Séminaire Transformations numériques

Séance n°3 du 10 mars 2016

De l'entreprise traditionnelle à l'entreprise logicielle

par

Yves Caseau

Head of Digital and Innovation – AXA Group

Membre de l'Académie des technologies

En bref

Le digital n'est désormais plus le pré carré des bâtisseurs de cathédrales que furent les informaticiens des dernières décennies. Son irruption dans tous les secteurs de la vie, la puissance des réseaux, l'évolution des clients ont fait émerger une nouvelle culture, celle des big data et des *big players*, Google, Apple, Amazon et consorts, qui ont compris les premiers que la rapidité n'était pas incompatible avec l'excellence. Le temps est venu des systèmes jetables, des processus de fabrication automatisés, du *machine learning*, des plateformes et du *cloud*, tous nécessitant un code noble, structuré et élégant. Les nouveaux programmeurs, au sein de *cross-functional teams*, inventent aujourd'hui, sous nos yeux, l'entreprise logicielle.

Compte rendu rédigé par Pascal Lefebvre

Un monde de compétition numérique

Pour les entreprises, la première révolution est d'être désormais en contact permanent avec des clients détenant un pouvoir qui a quitté la force de production pour passer entre leurs mains. Les marchés sont devenus, en quelque sorte, des conversations et, dans ce monde nouveau, les entreprises doivent écouter leurs clients, détecter leurs besoins et les opportunités de les satisfaire.

La deuxième révolution est que les clients sont devenus les architectes de leur propre expérience. On ne cherche plus tant à leur vendre un bien ou un service qu'à produire une expérience dans laquelle ils tiennent le rôle principal.

Une troisième révolution vient s'y ajouter : les entreprises doivent passer de plans stratégiques, qu'il s'agissait d'exécuter, à une forme d'agilité stratégique. Tout l'art du dirigeant va alors être de maintenir le cap dans cet environnement mouvant où le client change constamment d'attentes.

En conséquence, il n'est désormais plus possible d'être un acteur efficace sans avoir une stratégie logicielle devenue le cœur de la production de valeur numérique. Ce que l'entreprise sait faire ou ne pas faire, ce qu'elle achète, comment elle joue avec l'autre, etc., vont déterminer les écosystèmes auxquels elle participe. Son agilité stratégique va donc être déterminée, pour une large part, par ses stratégies logicielles.

Un autre point frappant est que cet effort logiciel a profondément changé en quinze ans. Il y a trente ans, il était noble et passionnant de faire du logiciel, puis sont arrivés l'*outsourcing* du code et des unités d'œuvre réalisées en Inde qui ont, pour un temps, radicalement changé nos façons de faire. Mais nous revenons à notre point de départ, car un code qui change sans arrêt doit désormais être beau, partageable et montrable. Si, comme c'est le cas chez Google, un module sur deux doit être changé chaque mois et que votre code n'est pas noble, structuré et élégant, vous ne serez pas en mesure de devenir un acteur dominant. En d'autres termes, si l'objet logiciel change de façon continue, la façon de faire devient tout aussi importante que ce que l'on construit. Le temps des systèmes pérennes est désormais révolu, mais lorsque l'on est dans des systèmes jetables, tous les processus de fabrication deviennent absolument fondamentaux et s'ils ne sont pas extrêmement automatisés et de très haute qualité, nul ne sera capable de se maintenir dans la course.

L'apprentissage organisationnel permanent

Puisque les objets sont en perpétuel renouvellement, les technologies, les outils et les éléments que l'on assemble le sont également. Pour l'entreprise, la grande ambition de l'apprentissage organisationnel permanent, plus qu'une évidence intellectuelle, est devenue une nécessité impérieuse.

Nous sommes désormais passés de la *black box* à la *white box*, voire à la *transparent box*. Cette nouvelle façon de faire du code implique de le réassembler en permanence. Le code source, que l'on avait longtemps caché, redevient la matière première. On fabrique désormais de grands systèmes avec 90 % de code déjà créé par ailleurs, et de code d'assemblage pour le reste. Malgré les efforts très importants réalisés pour faire de l'assemblage automatique, le monde digital, tel qu'il est aujourd'hui dans la Silicon Valley, est essentiellement constitué de scripts Python et d'assemblages en tous sens. Cela signe donc le retour à des usages et une culture logicielle tels que j'ai personnellement pu les connaître et les pratiquer lors de mes débuts dans ce métier.

L'autre caractéristique du moment est que, puisque l'on fabrique très rapidement des choses qui n'ont qu'une faible durée de vie, il devient fondamental de ne pas avoir les coûts par ligne de programmes que nous connaissions il y a dix ans. Il faut donc chercher à les réduire de façon massive. Les logiciels *open source* vont ainsi faire apparaître des trésors de valeur car on a besoin de logiciels non seulement beaucoup plus nombreux qu'avant, mais aussi qui changent plus souvent et qui doivent être de très grande qualité.

