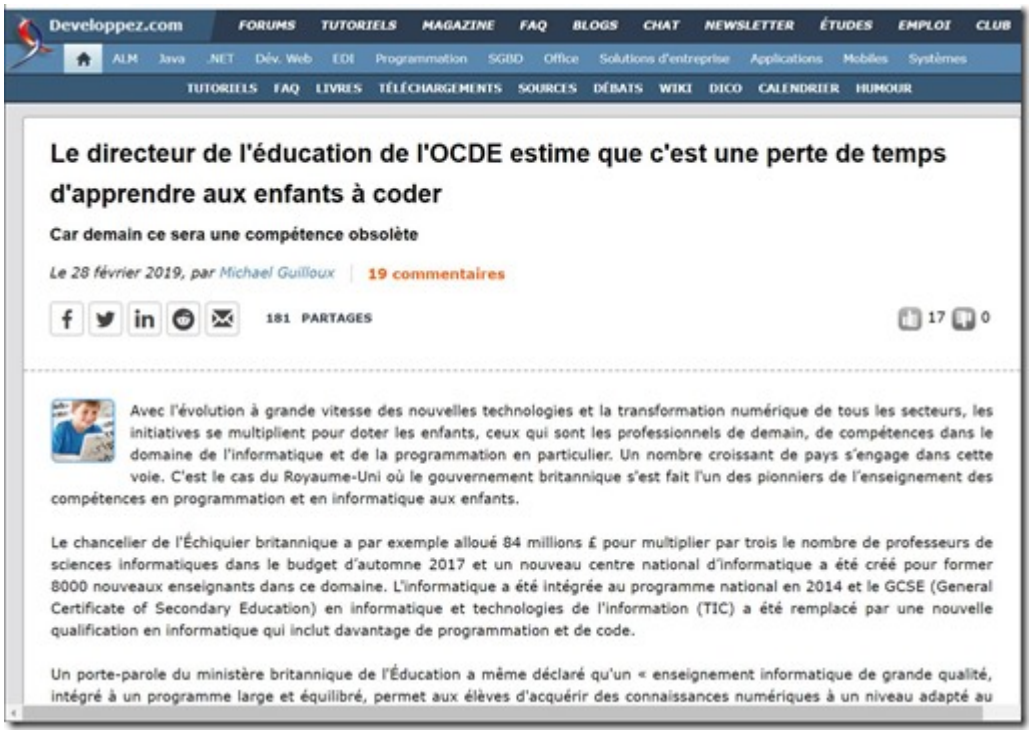
Opinions Libres
le blog d'Olivier Ezratty

Le mythe de la fin du développement logiciel

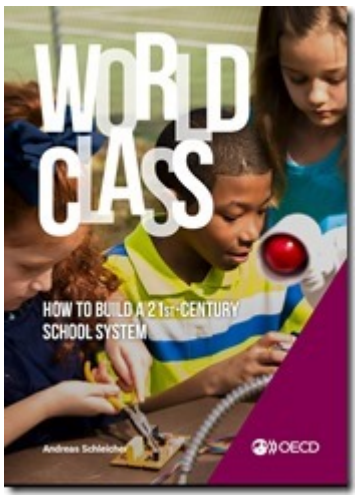
Un **article publié sur le site Developpez.com** fin février 2019 relançait il y a quelques jours un vieux débat sur l'intérêt d'apprendre "à coder" aux enfants. Il cite **Andreas Schleicher**, Directeur de l'Education et des Compétences au Secrétariat Général de l'OCDE. Il s'exprimait sur le sujet lors de la conférence "World Innovation Summit for Education" qui avait lieu à Paris fin février.

