes, des logiques, des programmes, qui gouvernent les machines. La notion est souvent, si ce n'est exclusivement, expliqué en relation au hardware, qui sert de fondement théorique pour circonscrire une définition négative : si le software en lui-même est difficile à délimiter, il est tout ce que le hardware n'est pas, c'est-à-dire tout ce qui n'est pas strictement de la matière sensible, concrète. C'est ce que confirme l'occurrence suivante :

> *« SOFTWARE n.m. est emprunté (1966) à l'argot des ingénieurs américains, formé plaisamment sur le modèle de hardware « quincaillerie », par opposition de soft « mou » à hard « dur ». Soft est d'origine germanique (néerlandais zacht). Cet anglicisme, abrégé en soft n. m. (1971) et entré dans le vocabulaire international de l'informatique, désigne l'ensemble des moyens d'utilisation, des programmes, etc., d'un système informatique, par opposition aux éléments matériels. L'équivalent français logiciel (1972), largement répandu, a pratiquement éliminé software, mais soft et hard n.m. sont toujours en usage. »*

Alain, REY (dir.), *Dictionnaire Historique de la langue Française*, Paris, Dictionnaires Le

Robert, 1998, t. 3, p. 3535.

En revanche, si nous nous affranchissons des définitions usuelles et des dictionnaires étymologiques pour explorer les définitions encyclopédiques, nous nous rendons compte que le terme *software* ne donne pas lieu à une entrée propre dans l'*Encyclopédie Universalis*, alors qu'il est présent dans plusieurs définitions périphériques comme celles de *logiciels* ou *systèmes d'exploitation*. Nous le retrouvons notamment au sein de la définition du mot *ordinateur* pour décrire l'architecture von Neumann, soit le modèle des machines dont le programme est enregistré dans une mémoire définit par le mathématicien John von Neumann :

> *« L'ensemble formé par l'unité arithmétique et logique, d'une part, et l'unité de commande, d'autre part, constitue l'unité centrale ou processeur. L'ensemble des composants physiques, appelé matériel (hardware), est commandé par un logiciel (software), suite cohérente d'instructions et de données structurées stockées en mémoire et exécutant un algorithme ».*

Danièle, DROMARD, François, PÊCHEUX, « ORDINATEURS », Encyclopædia Universalis [en ligne], consulté le 29 septembre 2021. URL : http://www.universalis-edu.com/encyclopedie/ordinateurs/

Comme l'attestent les deux extraits précédents, le terme logiciel sert régulièrement de traduction directe de *software*. Toutefois, l'anglicisme demeure d'usage dans certains cas, probablement parce que l'informatique théorique se développe historiquement aux États-Unis. Les termes du champ disciplinaire restent bien souvent ceux de la langue d'origine, l'anglais, pour permettre un système lexical commun qui s'adosse à une syntaxe programmatique dérivée de l'anglais. Les deux exemples suivants permettent d'illustrer différents contextes d'utilisation du terme *software*. Pour la théoricienne et chercheuse Olga Goriunova, le terme est utilisé pour qualifier par extension, tout un champ, comme le Software Art, qui provient de pratiques techniques spécifiques.

> *« Le Software Art est devenu le domaine qui remettait en cause l'invisibilité et la neutralité des logiciels, mettant en évidence les partis pris et discriminations politiques, sociaux et culturels qui l'animent. Aujourd'hui, l'écriture ou l'utilisation de logiciels sont loin de se réduire à une simple manipulation directe d'idées. Une couche de programmation n'est pas transparente, elle peut restreindre ou élargir le champ des possibles. »*

Olga, GORIUNOVA, « L'histoire de Runme.org, répertoire de Software Art », dans LARTIGAUD, David-Olivier (dir.), *Art++*, Orléans, Hyx, 2011, p. 117.