Face à toutes ces contraintes, l'intérêt du logiciel *open source* est qu'il est le seul à être réalisé de façon incrémentale, avec beaucoup de feed-back et de *eyeballs*[1]. La recette de fabrication tient d'une part à ce que vous allez trouver dans le code *open source* et, d'autre part, à ce que vous allez développer vous-même.

Par ailleurs, la seule façon de développer un logiciel évolutif et de grande qualité est qu'il soit soumis à un très grand nombre de testeurs. La productivité, en trente ans, en termes de nombre de lignes ou de défauts (*defects*) par heure, ne progresse que très lentement. Pour l'améliorer, cela impose donc de travailler autrement, c'est-à-dire en mode communautaire.

Un réel jeu culturel

Que font les compagnies de référence, les GAFA[2] ou autres, tels Netflix ou Spotify ? Ce qui frappe en premier lieu, c'est que toutes ont une culture fondée sur les données et les décisions factuelles. On mesure absolument tout afin d'entrer dans une démarche d'amélioration continue, facile à mettre en œuvre dès lors que l'on est dans des systèmes de logiciels que l'on renouvelle sans arrêt. Améliorer une cathédrale de façon continue n'est pas évident mais, avec un objet que l'on remplace tous les mois, cela devient possible.

Parmi tout ce que l'on mesure de façon obsessionnelle, le temps que passe le client à utiliser votre service figure en première place. Apprendre qu'il n'y a rien de plus précieux que le temps de ses clients serait une grande leçon pour les entreprises du CAC 40. Le genre de chose à absolument éviter, c'est de développer un site dans le seul but que des tâches auparavant dévolues au *back office* soient désormais réalisées par le client. Chacun a déjà vécu cette déplaisante expérience où, pour la dixième fois, il doit remplir le même formulaire d'informations pour un même service.

L'autre caractéristique de ce *new way of working* est l'intégration du processus de fabrication des logiciels. Si je passe mon temps à tout changer, les coûts de transaction avec des producteurs externes de logiciel deviennent trop importants. Les grands du web ont donc tous développé une maîtrise remarquable de leur création logicielle.

Le mot *plateforme* est également devenu un incontournable. Ce n'est pas un hasard car la plateforme a deux avantages. C'est un lieu de diversité, ce qui crée de la satisfaction client, mais c'est aussi un lieu qui génère de la valeur économique en maximisant la réutilisation. On pourrait donc considérer que c'est la solution d'une équation permettant de faire plus de choses en moins de temps, avec moins d'efforts. Cela implique alors d'aller enrôler de l'intelligence à l'extérieur, pour des raisons tant économiques que d'innovation. L'extraordinaire course des géants du web, d'une part sur les méthodes de l'intelligence artificielle et, d'autre part, sur l'exposition de ce qu'ils font, montre qu'ils sont bien conscients qu'à vouloir tout faire tout seuls, ils risquent de mauvaises surprises venant de la communauté ouverte.

Dans les années 2000, les grandes compagnies de télécommunications, comme Orange ou Verizon, avaient déjà la même tendance à développer des plateformes. Toutes ont eu la volonté d'en créer et d'y faire venir les développeurs, mais c'en sont d'autres qui ont marché, construites par des gens qui avaient une réelle culture de l'*open source*. Aujourd'hui, le challenge des entreprises françaises est d'internaliser cette culture, car la meilleure façon de réaliser un logiciel qui va donner l'envie à d'autres développeurs de venir jouer avec, c'est, d'abord, d'avoir soi-même passé beaucoup de temps sur la plateforme des autres. C'est un réel jeu culturel, du *give and take*, qui suppose de commencer par contribuer au travail des autres et d'abandonner la vision égocentrique si naturelle chez les acteurs dominants.

1. « *Beaucoup d'yeux qui voient le même code* » : l'*open source* conduit à une "revue de code" généralisée, ce qui apporte qualité et simplicité.
2. GAFA désigne les actuels géants du Net américains : Google, Apple, Facebook, Amazon.

Fabriquer des expériences

Mon rôle chez AXA est de fabriquer des expériences et de la satisfaction client. L'intérêt d'une expérience, c'est de remettre le client au centre mais, surtout, de lui faire vivre toute la subjectivité d'une situation particulière réunissant un ensemble de produits et de services. L'expérience est donc la combinaison d'un service et de la prise en compte de tout un contexte.

