Exemple Référence et Bibliographie

Florian Rascoussier

March 2022

1 Exemple

1.1 Petite Histoire du code

La programmation est une technique dont les prémisses remontent au début du XIXème siècle, avec l'invention du métier à tisser Jacquard. Il faut cependant attendre le milieu du XXème siècle, et notamment les travaux du mathématicien anglais Alan Turing et son article fondateur de la science informatique, « On computable numbers, with an application to the entscheidungsproblem », pour que cette technique se développe réellement.

1.2 Le Langage

Le terme *langage* est souvent associé en premier lieu à une faculté intrinsèque de l'homme. En informatique, il désigne la façon dont instructions et données sont codées et de les manipuler. Le langage a enfin un sens différent du point de vue de l'artiste. Aborder le sujet du langage et du code requiert ainsi de bien comprendre les notions couvertes par le terme en question.

Pour le dictionnaire Larousse, le langage à donc de multiples définitions : «

- Faculté propre à l'homme d'exprimer et de communiquer sa pensée au moyen d'un système de signe vocaux ou graphiques ; et ce système.
- Système structuré de signes non verbaux remplissant une fonction de communication.
- Ensemble des procédés utilisés par un artiste dans l'expression de ses sentiments et de sa conception du monde.
- Mode d'expression propre à un sentiment, à une attitude.
- Ensemble de caractères, de symboles et de règles permettant de les assembler, utilisé pour donner des instructions à un ordinateur.

— (machine) Langague directement exécutable par l'unité centrale d'un ordinateur, dans lequel les instructions sont exprimées en code binaire.

» [2]. On remarque que ces définitions recoupent en parties celles que l'on a pu voir dans les parties précédentes au sujet du code. Code et langage sont ainsi très lié. Du point de vue du programmeur, le langage est en effet la langue qui définit comment organiser les instructions et les données afin de produire un programme [3].

Un langage de programmation est un outil pour le programmeur. En effet, tout programme étant par définiton une suite d'instructions machines c'est-à-dire de 0 et de 1, il n'est pas aisé pour un humain d'écrire ses programmes directement dans ce langage bas niveau. Cependant, il est possible d'écrire des programmes dans des langages compréhensible par un humain mais qui puisse être tout de même être transformé en langage machine pour être exécuté par un ordinateur. De mêmes que les langues parlées, il existe une multitude de langages de programmation. Comme le dit AMADE, «Il existe des pratiques de programmation très différentes, il existe aussi des langages de programmation qui proposent des modes d'approche très différentes» [3]. La variété des premières expliquant naturellement la multiplicité des seconds, en plus d'autres facteurs comme les évolutions de la recherches, l'histoire jusque même aux goûts, habitudes et usages des programmeurs.

Le choix du ou des langages que l'on souhaite utiliser pour écrire un programme est donc un aspect important. Comme l'écrit DIJKSTRA dans son livre *A Discipline of Programming* en 1976 : «[...] one is immediately faced with the question : Which programming language am I going to use ?, and this is not a mere question of presentation ! A most important, but also most elusive, aspect of any tool is its influence on the habits of those who train themselves in its use. If the tool is a programming language, this influence is, -whether we like it or not- an influence on our thinking habits.» [4].

La création d'un langage de programmation est en elle-même un tâche complexe. En effet de très nombreux éléments rentrent en comptent dans la création d'un langage. Comment les rappellent les auteurs de *Introduction à la théorie des langages de programmation*, «Nous sommes encore très loin d'avoir trouvé un langage de programmation définitif. Presque chaque jour, de nouveaux langages sont créés et de nouvelles fonctionnalités sont ajoutées aux langages anciens. [...] La première chose que l'on doit décrire, quand on définit un langage de programmation, est sa syntaxe. Faut-il écrire x := 1 ou x = 1 ? Faut-il mettre des parenthèses après un if ou non ? Plus généralement, quelles sont les suites de symboles qui forment un programme ? On dispose pour cela d'un outil efficace : la notion de grammaire formelle. À l'aide d'une grammaire, on peut décrire la syntaxe d'un langage de manière qui ne laisse pas de place à l'interprétation et qui rende possible l'écriture