Dans cet extrait, notons que le terme software est utilisé en même temps que logiciel, en ce que le premier permet de qualifier un domaine et le second un outil. Notre exemple suivant de Jean Beaudrillard illustre que le terme de logiciel ne délimite pas exactement les mêmes enjeux conceptuels que le software, et favorise en ce sens l'usage de ce dernier.

> *« Les prothèses de l'âge industriel sont encore externes, exotechniqnes, celles que nous connaissons se sont ramifiées et intériorisées : ésoTechniques. Nous sommes à l'âge des technologies douces, software génétique et mental. »*

Jean, BAUDRILLARD, *Simulacres et simulation*, Paris, Éditions Galilée, 1981, p. 150.

2. De la langue d’origine au français

Le terme *Software* est donc emprunté de l’anglais. Les premières occurrences du terme original font références au textile et se définit en opposition directe au hardware, en substituant le préfixe hard- (dur, rigide) par soft- (mou, doux). Il est introduit pour la première fois dans le champ sémantique de l’informatique en 1958 afin de qualifier les routines programmatiques. Si, par ce déplacement du terme anglais vers le français, le préfixe persiste, la matérialité du terme français *software* disparaît au profit d’une définition d’un concept immatériel, symboliquement détaché du médium, comme le suggère John W. Tukey :

> *« Today the "software" comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its "hardware" of tubes, transistors, wires, tapes and the like*[2]*. »*

Dans son passage de l’anglais au français, le terme a conservé ce détachement en rapport à son médium. Dès son apparition dans le champ lexical de l’informatique, la dichotomie instaure une hiérarchie entre les machines et les logiques qui les gouvernent. Elle permettait d’appréhender, avec un terme consacré, l’immatérialité des objets traités par les machines. Cette dichotomie sémantique conditionne l’*informatique théorique* à aborder les machines par la dualité suivante : l’abstrait versus le concret. Pourtant, des théoriciens et théoriciennes émettent des hypothèses divergentes quant à la simplification linguistique au dualisme conceptuel. Dans l’exemple suivant, cette dichotomie pourrait être réagencée en introduisant davantage de termes pour requalifier d’autres aspects des machines (comme le contenu pour T. Nielson).

> *« In publishing, the terms "hardware" and "software" have for some reason been adopted as meaning objects (such as physical books) and content (what's printed in them). This is unfortunate, since in computer-based text systems we must distinguish between the hardware (computer and reading screen), software (computer and display program) and content (what is read)*[3]*. »*

Les contours du concept semblent donc variables selon les approches théoriques et le contexte lequel le terme s’inscrit. Notre second exemple atteste que la stricte dichotomie *software/hardware* ne relève pas de la nature des machines mais plutôt d’une détermination pragmatique, c’est-à-dire d’un usage dépendant directement de son contexte. Autrement dit, l’usage de cette distinction est convenable mais ne reposerait sur aucun fondement ontologique.

> *« since programming can occur on many levels, it is useful to understand the software/hardware dichotomy as a pragmatic distinction. […] This pragmatic view of the sofware/hardware distinction makes the distinction both understandable and useful. A myth concerning the software/hardware distinction can arise, however, if the distinction is understood and taken to have more ontological significance than it has*[4]*. »*

Le terme *software* pourrait être traduit par *mentaille*. Il qualifie ce qui relève *du mental*, par analogie à la *quincaillerie*, mot servant quelquefois de traduction au *hardware.*

3. Concept et problématisation

Selon James H. Moor, la dichotomie software/hardware est donc un mythe. De fait, elle ne repose sur aucun fondement ontologique et n'a de sens que lorsqu'elle permet de clarifier les tâches qui incombent à chaque corps de métiers impliqués dans la conception des systèmes informatiques (développeurs, ingénieurs, designers d'interfaces etc.) Elle rend possible une opposition entre un niveau symbolique (langage, programme, etc.) et un niveau matériel (microcontrôleurs, circuits logiques etc.) Et cette distinction pragmatique est toute relative puisque ce que certains qualifieraient de software, ce que d'autres qualifieraient de hardware. En revanche, lorsque l'on s'interroge sur la définition du software dans son essence, il n'y a pas lieu pour aucune distinction entre celui-ci et le hardware. Le software n'existe que par son médium ; c'est en outre un des points de l'exposé de Friedrich Kittler[5] allant dans le sens d'une dépendance inextricable entre la machine et le software. L'articulation entre une logique abstraite et un artefact serait l'implémentation[6], autrement dit, le programme est l'implémentation d'un algorithme dans un médium.