Par exemple, dans le contexte de vieillissement de la population, nous nous intéressons aux chutes non détectées des seniors de 80 ans et plus. À cette fin, certains cherchent à mettre en place des capteurs. Quant à nous, nous cherchons à le faire de façon peu intrusive par le biais des big data. Nous avons développé une application mobile sur smartphone qui réalise un certain nombre d'observations, certes imparfaites, mais qui permettent globalement de détecter une possibilité de situation inquiétante. Nous nous intéressons, en particulier, à toute la puissance des capteurs des smartphones. Leur GPS donne la localisation et leur accéléromètre calcule la vitesse de déplacement dans son logement. On sait que, pour une personne de plus de 80 ans, cet indicateur est fiable et prédictif, quelques mois à l'avance, de dépression ou de l'apparition de troubles fonctionnels. Quand on connaît cette vitesse moyenne de déplacement, on est capable de déceler une immobilisation ou une baisse d'activité anormale, justifiant une éventuelle intervention de l'entourage. Cela suppose évidemment que le senior ait un smartphone et le garde sur lui, souhait émanant généralement de la famille mais pas toujours accepté par la personne, ce qui limite malheureusement la portée de cette expérimentation !

Chez AXA, dans ce cadre, nous travaillons selon une recette classique de l'Internet, en combinant des experts en marketing produits, des développeurs et des designers. Le design joue, en effet, un rôle essentiel pour réduire les frictions lors de l'utilisation et accroître la partie ludique de l'expérience.

Do you write code ? Do you Google it ?

Le monde de la programmation a changé radicalement. Si je pose un problème algorithmique quelconque à un programmeur de ma génération, il va, sur le champ, se mettre au clavier et commencer à coder. Si je pose ce même problème à un jeune programmeur, il va immédiatement aller sur Google pour trouver le bout de code qui va résoudre le problème. Le message que j'essaie de faire passer aux grandes entreprises est qu'il leur faut trouver le bon équilibre entre les deux méthodes. La façon de programmer des jeunes est très différente et, aujourd'hui, ils font de l'assemblage, ce qui est une toute autre culture, même si dans la Silicon Valley, on continue à employer aussi des programmeurs aux cheveux gris ! Cette culture permet de rentrer dans le monde du massivement distribué. Les jeunes programmeurs qui sont imprégnés de cette culture et ont acquis les bons réflexes sont alors capables de faire rapidement des choses assez extraordinaires en *massive open online*.

Quand je suis arrivé chez AXA, j'y ai trouvé l'équivalent d'une plateforme, un peu sophistiquée, que j'avais conçue quelques années auparavant, pour une entreprise de télécommunication. Dans cette entreprise, une équipe de dix personnes y avait travaillé pendant un an. Chez AXA, cela n'avait nécessité que trois personnes pendant trois mois, de la conception à l'intégration, et ils avaient réalisé une véritable Formule 1 ! On ne peut plus se permettre d'ignorer ce monde-là, tant il est désormais facile de trouver, en *open source*, d'innombrables algorithmes, franchement astucieux et intelligents, prêts à être intégrés pour peu que l'on ait un peu de talent.

Le big data peut aussi être vu comme une des raisons du changement des façons de coder. Au sein de l'Académie de technologie, nous organisons des conférences où nous faisons venir des gens aussi remarquables que Thomas Hofmann, professeur à l'École polytechnique fédérale de Zurich, qui a développé les algorithmes de AdSense pour Google, ou Jeffrey Hammerbacher, le premier "Monsieur big data" de Facebook. Leur message est que le big data ne se limite pas à la seule opportunité de créer de nouveaux services, mais que c'est la possibilité de refaire des choses classiques en s'appuyant sur une disruption très forte des coûts technologiques. Il est frappant de voir que les entreprises qui maîtrisent cette nouvelle panoplie d'outils sont non seulement capables de créer des choses nouvelles, mais aussi de reconstruire des anciens systèmes transactionnels, avec des performances et des coûts très différents.

Désormais, quand on construit un système à partir de larges bases de données, on le fait différemment, en combinant des éléments très simples et très paramétriques, et en utilisant beaucoup de *machine learning*[3]. Il est intéressant de remarquer que les deux systèmes les plus extraordinaires de *machine learning* du point de vue de la création de valeur, AdSense, de Google, et Criteo, sont de la régression logistique relativement classique mais basée sur des millions de paramètres et un savoir-faire de mise en œuvre qui, lui, représente des années de travail et est difficilement reproductible. Ce qui est en train de se passer avec ce *new way of programming*, c'est que l'on construit des systèmes avec un nouveau savoir-faire et une nouvelle culture logicielle. Il nous faut désormais l'acquérir, tous les géants du Net étant, sur ce point et chacun dans leur domaine, très en avance.