Selon lui "*enseigner aux enfants à coder est une perte de temps, car c'est [...] une compétence qui sera bientôt obsolète*". "*Cette compétence est simplement « une technique de notre temps* » et elle deviendrait inutile à l'avenir". Il considère qu'apprendre à coder n'est qu'une "*technique de notre époque*". Il pense que ce serait une grave erreur que cet "outils" soit enraciné. "*Il affirme qu'il est beaucoup plus enclin à enseigner la science des données ou la pensée informatique que d'enseigner une technique très spécifique d'aujourd'hui.*". Son point de vue intervient en réaction aux initiatives de nombreux pays d'apprendre le code aux enfants dès le plus jeune âge, comme au Royaume-Uni.

Voilà de quoi enflammer un beau débat, ce qui n'a pas manqué de se produire sur Twitter avec une réprobation plutôt majoritaire de ces propos. Mais lisez plutôt les commentaires de l'article de Développez.com qui sont, pour la plupart, un peu plus élaborés.



Le Docteur Schleicher n'est pourtant pas né de la dernière pluie. C'est un spécialiste de l'éducation qu'il a traitée en long et en large. Son avis est toutefois légèrement entaché d'un biais cognitif classique. Son métier d'origine étant la statistique, il n'est pas étonnant qu'il valorise la data science. Mais on ne peut juger un avis par quelques phrases sorties de leur contexte. Il est notamment l'auteur du très intéressant "World Class", un ouvrage OCDE **librement téléchargeable ici**. Il y déconstruit de nombreux mythes sur l'éducation dans le monde. C'est un peu un analogue de feu Hans Rosling (GapMinder) sur l'éducation. Cela mérite le détour.



Et cela tombe plutôt mal car, approchant de la **Journée Internationale des Droits des Femmes** (8 mars), on a droit à une très utile piqûre de rappel dans les médias sur la trop faible représentation des femmes dans les métiers techniques du numérique. Comment valoriser ces métiers si dans le même temps, certains donnent l'impression de les dévaloriser ?

Ci-dessous, **Marie-Claire Capobianco** de la BNP relayait un article sur cette triste réalité et propose d'orienter massivement les filles vers les "études de codages" (pourquoi au pluriel ?). Un contre-point saisissant à celui d'Andreas Schleicher.

Je réagissais au quart de tour à ce tweet en indiquant qu'il faudrait commencer par utiliser des expressions plus valorisantes comme ingénieur(e) logiciel, architecte logiciel ou développeur(euse). Evitons de définir un métier par une tâche ou un outils ! Privilégions la mission et l'objet réalisé, et aussi sa dimension humaine. Stéphane Zibi **renchérissait** à juste titre "*Et d'ajouter la créativité nécessaire, la passion que cela engendre et les aspects stratégiques de ces postes dans les entreprises*". D'autres évoquent aussi l'appellation ancienne d'analyste programmeur.

Avec le recul, Andreas Schleicher et Marie-Claire Capobianco incarnent deux facettes opposées du biais négatif associé au développement logiciel. L'un en relativise l'intérêt et l'autre le valorise un peu maladroitement. Ils confondent peut-être les outils et leur utilité. Ils se méprennent sur ce que l'on apprend en programmant.

Ce n'est peut-être qu'une question de jargon employé et de sens donné à ces différentes appellations. Je vais donc enfoncer quelques portes ouvertes sur le sujet dans ce qui suit.

Une remise en cause permanente des acquis

Quand on se met à programmer "avec du code", peu importe le langage de programmation. Il faut surtout se préparer au changement permanent pendant sa vie professionnelle.

Comme j'approche de la soixantaine, je suis passé par le FORTRAN (avec des cartes perforées… le top du top de l'apprentissage de la patience et de l'humilité pour le développeur !), le COBOL (très peu, fort heureusement pour moi) et le BASIC (sur un Pet Commodore en 1978 ou un TRS-80 en 1980). Entre deux cours de Math Sup, j'ai aussi développé plus d'une centaine de programmes pour ma calculatrice HP41C en notation polonaise inversée (dingue non ?). Ces différents langages n'ont plus du tout cours. J'ai aussi appris l'APL, qui est moins connu, un langage mathématique extrêmement concis qui sert notamment à manipuler des matrices. Plus personne ne l'utilise aujourd'hui mais c'était marrant. C'est ce que l'on appelle un langage "write once, read never" ! On peut l'écrire mais le lire est une sacrée paire de manche !

```
      ∇DET[□]∇
     ∇ Z←DET A;B;P;I
[1]   I←□IO
[2]   Z←1
[3]  L:P←(|A[;I])⍳⌈/|A[;I]
[4]   →(P=I)/LL
[5]   A[I,P;]←A[P,I;]
[6]   Z←-Z
[7]  LL:Z←Z×B←A[I;I]
[8]   →(0 1 ∨.=Z,1↑⍴A)/0
[9]   A←1 1 ↓A-(A[;I]÷B)∘.×A[I;]
[10]  →L
[11] ⍝EVALUATES A DETERMINANT
     ∇
```