James H. Moor appuie son argument en soulignant que les premiers softwares sont *hardwired*, c'est-à-dire, câblés directement dans le circuit, comme on peut le voir avec les programmes des *ENIAC Girls*, et ne permettent en ce sens aucune définition ontologique du software. Il approfondit son argument en émettant l'hypothèse suivante : même si l'on fixait une délimitation entre le software et le hardware, il en reste qu'au niveau applicatif, le potentiel de simulation du hardware par du software demeurerait puisque, dans ce cas, le hardware pourrait être décrit en termes de fonction computable[7]. Si l'on revient à l'idée de la machine universelle de Turing[8], celle-ci repose sur la possibilité de simuler la machine par la machine, et donc, sur la capacité des machines à « abstraire » des logiques hardware. Autrement dit, la machine universelle n'est conceptuellement envisageable qu'à partir du moment où la délimitation hardware/software n'est pas « figée » puisque le hardware peut théoriquement relever d'une simulation software.

La substitution d'usage en français, dans de nombreux cas, par le terme *logiciel* souligne davantage l'intrication des concepts, le terme provenant de *logique* et *matérielle*, proposé par la Délégation à l'informatique en 1969 et adopté par l'Académie Française en 1972. Pourtant, une distinction perdure en ce que l'usage de machines comme des outils techniques scindent les domaines « d'expertises » : le hardware relèverait strictement de l'ingénierie tandis que le software demeure encore flou quant à son champ, brouillant quelque peu les rôles des designers/concepteurs quant à leur relation à l'ingénierie. Un designer de programmes (*software designer*) ne limite pas sa pratique à une approche mentale des logiques programmatiques mais transpose ces logiques en programme par le biais d'un langage, lui-même dépendant d'une infrastructure technique matérielle.

La cohabitation, dans l'usage français courant, des termes software et logiciel illustre les ambiguïtés des définitions : là où le logiciel qualifie davantage les applications utilisateurs (avec des interfaces homme-machine), le software est défini comparativement au hardware, au matériel et en ce sens encapsule tous les aspects des machines qui relèvent d'une logique programmatique. Or, sur ce dernier point, le software se distingue du logiciel par sa capacité à s'émanciper de la nécessité de la représentation, et en conséquence, de l'interface utilisateur. Le software qualifie également les programmes qui font exclusivement interagir des machines entre elles, ou d'autres programmes entre-eux. C'est sur ce point que le texte de Jack Burnham, publié à l'occasion de l'exposition *Software* de 1970[9], aborde la résistance du software à la représentation.

De même, Nurbay Irmak[10] propose de penser le software comme un *artefact abstrait*. Il postule que, si un artefact est la production intentionnelle d'une activité humaine, alors, nécessairement, le software est un artefact. Toutefois, pour Irmak, en l'absence d'existence spatio-temporelle dans un objet concret, le software ne peut en conséquence qu'être défini par une forme d'artefact abstrait qui échappe aux définitions philosophique d'« objet abstrait » en

ce qu'il relève d'un « type », d'une « idée » qui survivrait dans l'esprit de l'auteur malgré une potentielle destruction de toutes les implémentations dans du hardware. En conséquence, il avance que la double nature du software est une contradiction. En ce sens, il s'oppose à la définition de Timothy Colburn[11] qui distingue l'*abstraction concrète* du software aux machines constituées par des textes, en appuyant le fait que la double nature du software ne peut être définit qu'en comprenant la nécessité simultanée d'un « médium de description » (langage d'expression des algorithmes) et d'un « médium d'exécution » (un circuit hardware).