L'usine logicielle

La métaphore de l'usine est utile pour rendre compte du besoin d'automatisation, de configuration et de rigoureuse synchronisation de l'exécution. Elle fait contrepoids à l'idée qu'agilité rime avec esprit libre et absence de contraintes. Dans le monde actuel, on fabrique du code qui change sans arrêt et, en conséquence, si vous n'êtes pas à l'état de l'art en automatisation, vous n'avez aucune chance d'exister. Les méthodes d'il y a dix ans sont totalement dépassées dès lors qu'il faut faire tourner la machine tous les jours. Quand, pour de multiples raisons, on a eu besoin d'intégrer dix millions de lignes de code par jour, on s'est rendu compte que l'on ne savait absolument pas le faire. Pour y parvenir, le jour, et réaliser les tests, la nuit, cela demande donc un réel effort mais, quand on y parvient, on obtient des résultats extraordinaires. Cela implique donc que, si le *process* doit être modifié en permanence, il doive être entièrement automatisé afin que la fabrication, l'intégration et le déploiement se fassent de façon continue. L'une des stars au sein du monde des opérations de chez Google, interrogé sur la façon dont il avait procédé, a répondu que s'il ne donnait à ses développeurs que des opérations, ils s'ennuieraient tellement qu'ils automatiseraient autant que possible, ce qui s'est effectivement avéré.

L'approche DevOps[4] est une école de pensée qui explique cela très bien et à laquelle je me réfère souvent. Elle prône à la fois l'idée de traiter toute son infrastructure comme du code et de donner à sa machine de production physique le même niveau de souplesse et de contrôle d'utilisation qu'un logiciel.

La deuxième caractéristique d'une organisation moderne est de travailler en équipes fonctionnelles croisées (*cross-functional teams*) autonomes avec un certain nombre de rituels dont le fameux *stand-up meeting*, avec l'idée que, pour fabriquer ces produits logiciels, il faut surtout des équipes qui vivent ensemble. Le livre d'Eric Schmidt et Jonathan Rosenberg, *How Google Works*[5], montre bien que, pour faire du logiciel, les gens doivent « *vivre, manger et respirer ensemble* ». L'utopie selon laquelle chacun travaille seul et que la coordination se fait à distance marchait pour la réalisation des grandes cathédrales techniques mais plus pour des produits grand public, où l'on fonctionne en cycles itératifs.

Dans l'esprit du *lean management*, on pratique le *Kaizen*, c'est-à-dire la résolution de problèmes en équipe comme outil de formation collective. On a également importé un certain nombre de techniques du *lean management* pour faire travailler les gens en flux tiré (*pull architecture*). Ce qui est frappant, avec les méthodes classiques, c'est que les gens passent leur temps à s'attendre. Dans une organisation logicielle moderne, on cherche à avoir des flux continus afin d'éviter ces ruptures de charge. Le management visuel, qui repose sur un ensemble d'affichages d'indicateurs visuels, quantitatifs et qualitatifs situés à des endroits stratégiques de la zone de travail, constamment mis à jour par les collaborateurs, est aussi largement utilisé dans cette perspective, de même que l'organisation de hackathons[6].

3. L'apprentissage automatique ou apprentissage statistique (*machine learning*), champ d'étude de l'intelligence artificielle, concerne la conception, l'analyse, le développement et l'implémentation de méthodes permettant à une machine (au sens large) d'évoluer par un processus systématique, et ainsi de remplir des tâches difficiles ou impossibles à remplir par des moyens algorithmiques plus classiques (source : Wikipedia).
4. https://en.wikipedia.org/wiki/DevOps
5. Eric Schmidt et Jonathan Rosenberg, avec Alan Eagle, *How Google Works*, Grand Central Publishing, 2014.
6. Un hackathon est un événement où des développeurs se réunissent pour faire de la programmation informatique collaborative, sur plusieurs jours. Le terme est un mot-valise constitué de *hack* et marathon.

La culture nécessaire pour jouer selon ces nouvelles règles, passe par l'amour du code (*Love your code*). Lorsque l'on y passe le plus clair de son temps, pendant des jours, on ne peut plus se permettre de le distancer comme on le faisait il y a vingt ans. Cela passe également par la reconnaissance et la valorisation des équipes de développement. Le *Love your code* est d'autant plus important qu'il est impossible de faire du *pair programming*[7] ou des *code reviews*, qui ne sont efficaces que s'ils sont fun, si l'on n'a pas une culture d'appréciation de l'objet sur lequel on travaille.

Si l'on veut faire vivre son code, l'appréciation et l'élégance ne suffisent pas. Il faut aussi des choses plus classiques, comme des standards de code et des disciplines. Quand on visite de vraies entreprises logicielles, il est frappant de voir à quel point elles ont su créer cette fierté et ce souci de l'élégance du code.