Je suis ensuite passé par la programmation objet (C++) puis par la programmation événementielle à partir de Windows 1.0 à la fin des années 1980 et ensuite avec Javascript et AJAX il y a une dizaine d'années pour la création de plugins dans WordPress.

Plus récemment, j'ai découvert les méthodes déclaratives de création de réseaux de neurones ainsi que la programmation quantique qui remet beaucoup de choses en cause. A chaque fois, les outils étaient différents mais les bases apprises il y a des décennies m'étaient toujours utiles. Et pourtant, je ne suis plus développeur depuis 1989, au sens du métier principal. Les basiques du départ m'ont permis à chaque fois de découvrir de nouveaux langages, outils et architectures. Il faut surtout savoir parfois remettre les pendules à l'heure et évoluer.

Le changement permanent est le lot commun du développeur d'aujourd'hui qui passe de framework en framework aussi bien pour du développement web "full stack" (serveur et poste de travail, REACT, Angular, Node.js, et je ne dois plus être au point tellement cela bouge vite dans le domaine) ou pour créer des solutions de machine learning (de Theano à Tensorflow en passant par Caffee ou Pytorch). Le développement logiciel d'aujourd'hui est une juxtaposition d'une quantité de plus en plus grande d'outils spécialisés dont il faut maîtriser l'agencement.

Et les jeunes d'aujourd'hui qui développent auront eux-aussi à traverser de nouvelles générations d'outils qui transformeront incessamment leur métier. Je leur dis souvent que les outils d'aujourd'hui sont les cartes perforée de *dans 30 ans* !

La programmation au-delà du métier de développeur

Apprendre à développer, c'est aussi un moyen d'automatiser les tâches répétitives de sa vie numérique. Sans être développeur à temps plein, je créé ainsi des batches et macros diverses avec des outils qui n'ont rien d'extraordinaire (genre VBA dans Office). Ils me font gagner énormément de temps. C'est aussi cela "le codage" : savoir automatiser ce qui peut l'être pour devenir plus efficace. Cela me sert par exemple à produire le **Rapport du CES de Las Vegas** chaque année. Plein d'anciens développeurs ont conservé ces réflexes et pratiques. C'est leur petit secret de productivité !

En apprenant à créer du logiciel, petit ou grand, on développe sa capacité à structurer un raisonnement et à décomposer un problème en composantes. On (re)découvre la logique. Ce bénéfice est souvent mis en avant avec brio par **Aurélie Jean** dans ses différentes interventions comme lors de la conférence USI en juin 2017 (**vidéo**).

On s'aperçoit aussi qu'une bonne partie de la vie est une forme de programmation. Tous les jours, nous prenons des décisions dans nos actions qui prennent en compte un grand nombre de paramètres. Sans que l'on s'en rende compte, c'est une forme de programmation. Elle est moins explicite que dans la programmation, mais cela reste une forme de "code". Notre "libre arbitre" est ainsi la juxtaposition d'un nombre élevé de données de contexte, de notre passé, des règles apprises et de nos réactions hormonales à des émotions.