En comparant, entre autres, les définitions de Colburn, Irmak, Moor, Burnham, il ressort que le terme software englobe des nombreux concepts liés aux machines computationnelles, sans pour autant avoir une définition claire de ses contours, en particulier dans sa relation au hardware. Qu'il s'agisse d'une définition supportant l'inséparabilité hardware/software ou d'une définition argumentant en faveur d'une double nature, de façon transversale à ces différentes approches, le concept de software s'inscrit dans un ensemble de termes connexes du champ qui aident à circonscrire le software : il est distinct des programmes et des algorithmes (on pourra se référer à la définition de Knuth[12]), de la data, du bruit, des patterns d'informations (hypothèse soutenue par Peter Suber[13]), du texte, de la copie ou encore de l'exécution même du software (séparation soulignée par Irmak[14]). Notons que la confusion sémantique entre le software et les autres termes du champ est également présente pour d'autres concepts dont la substitution est d'usage dans le langage courant, comme par exemple *programme* et *algorithme*.

Kim SACKS, Maître de conférences en Design, Université de Strasbourg.

1. Nous traduisons : « Une définition simple du software est qu'il s'agit de l'ensemble des instructions qui gouvernent un ordinateur pour effectuer une tâche spécifique. Chaque
2. Nous traduisons : « Aujourd'hui, le "software", qui comprend les routines d'interprétation soigneusement planifiées, les compilateurs et d'autres aspects de la programmation automatique, est au moins aussi important pour la calculatrice électronique
3. Nous traduisons : « Dans le domaine de l'édition, les termes "
4. Nous traduisons : « étant donné que la programmation peut se faire à plusieurs niveaux, il est utile de comprendre la dichotomie software/
5. Friedrich, KITTLER, « There Is No Software », *Stanford Literature Review*, 9:(1) (Spring 1992), pp 81-90.
   Friedrich, KITTLER, « There Is No Software », *The Truth of the Technological World: Essays on the Genealogy of Presence*, Redwood City, California, United States, Stanford University Press, 2013, pp. 219-229.
6. William J., RAPAPORT, « Implementation Is Semantic Interpretation », *The Monist*, 82(1), 1999, pp. 109–130.
   William J., RAPAPORT, « Implementation as Semantic Interpretation : Further Thoughts », *Journal of Experimental & Theoretical Artificial Intelligence*, 17(4), 2005 pp. 385–417.
7. James H. MOOR, *op. cit.*, p. 216.
8. Alan TURING, « Computing Machinery and Intelligence », *Mind*, vol. LIX, no. 236, Oxford UK, Oxford University Press, 1950.
9. Jack BURNHAM, « Notes on art and information processing », *Software :Information technology : its new meaning for art*, New York: Jewish Museum, 1970
10. Nurbay, IRMAK, « Software is an Abstract Artifact », *Grazer Philosophische Studien*, 86(1), 2012, pp. 55-72.
11. Timothy R. COLBURN, « Software, Abstraction, and Ontology », *The Monist*, 82(1), 1999, pp. 3-19. doi:10.5840/monist19998215 et Timothy COLBURN & Gary SHUTE, « Abstraction in Computer Science », *Minds and Machines*, 17(2), 2007, pp. 169-184. doi:10.1007/s11023-007-9061-7
12. Donald E. KNUTH, *The Art of computer programming volume I, Fundamental Algorithms, (third edition)*, Massachusetts, Addison-Wesley, 1997, pp. 1-10.
13. Peter SUBER, « What Is Software? », *Journal of Speculative Philosophy*, 2(2), 1988, pp. 89-119.
14. Nurbay, IRMAK, *op. cit.*