Source : ***[The value of software - Planet Lean (planet-lean.com)](#)***

La valeur du logiciel

Michael Ballé, Céleste Dhellemmes et Woody Rousseau, expliquent pourquoi il est clé de mettre la valeur au cœur des discussions avec le client pour réussir le développement d'une entreprise prospère et robuste dans le domaine de la tech.

56 % de croissance d'une année à l'autre, c'est fantastique – une bénédiction même. Atteindre de tels résultats est enivrant, mais cela vient également avec son lot de tracas : où trouver des développeurs ? Une fois que nous les avons convaincus de nous rejoindre, comment allons-nous trouver des projets sympas pour maintenir leur intérêt – et les garder ? Comment allons-nous créer un environnement convivial en des temps si difficiles ? Dans la frénésie du quotidien, il est aisé de se retrouver embourbé dans les problèmes sans fin de ventes ou d'opérations et de passer en mode réactif.

Avec mon co-fondateur Rodolphe Darves-Bornoz, nous sommes tout à fait conscients que nous avons de la chance de nous trouver sur un marché où la demande surpasse largement l'offre, et d'avoir choisi le marché de la technologie financière (FinTech) pour notre start-up Tech. Il y a plein de boulot et de projets fascinants. Nous sommes très fiers d'avoir aidé BpiFrance, la banque d'investissement française, à promouvoir l'innovation et la finance auprès des entrepreneurs, pour répondre à la pandémie du Covid-19, et à adapter leur offre auprès des entreprises en difficulté dans ces temps de grande nécessité. Mais nous avons réalisé à quel point il était simple de passer en mode « livrer à tout prix » pour satisfaire nos clients et passer à autre chose.

Mesurer notre taux de recommandation (NPS) tout au long de nos projets fait partie intégrante de l'ADN de notre entreprise. Nous avons donc une bonne perception de l'état d'avancement du projet et savons si les clients sont satisfaits de nos équipes. Nous savons quand nous faisons les choses correctement ou pas. Mais le NPS ne nous dit pas si nous faisons les bonnes choses ! Il y a quelques années, nous avons commencé à nous questionner sur la qualité des systèmes que nous fabriquons – la qualité fondamentale, pas seulement « le moins de bugs possibles et livrer à temps ». Nous avions besoin de nous construire une réputation de créateurs de logiciels intelligents afin de créer une

entreprise prospère et ainsi devenir incontournables pour l'acquisition de systèmes utiles et élégants. C'est ce qui nous a amenés à explorer le Lean.

L'une d'entre nous (Céleste) a rejoint l'Académie Lean Ingénierie (animée par Michael) et a commencé à se demander comment les concepts du Lean en Ingénierie pourraient s'appliquer à la production de logiciels. En commençant par les outils de QFD (Quality Function Deployment), elle s'est rendu compte que l'architecture des produits pourrait également s'appliquer aux systèmes de logiciels : développeurs et ingénieurs pensent de la même façon quand il s'agit de livrer des fonctionnalités aux clients. Céleste a séparé un produit en 3 parties : les tâches à faire, les systèmes, et les technologies.

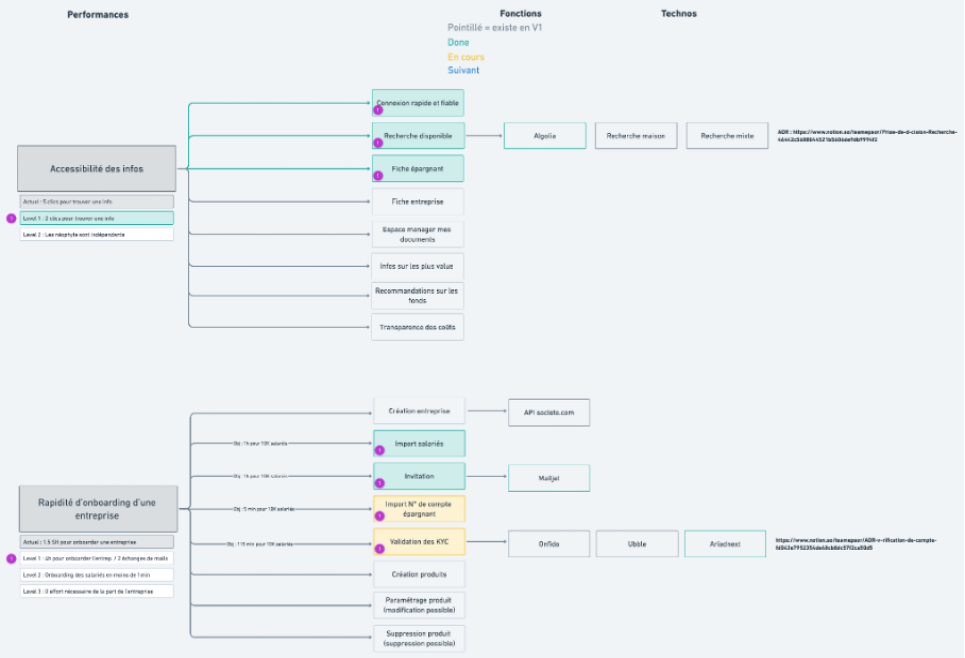
- ***Les tâches à faire*** définissent les niveaux de performance dont les clients ont besoin, comme la facilité de connexion, la vitesse à laquelle le système répond, et les niveaux de défaillances ;
- ***Les systèmes*** sont les pavés de code qui réalisent les tâches demandées à travers l'interface utilisateur ;
- ***Les technologies*** consistent à choisir la bonne technologie pour coder le système le plus intelligemment possible. On s'intéresse alors à la sécurité (vulnérabilité au piratage), à la vitesse (le temps de réponse), et la stabilité (les risques de pannes ou de surcharge d'autres systèmes par des interfaces lourdes).