En troisième lieu, puisque l'on est dans ce monde agile et incrémental où l'on avance à petits pas, cela produit nécessairement des accumulations. C'est un principe systémique qui veut que, si l'on abandonne complètement les notions d'architecture et que l'on ne produit que par petites étapes, on génère du *junk* (déchet). Une approche incrémentale nécessite donc du "jardinage", du *refactoring*. Si l'on fait beaucoup d'agile, il faut nécessairement, de temps à autre, s'arrêter, réfléchir et nettoyer son code. C'est là également une caractéristique de toutes les grandes entreprises logicielles, dont Spotify est un bon exemple.

Les deux grandes références que j'utilise depuis deux ans sont *The Lean Startup*, d'Eric Ries[8], et DevOps. Ce sont deux métaprocessus composés de deux flèches, l'une allant du client vers le code et l'autre du code vers le client.

La flèche qui va du client vers le code est celle qui cherche à coconstruire avec lui quelque chose d'utile. Le grand slogan du *Lean Startup* est : « *Life is too short to make things people don't use.* » J'ai personnellement passé dix ans chez Bouygues Telecom à créer des services que trop peu de clients ont utilisés ! Désormais, chez AXA, nous faisons du *Lean Startup*, du *Design Thinking* avec les clients pour comprendre leurs attentes, puis nous réalisons un *Minimum Viable Product*[9] que l'on amène, le plus rapidement possible, entre les mains du client pour qu'il puisse repérer ce qui marche ou pas. Ensuite seulement, quand le produit a acquis un minimum de légitimité, nous faisons du *Growth Hacking*[10]. C'est une démarche qui part du haut et du client pour produire un artefact digital. Il faut ensuite fabriquer vite et bien, en processus continu et avec un bon niveau de qualité. C'est la tâche de l'usine logicielle, avec ses équipes agiles et les approches de DevOps.

Conclusion

Je m'inscris complètement dans les pas de Marc Andreessen qui déclare : « *software is eating the world.* »[11] Dans ce monde, pour être un acteur majeur du numérique, il faut donc devenir une *software factory*, une entreprise logicielle, ce qui pose un certain nombre de questions. En premier lieu, il faut apprendre à construire son propre logiciel et se dire qu'il y a quelques domaines de l'activité numérique dans lesquels on devra rapidement être au fait de l'état de l'art extrêmement fragilisé. Pour apprécier l'ampleur de cette transformation culturelle, je recommande chaudement la lecture de *Joy inc.*, l'ouvrage de Richard Sheridan[12].

Si je vous ai convaincu de faire ce grand pas vers l'entreprise logicielle, la bonne nouvelle est que nous ne sommes pas dans le vide et qu'il existe un certain nombre de références pertinentes, dont *The Lean Startup*, le bestseller d'Eric Ries déjà cité, qui est un excellent guide pour les entreprises souhaitant fabriquer de l'innovation digitale

7. La programmation en binôme (*pair programming*), parfois appelée programmation par paires ou binômage, est une méthode de travail dans laquelle deux développeurs travaillent ensemble sur un même poste de travail.
8. Eric Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, Pearson, 2011.
9. Le produit minimum viable (*Minimum Viable Product*, MVP) est une stratégie de développement de produit, utilisée pour de rapides et quantitatifs tests de mise sur le marché d'un produit ou d'une fonctionnalité.
10. Ensemble de techniques de marketing permettant d'accélérer rapidement et significativement la croissance d'une start-up.
11. http://genius.com/Marc-andreessen-why-software-is-eating-the-world-annotated
12. Richard Sheridan, *Joy, Inc.: How We Built a Workplace People Love*, Penguin, 2013.

pertinente. Avec cet ouvrage, le lecteur doit avoir l'humilité de se dire, qu'en dépit de toute son intelligence, ses intuitions sur ce qui doit être le bon service pour le client sont souvent fausses. Avec le *Lean Startup*, je fais des hypothèses (*assumptions*) sur ce que les clients vont aimer ou pas et je construis mon innovation par un *process* itératif qui va chercher à les valider ou pas. Cela demande de l'humilité.

Nous sommes dans un monde d'accélération exponentielle des technologies. Ce qui est passionnant, c'est que cela implique, de façon absolument vitale, une nouvelle façon de travailler pour absorber et mettre en valeur cette avalanche de technologies nouvelles. J'ai la profonde conviction que, si l'on n'applique pas ce nouveau mode d'organisation, nous serons très rapidement dépassés par les entreprises qui l'auront fait avant nous.