d'un programme qui reconnaît les programmes syntaxiquement corrects. Mais savoir ce qu'est un programme syntaxiquement correct ne suffit pas pour savoir ce qui se passe quand on exécute un tel programme. Quand on définit un langage de programmation, on doit don également décrire sa sémantique, c'est-à-dire ce qui se passe quand on exécute un programme. Deux langages peuvent avoir la même syntaxe mais des sémantiques différentes.» [5]. Un langage de programmation est ainsi composé plusieurs éléments qui permettent à une machine de comprendre et d'excuter le programme sans ambiguité.

En plus ces considérations, les langages se définissent par la ou les grands approches qu'ils choisissent de considérer ou favoriser. Ce sont les paradigmes. Dans la conférence intitulée *L'importance des langages en informatique* en 4 nov. 2015 à l'INRIA de Rennes, Berry reviens sur le processus créatif derrière la création des nombreux langages. Au départ, on peut considérer deux langages A et B différents avec des approches fondamentalent uniques l'un par rapport à l'autre. Chacun forme sa propre communauté d'utilisateur. Puis arrive un nouveau langage C qui ne propose pas un nouveau point de vue mais reprend les concepts des deux langages précédents pour tenter tant bien que mal de les associéer. Ce dernier viens grapiller des utilisateurs aux deux premiers. Les langages A et B originaux se mettent donc à incorporer des concepts de l'autre afin de récupérer des utilisateurs et venir proposer de nouvelles améliorations à ses utilisateurs.«C'est-à-dire que vous avez des tas de gens qui ont des idées complètement baroques. Vous avez des tas de groupes d'utilisateurs avec des traditions totalement différentes dans des pays qui n'ont rien à voir [...] et la paysage deviens très joyeusement incompréhensible. Et c'est là vie, parce que c'est un espace de création. Difficile.» [6]. Berry propose ainsi une vision caricaturale mais néanmoins descriptive derrière l'apparition et l'évolution des langages informatiques.

Références

[1] A M Turing. « On computable numbers, with an application to the entscheidungsproblem ». In : *Proc. Lond. Math. Soc. (3)* s2-42.1 (1937), p. 230-265.

[2] Claude Nimmo et Larousse (Firm). *Le petit Larousse illustré*. 21, rue de Montparnasse 75283 Paris Cedex 06 : Larousse, 2017.

[3] Bernard Amade. *Introduction à la programmation*. Paris : MA Editions, 2019.

[4] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976, p. vii-30.

[5] Gilles. Dowek et Jean-Jacques Levy. *Introduction à la théorie des langages de programmation*. Ellipses, 2006.

[6] *L'importance des langages en informatique*. INRIA de Rennes, 4 nov. 2015.

Bibliographie complémentaire

[7] M Romero, B Lille et A Patiño. *Usages créatifs du numérique pour l'apprentissage au XXIe siècle*. Presses de l'Université du Québec, 2017.

[8] *code*. fr. `https://fr.wiktionary.org/wiki/code`. Consulté le 19-3-2022.

[9] Florian Cramer. « Digital code and literary text ». In : *Beehive Hypertext/Hypermedia Literary Journal* (27 sept. 2001).

[10] Andy Oram et Grew Wilson. *L'art du beau code*. 1ère Ed. Paris : Edition O'REILLY, 2008. 621 p. isbn : 978-2-84177-423-4.

[11] Martin Fowler. *Refactoring*. 2nd Ed. Malakoff : Dunod, 2019. 419 p. isbn : 978-2-10-080116-9.

[12] Robert C. Martin. *Clean Code : a handbook of agile software craftsmanship*. Montreuil : Pearson, 2009. 457 p. isbn : 978-2-3260-0227-2.

[13] Dustin Boswell et Trevor Foucher. *The art of readable code*. 1st Ed. O'Reilly Media, Inc, 2011. 204 p. isbn : 978-0596802295.