Nous passons d'ailleurs notre vie à mettre en œuvre les deux grands volets de l'intelligence artificielle : l'IA symbolique avec ses règles et sa logique et l'IA connexionniste qui consiste à apprendre par l'observation des données. Notre savoir empirique est ainsi construit par l'accumulation de règles que nous déduisons de l'observation : de la nature, des choses, des comportements, etc. Les règles et savoirs appris profitent de ce travail réalisé par les autres homo-sapiens. De la même manière, les parents programment les enfants avec son éducation mais l'enfant en bas âge "programme" aussi ses propres parents avec ses pleurs, cris et sourires. Action. Réaction.

Cela ne doit pas pour autant induire une approche "mécanique" de la vie et des relations humaines. On reproche souvent aux développeurs d'être trop introvertis et de ne pas s'intéresser assez aux autres et aux utilisateurs. C'est un reproche recevable pour une part d'entre eux/elles. Mais les développeurs n'ont pas l'exclusive de ce genre d'écueils. Plein d'autres métiers que je ne citerai pas et qui sont censés être plus "tournés vers les gens" présentent des travers voisins.

Ces travers déshumanisants sont aussi liés à la tendance à tayloriser un peu trop les métiers du logiciel, notamment dans les sociétés de services informatiques (les ESN : entreprises de services numériques). Un peu comme il existe des "stages ouvriers" dans les écoles d'ingénieur, on devrait peut-être créer des "stages utilisateurs" pour les développeurs, et les exposer plus aux personnes qui utilisent ou utiliseront leurs créations (c'est prévu dans les méthodes Agile).

La fin du code n'est pas pour demain

Et pourtant, la fin du code est une vieille rengaine. Je me souviens de mes débuts professionnels dans les années 1980 où la tendance était aux "langages de quatrième génération" qui allaient faire disparaître la programmation. Avec eux, on allait décrire les besoins métiers à haut niveau et les générateurs de code s'occuperaient du reste. C'était les beaux jours de la méthode MERISE de conception d'applications de gestion. La méthode a changé mais la programmation est restée.

Aujourd'hui, la grande vague de l'intelligence artificielle relance la tendance avec de nombreux futurologues – non développeurs en général – qui prévoient la fin de la programmation. C'est alimenté par la mode du machine learning qui décrit les méthodes de création de programmes qui apprennent eux-mêmes par les données. On oublie qu'en général, ces outils sont à la base de véritables logiciels qui requièrent tout de même de la programmation. Quand AlphaGo de Google DeepMind gagne au jeu de Go, c'est aussi le résultat de la création de logiciels ! Ce n'est pas magiquement sorti tout seul de l'ordinateur !

Mais l'IA peut aider les développeurs à améliorer leur productivité. Les environnements de développement sont de plus en plus productifs comme peut éventuellement l'être un traitement de texte. C'est ce qui est exposé dans l'article ci-dessous dont le titre est trompeur (**source**). Il existe aussi un outil dénommé **Bayou** qui est censé coder par lui-même en s'entraînant avec les millions de lignes de code stockées dans Github qui contient une bonne part des logiciels open

source du monde entier (**source**).

Dans **Is The End Of Code Really Coming?**, Ken Mazaika (2016) déconstruit efficacement ces mythes sur la fin de la programmation en rappelant que l'on n'a jamais autant codé. Et si des outils permettent de créer des sites web sans coder (WordPress de base, Wix, …), dès que l'on veut interagir avec des données ou ajouter une interaction non standard, il faut s'y coller ! Il conclue en démontrant que l'on n'a jamais autant eu besoin de développeurs pour une raison simple : le logiciel fait maintenant partie de tous les métiers et de tous les marchés. J'ajouterai que l'amélioration de la productivité de la production de logiciels divers a été accompagnée d'un élargissement constant de son marché.

Ken Mazaika rappelle aussi qu'un développeur fait bien plus que coder. Il doit déjà commencer par interpréter ou traduire les besoins métiers ! C'est une tâche parfois infinie, comme l'ont illustré à leurs dépends les projets serpents de mer type **Louvois** (paye de l'armée en France). Je reprend *ci-dessous* une illustration d'un commentaire de l'article de Developpez.com qui est éloquente !