Débat

Une amplification communautaire des bonnes idées

Un intervenant : *Au début de votre intervention vous avez dit qu'un code doit être beau et montrable, mais beau pour qui et montrable à qui ? Qui en est juge ?*

Yves Caseau : C'est la résultante d'un jugement communautaire des développeurs. Cela ne passe ni par des règles formelles, ni par des critères esthétiques. Lorsque quelqu'un montre son travail, les autres viennent-ils juste voir par-dessus son épaule ou, au contraire, est-ce que tout le monde suit son avancée ? Le but est d'obtenir une amplification communautaire des bonnes idées. Ce que l'on cherche à faire, c'est de créer une sorte d'*open source* interne. AXA est présent dans 55 pays avec plus de 100 000 personnes et ce que nous voulons obtenir, depuis trente ans, c'est ce partage. Désormais, nous ne souhaitons plus le faire en partageant des librairies ou des systèmes, mais en partageant du code. Pour cela, le fait que beaucoup de gens, ou pas, viennent regarder ce que je fais, quand je m'installe devant ma machine, est un signe qui ne trompe pas.

Int. : *Comment, chez AXA, accueille-t-on ces nouveaux programmeurs qui fonctionnent de manière si différente mais qu'il faut pourtant savoir intégrer et valoriser ?*

Y. C. : L'objectif est de favoriser la collaboration entre les nouveaux et les anciens. Quand on parle de big data, on pense que l'on entre dans un monde où il faut, à tout prix, rechercher des surdoués qui sachent tout faire. En fait, ils sont assez faciles à trouver car, si vous confiez à un jeune ingénieur un cluster Hadoop[13] et que vous le laissez faire à sa façon, vous obtiendrez très vite des choses remarquables. Les entreprises qui décident de faire confiance à leurs employés et savent créer les bonnes conditions de travail ont toujours de bonnes surprises, car nos jeunes ingénieurs sont très bien formés et sont parfaitement capables de faire de l'intelligence artificielle, du big data, etc.

Nous avons également mené avec succès des reconversions de programmeurs Cobol en programmeurs iOS[14] et ils font désormais des applications mobiles de très grande qualité. Tout se passe bien si, dans la culture du management, les informaticiens qui pendant vingt ans n'ont été que des exécutants sont désormais reconnus à leur juste valeur. Les compétences sont là, ce que nous n'avons pas, c'est cette confiance dans nos propres équipes permettant aux gens d'apprendre, même à 45 ans.

13. Un cluster Hadoop est un type particulier de traitement informatique en grappe, conçu spécialement pour stocker et analyser de grandes quantités de données non structurées dans un environnement distribué.

14. iOS est le système d'exploitation mobile développé par Apple pour plusieurs de ses appareils.

Une dualité permanente

Int. : *Où sont les invariants dans cet univers qui change en permanence ?*

Y. C. : Nous sommes dans une dualité permanente. On ne peut avoir de réactions rapides que si l'on a établi de bonnes stratégies sur le long terme. Pour moi, ce qui est fascinant, c'est que la première condition en est la compétence des équipes. Cela prend du temps. Quelles que soient les situations dans lesquelles on vous plonge, surtout si elles sont un peu complexes, même avec un chéquier grand ouvert, vous ne pourrez rien faire si la "machine à redonner confiance" ne marche pas. Si vous la faite tourner, au bout d'un moment, vous aurez une équipe de gens compétents qui abattront du bon travail. Notre monde semble extrêmement fébrile, mais ce qui compte, dans les entreprises qui s'épanouissent dans cet environnement, c'est qu'elles ont construit sur un temps long une capacité à réagir sur un temps court. Les entreprises schizophrènes sont celles qui cherchent, en permanence, à s'agiter à la vitesse du temps court, ce qui ne produit rien d'autre que de la frustration.

Nous fabriquons des écosystèmes qui sont des cathédrales. Les invariants peuvent être architecturaux : ce sont les grandes idées directrices, telles que celles posées par Amazon, il y a cinq ans, et qui sont des principes remarquables. Ils peuvent aussi être culturels. Faire vivre un écosystème, que ce soit chez Apple ou chez Android qui sont des réussites sur ce plan, c'est avoir des invariants de cet ordre qui guident leurs développeurs.

Mais ce nouveau monde est aussi remarquablement compatible avec les critères de jugement de l'ancien, tel celui du temps annuel d'indisponibilité du service pour le client qui coûte toujours très cher. Je ne suis certes pas de la génération actuelle de programmeurs mais, avec mon expérience, je peux cependant toujours les évaluer avec des critères qui, eux, n'ont pas varié. Cela me donne également une vraie légitimité vis-à-vis de mes collègues directeurs des systèmes d'information (DSI) quand je leur dis qu'il n'est plus possible de travailler comme avant alors que les clients ont changé. Cette nouvelle façon de faire de l'informatique concerne absolument tout le monde, y compris au cœur de métier des entreprises.

Int. : *Faut-il être jeune pour être un virtuose ?*

Y. C. : C'est souvent le cas. Mais, en général, un virtuose n'est pas synonyme de la satisfaction du client. Il y a plusieurs raisons à cela. Pour faire une application mobile intéressante, il faut gérer une plateforme, le *cloud*, etc., et l'objet physique que l'on tient en main est l'aboutissement de tout un système. Les gens qui ont construit de tels systèmes mais qui ont échoué, pour des questions de plateformes, de sécurité ou autres, ont un avantage sur les jeunes de 25 ans qui sont parfois un peu naïfs. C'est pour cela que, lorsque l'on fait des services digitaux à grande échelle, on a des équipes constituées d'une diversité de talents. Il suffit ensuite de voir les compétitions de code pour constater que, sur ce terrain, ce sont indiscutablement les jeunes qui gagnent.

Int. : *A priori, tout cela est très technique, mais les conséquences en matière de stratégie sont redoutables. Vous nous dites que l'externalisation, la CRM (*Customer Relationship Management*), les ERP (*Enterprise Resource Planning*) et toutes ces choses auxquelles nous avons cru, sont des erreurs. Comment dit-on cela dans une entreprise ? Passe-t-on sous silence ces errements qui nous ont coûté des sommes pharamineuses ?*

Y. C. : Je suis plus nuancé quand j'évoque ce passé. J'ai été élevé dans la culture des grands systèmes et de l'abstraction, et je ne jette pas le bébé avec l'eau du bain. Il y a des domaines pour lesquels un esprit formel, un cahier des charges, des plans de tests à l'ancienne sont toujours nécessaires. Si je devais faire un grand système pour une centrale nucléaire, je pense que c'est encore ainsi que je procéderais. En revanche, il est évident que l'on a abusé de ces méthodes au détriment des utilisateurs, que ce soient les clients finaux ou les malheureuses entreprises à qui l'on a imposé des systèmes absolument inhumains, bridant la créativité et l'efficacité des salariés.

Comment le dire alors ? Selon moi, avec beaucoup de modestie. Chez AXA, j'essaye de faire avancer les choses et j'y bénéficie d'une certaine liberté pour mettre en œuvre ce que je viens d'évoquer : je dispose d'un petit budget, qui serait néanmoins conséquent au regard des moyens d'une entreprise normale, pour fabriquer, selon les principes du *Lean Startup*, des innovations digitales. Le juge de paix est alors la satisfaction des clients. Mon rôle est de démontrer que, lorsque l'on invite le client dans le processus de développement, on crée de la valeur supplémentaire, sauf si cette expérimentation tourne court. En revanche, quand ça marche, je peux dire que

le périmètre ne s'arrête pas aux trois ou quatre applications que nous développons, mais que c'est bien le cœur du métier d'assureur qui est profondément digital. L'impact est alors plus fort et mon discours n'est que modérément disruptif.

Quels talents pour l'entreprise logicielle ?

Int. : *Selon vous, une entreprise qui veut devenir un* big player *dans le monde du numérique doit donc devenir une entreprise logicielle. Si l'on pousse ce raisonnement, cela veut-il dire que, d'ici peu, toute entreprise qui voudra devenir un* big player*, quel que soit son métier, devra savoir faire du logiciel ? N'est-ce pas alors dès le collège qu'il faudrait commencer à se poser la question des aptitudes et compétences nécessaires qui devront être développées chez les futurs programmeurs, sauf à prendre un retard irrattrapable en cas d'erreur ?*

Y. C. : La Finlande a réalisé un très bon document sur cette question. Pour que dans dix ans nous puissions avoir des équipes fonctionnelles croisées efficaces, il faut qu'un ensemble de compétences soit disponible autour du design du *system engineering*, des écosystèmes, etc. Cela dit, hormis quelques secteurs très en pointe, le niveau d'excellence nécessaire pour faire quelque chose d'utile dans l'axe design, du big data ou du *system engineering*, n'est pas si élevé. Un des domaines qui cristallise ces problèmes, c'est celui de la *smart assistant*, de l'intelligence artificielle ou du contextuel. Que l'on soit chez AXA ou chez Renault, on devient une sorte d'ange gardien du client, on écoute les signaux faibles et au bon moment, on lui propose notre produit. Quand on voit les moyens et les technologies de Google, on se dit qu'il va monopoliser ce domaine. Mais, en fait, le nombre d'opportunités dans cet univers-là est tellement immense que sans pour autant avoir une armée de PhD à sa disposition, on est capable de faire des choses tout-à-fait intéressantes. Dans l'équation "compétences acquises par l'éducation" contre "*empowerment* par la mise en place de bonnes conditions de création", 80 % des résultats dépendent de la deuxième proposition, par ailleurs beaucoup plus facile à mettre en place. Maintenant, pour aller affronter les GAFA et faire des logiciels encore meilleurs que les leurs, il faudra effectivement des gens particulièrement bons, mais ce sont alors des compétiteurs exceptionnels.