Cet exercice de spécification des besoins est déjà une forme de programmation. La programmation intervient dès lors qu'il s'agit d'interpréter des règles métiers ! Là-dessus sont apparues les notions variées de RAD (rapid application development), d'Extreme Programming, de Lean et de méthodes agiles. En général, elles servent à raccourcir la boucle entre utilisateurs, spécifications et réalisations et à améliorer la flexibilité des développements logiciels qui doivent répondre à des demandes fluctuantes des utilisateurs.

Le développeur conçoit un système et il utilise des systèmes existants. Selon la taille du projet et des équipes, il intervient dans la définition de l'architecture du système. En termes de connaissances, un développeur d'aujourd'hui doit bien plus maîtriser les interface avec les nombreuses bibliothèques logicielles qu'il utilise (les "frameworks") que les constructions standards des langages de programmation qui les mettent en musique.

Ayant développé un plugin pour WordPress (depuis 2012 pour **gérer mes photos**), je me suis rendu compte de la grande diversité des tâches à réaliser pour faire du bon boulot : conception graphique et expérience utilisateur (souvent déléguée à des spécialistes), architecture client et serveur, sécurité, gestion des données, outillage (devops), documentation, aide utilisateur, tutoriels vidéo, et… surtout, les tests et le débugging qui peuvent facilement prendre la tête tellement les problèmes peuvent être complexes à résoudre. Et encore, je ne suis pas éditeur de logiciels ! Ce n'est que du "code de garage".

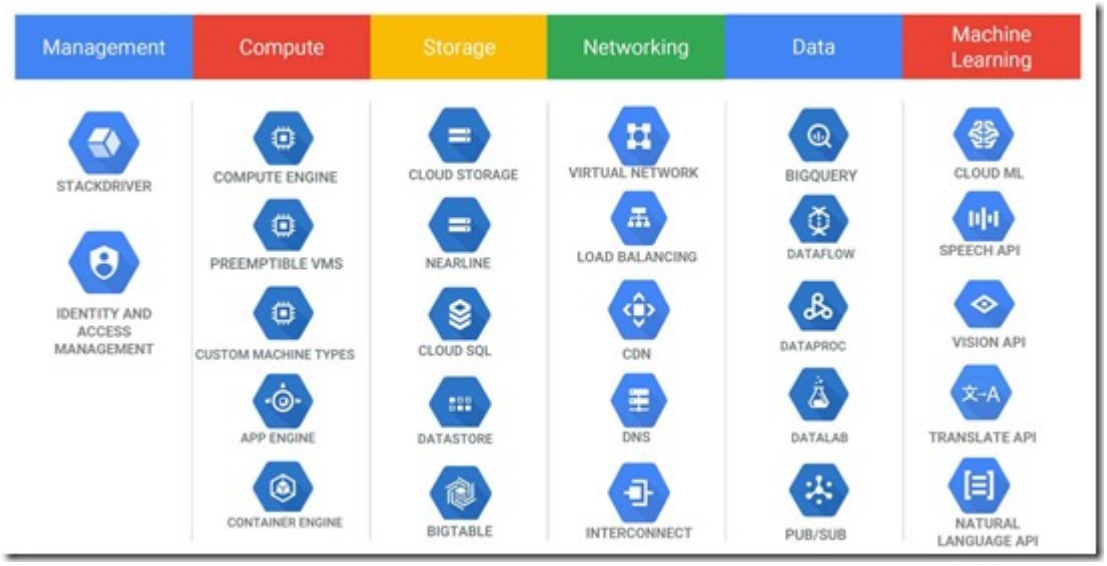
La capacité à résoudre les bugs d'un logiciel est un véritable savoir en soi. C'est une compétence intéressante et réutilisable dans de nombreux domaines. Elle aide par exemple à rechercher les causes d'un problème non pas là où il se manifeste mais dans sa périphérie et ses composantes. Il requiert de se créer une image mentale aussi large que possible du système à corriger.