Int. : *Est-ce si facile, pour les DSI et les informaticiens qui vivent depuis longtemps dans un système très formaté, de passer dans cet autre monde ?*

Y. C. : C'est tout sauf facile ! Pour innover, selon Eric Ries, ce ne sont pas des équipes d'informaticiens que l'on crée mais des *cross-functional teams* dans lesquelles on va insérer des informaticiens. L'autonomie n'est pas, en soi, une recette du succès. Ensuite, dans mon expérience récente, les programmeurs Cobol qui sont devenus des développeurs iOS sont des exceptions. La règle est plutôt la difficulté, car le schéma hiérarchique classique « *J'ai un niveau de compétences où je suis le meilleur, personne ne doit donc venir m'embêter* » est extrêmement pesant. Je suis frappé par les difficultés que je rencontre pour mettre en œuvre les schémas de *Lean Startup* et de *software factory* avec des gens qui ont fonctionné pendant des années sur un mode client/fournisseur et qui sont très frustrés quand on leur annonce que les règles vont changer.

Int. : *Les gens acceptent-ils ces concepts ?*

Y. C. : Deux écueils sont fondamentaux. Le premier est profondément conceptuel, avec l'idée que tout peut se comprendre, qui résulte de notre éducation. Le second est de vouloir comprendre avant de faire. Ce que l'on cherche à développer dans les entreprises, ce sont des compétences que l'on acquiert dans l'action. Dans d'autres cultures que la nôtre, cela ne pose pas de problèmes. On ne demande pas aux gens d'adhérer à une philosophie, mais de participer à un jeu, qui ne leur prendra pas plus de 5 % de leur temps, et dont on leur promet qu'il sera amusant. Ce genre de discours passe fort bien aux États-Unis et dans nombre de pays, mais très mal en France.

La devise « *Pour commencer à innover, il suffit que les HiPPOs*[15] *se taisent et qu'ils écoutent le client* » ne fait pas rire tout le monde chez nous.

Int. : *Comment incitez-vous le client à vous dévoiler son expérience ? N'y a-t-il pas incompatibilité entre la rapidité que vous souhaitez et le temps juridique ?*

Y. C. : Chez AXA, nous avons le *customer feedback learning loop*, la boucle d'apprentissage à partir des retours clients, et nous utilisons trois axes. Le premier, c'est tout ce qui est analytique et implicite : grâce au monde des technologies numériques il est possible de mettre des sondes et des *trackers* absolument partout, ce qui permet de faire remonter des informations sans effort pour le client, à ceci près que, pour des raisons juridiques, cela doit se faire avec son consentement. Si l'on joue ce jeu de manière transparente et honnête, on peut appréhender ainsi une part de son expérience.

La deuxième dimension concerne l'explicite. On donne alors la parole au client. C'est tout un art car les deux moments où le client a vraiment envie de prendre la parole, c'est quand il est soit vraiment très content, soit vraiment très mécontent. Entre les deux, il a plutôt envie d'être laissé tranquille. Il faut alors lui donner les outils pour qu'il puisse le faire de manière efficace. Néanmoins, nous sommes dans une culture où les clients ont envie de s'exprimer et, de façon générale, ils le font plutôt de bon gré.

Enfin, il faut rendre cette expression communautaire. Si la liste des mécontentements d'un client trouve une tribune au sein des équipes et que ses remarques peuvent rebondir entre leurs membres, la qualité des réponses s'enrichit d'un facteur 10. Si, au lieu de parler à des individus, on parle à des communautés, on amplifie cette communication et on enrichit de manière spectaculaire le dialogue entre l'utilisateur et l'entreprise. C'est un art qui est désormais devenu une pratique digitale chez tous les *best in class*.

15. Dans ce contexte, abréviation de *Highest paid person's opinion*, opinion de la personne la mieux payée.

Présentation de l'orateur

Yves Caseau : est le directeur digital et innovation du groupe AXA ; il développe des applications numériques innovantes pour l'ensemble des entités du Groupe; il est également en charge de la coordination informatique du domaine numérique, auprès du DSI Groupe et de l'animation de la transformation numérique et de l'innovation dans le Groupe ; il a été le directeur général adjoint technologies, services et innovation de Bouygues Telecom de 2007 à 2013 ; il est membre de l'Académie des technologies et auteur de trois livres chez Dunod.

Diffusion juin 2016