C'est un outil d'analyse très puissant dans nombre de métiers. On l'appelle aussi "approche systémique". Elle sert par exemple en économie ("comment relancer la croissance", "comment baisser le chômage", "pourquoi ci, pourquoi ça ?"…) et dans le marketing ("pourquoi les clients n'achètent-ils pas notre produit", "pourquoi les médias n'en parlent-ils pas ?", "pourquoi mon concurrent est-il mieux placé ?", …).

Et le cloud dans tout ça ? Là encore, s'il facilite un certain nombre de choses, il les complique aussi et requiert la maîtrise pour le développeur d'un nombre de plus en plus grand de concepts

et outils : machines virtuelles, hyperviseurs, containers, Kubernetes (gestion de containers originaire de Google), devops (pour automatiser la mise en production des logiciels), AIOps (utilisation de l'IA pour automatiser les tâches de la production et de la supervision informatique), etc. Les logiciels font aussi de plus en plus appel à des processeur spécialisés (GPU, neuromorphiques). Le développeur d'aujourd'hui doit donc avoir aussi des compétences en hardware.

Si vous développez par exemple des solutions logicielles sur la plateforme de cloud de Google (*ci-dessous*), vous devez découvrir des dizaines d'outils et de concepts associés. On est plus proche du métier d'architecte que de celui de maçon !



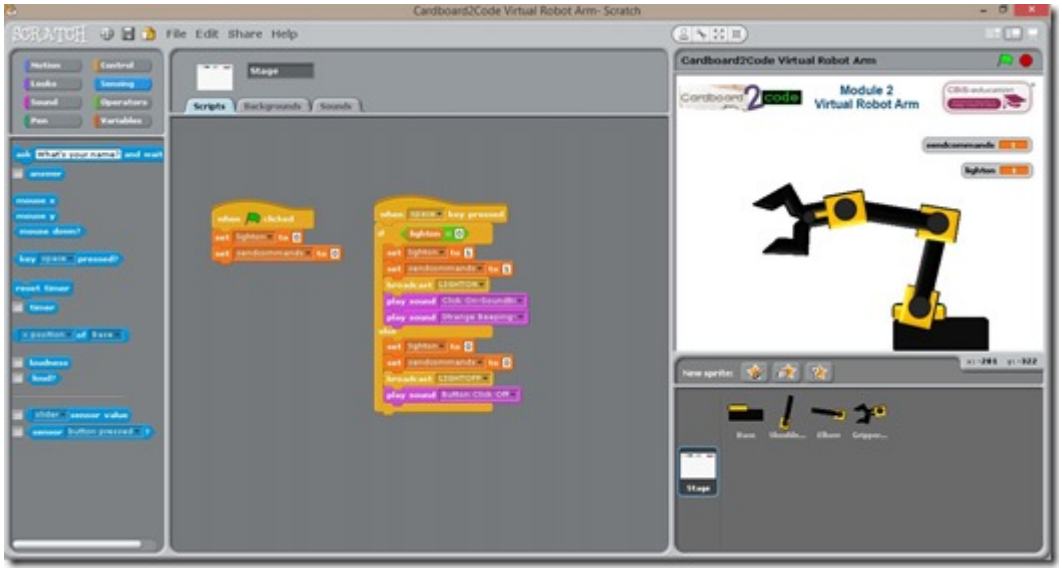
Et les données ? Elles jouent bien entendu un rôle de plus en plus clé, surtout dans les applications de machine learning. Le développeur doit en comprendre les aspects, se (re)plonger dans les basiques des statistiques et probabilités, comprendre aussi les biais des données d'entraînement de systèmes d'IA. Cela ne remplace pas les compétences existantes des développeurs, cela les complète. Même si dans la pratique, les entreprises et les startups ont tendance à bien (trop) séparer les métiers entre data scientists et les développeurs. Encore cette taylorisation !

Bref, développer aujourd'hui est loin d'être un métier monolithique comme certains aimeraient le croire. Le développeur s'appuie maintenant sur une bien plus grande variété d'outils de travail que nombre de spécialistes d'autres métiers (radiologues, experts-comptables, marketeur, …).

Revenons aux enfants

Malgré tout ce que nous venons de voir, je ne suis pas forcément un grand adepte de l'apprentissage de la programmation dès le plus jeune âge. L'enfant en bas âge a surtout besoin de bien se relier au monde physique. D'où l'importance des jeux de construction, pour les garçons comme pour les filles. Ils éveillent au réel, à la créativité, à la construction par modification de modèles préexistants.

C'est dans ce cadre que l'apprentissage de la programmation est pertinente, comme avec le langage Scratch du MIT utilisé en robotique. Elle a du sens lorsqu'elle est reliée au monde réel avant de permettre de gérer des raisonnements totalement abstraits dans un second temps.



Il faut surtout ne pas confondre ces outils de découverte qui permettent de comprendre quelques composantes de l'articulation du monde technique avec les métiers de développeurs. Les premiers sont des outils génériques qui complètent les autres savoirs fondamentaux (calcul, lecture, écriture, dessin, musique, sciences de la vie, ...). Les seconds sont des métiers spécialisés.

L'apprentissage du code ne créé pas directement de compétences métiers. Il permet juste de diversifier sa manière de comprendre le monde. Eventuellement, il donnera envie de s'intéresser à ces métiers tout comme les sciences de la vie peuvent donner envie de devenir médecin.

Et à la mixité...

Pour ce qui est de "*réorienter massivement les filles vers les études de codage*", ce n'est pas une mince affaire puisque presque toutes les tentatives d'améliorer la situation se sont soldées par des échecs des 10 dernières années. On est même plutôt en pleine régression, tout du moins en France avec moins de 10% de femmes dans les écoles spécialisées en informatique.

Comme s'y attachent les associations membres de la fondation "**Femmes@Numérique**", dont "Quelques Femmes du Numérique !" fait partie (j'en suis l'un des cofondateurs avec Marie-Anne Magnac), cela passe par de nombreuses actions de sensibilisation. Il n'y a pas de solution miracle mais une association d'ingrédients à assembler avec discernement.

Il faut commencer par utiliser un vocabulaire valorisant des métiers associés au développement logiciel qui en met en valeur la richesse à la fois technologique et humaine. Cela peut s'illustrer par les objectifs des logiciels et leur rôle dans une incroyable diversité de secteurs d'activité : dans la santé, dans l'agriculture, dans les transports, dans l'environnement, dans le marketing ou même dans les médias. De plus en plus, le développeur a une double compétence : technique et métier. Cela lui ouvre d'ailleurs des portes professionnelles inimaginables en début de carrière.

Comme le rappelle à l'envie Guy Mamou-Mani aussi bien dans son dernier ouvrage anti-anxiogène "**L'apocalypse numérique n'aura pas lieu**" que dans ses interventions en conférences ou sur Twitter, la demande est là ! C'est même la première branche de métiers en terme d'emplois toutes industries confondues. Même si l'attrait économique d'une filière ne doit pas être la seule motivation pour s'y intéresser, il est bon de le souligner, surtout au regard de nombreuses filières d'enseignement supérieur sans débouchés.

Cela passe aussi par la mise en valeur “en ligne” et “IRL” (in real life, sur le terrain) de “role models” et de parcours diversifiés qui sont passés par le développement logiciels où y sont même restés (*ci-dessus*, une cinquantaine de développeuses photographiées dans le cadre de QFDN). Les jeunes procèdent beaucoup dans leurs choix par identification. Ils sont influencés par ce qu’ils reçoivent, notamment via les médias et contenus divers, et par leurs inévitables stéréotypes envahissants. Changer ces contenus prendra du temps.

Enfin, on l’oublie parfois, cela demande aussi des efforts de la part de la gent masculine qui se doit d’être plus accueillante dans ces différents parcours. Et ce n’est pas le moindre des efforts à déployer !

Cet article a été publié le 4 mars 2019 et édité en PDF le 11 mai 2019.