Céleste a représenté sur un diagramme « système » ces 3 aspects de nos produits, en s'essayant dans un premier temps sur des produits existants, sur lesquels nous avions suffisamment de recul pour comprendre nos choix techniques, bons ou mauvais. Elle travaille désormais avec les architectes pour les convaincre de clarifier leurs choix au fur et à mesure qu'ils conçoivent de nouveaux produits ou fonctionnalités pour les clients.

L'observation de ces diagrammes d'architecture nous a donné un regard unique sur notre performance – nous savons mesurer si nos produits finaux sont bons, moyens ou faibles, en nous basant sur les « tâches à faire » une fois qu'ils sont entre les mains de l'utilisateur.

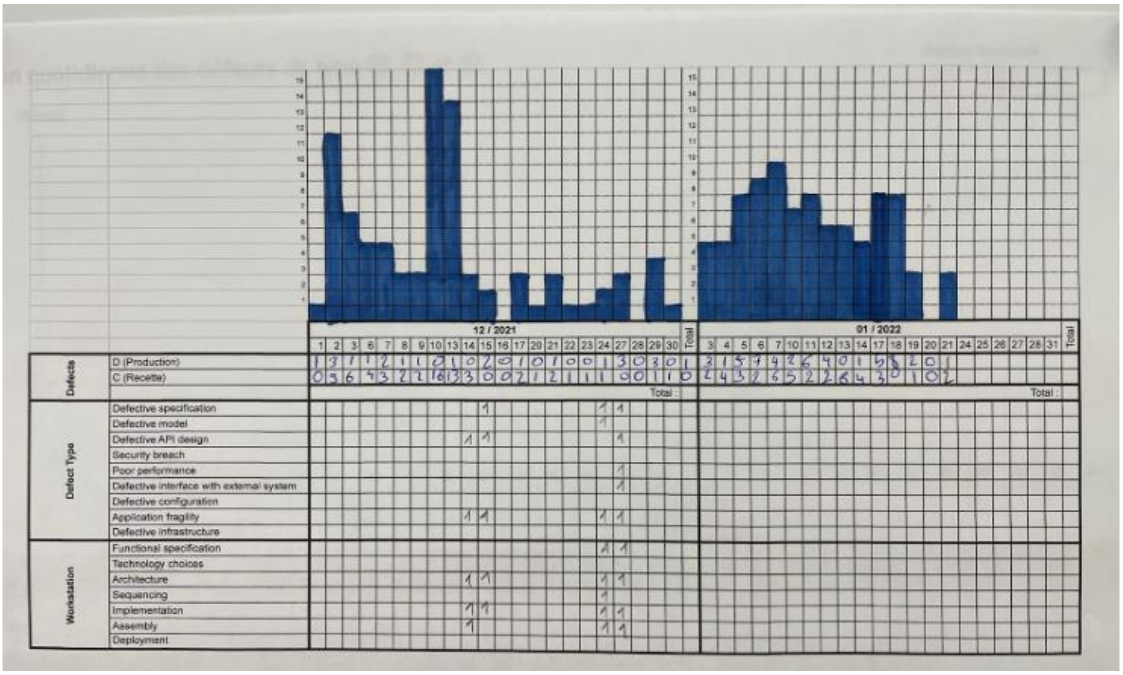
Mais cela nous a aussi amenés à nous poser une question plus profonde sur la valeur du point de vue du client : avons-nous réussi à bien identifier les « tâches à faire » ? Comme nous le savons tous, les distributeurs de logiciels continuent d'ajouter des fonctionnalités à leurs systèmes pour satisfaire tel ou tel soi-disant « besoin client » jusqu'au point où le système est si complexe qu'il en devient inutilisable pour l'utilisateur moyen et requiert une expertise réelle pour naviguer dans les menus et fonctionnalités. Nous savons également que ce qui rend un logiciel plus convivial qu'un autre est souvent compliqué à déterminer : pourquoi préférez-vous utiliser Whatsapp plutôt que Messenger ? Il y a de nombreuses hypothèses – des couleurs plus chaudes, la robustesse (toujours disponible), la confiance (messages cryptés), les temps de réponse, le moteur de recherche, la facilité d'archivage, etc. – mais qui sait exactement ? Les utilisateurs ne peuvent certainement pas nous le dire. Pour eux, choisir l'un par rapport à l'autre est question de pur instinct.

Il est primordial de clarifier les tâches à faire telles que *nous les voyons* si nous voulons avoir une réflexion vraiment profonde à propos de la valeur : travaillons-nous sur les tâches dont les utilisateurs ont réellement besoin ou envie, mais qu'ils ne nous communiquent pas parce que c'est plus intuitif que raisonné ? Nous ne connaissons pas les réponses à ces questions fondamentales, mais nous imaginons des façons de mieux évaluer le comportement réel des utilisateurs et d'analyser les réclamations pour tenter de comprendre ce qu'ils aiment ou n'aiment pas.



Les bénéfices pour les utilisateurs ne constituent toutefois qu'une partie de l'équation. L'autre aspect est le coût total pour les utilisateurs – leur coût de possession et notre coût de développement. Leur coût de possession apparaît dans les bugs qu'ils nous demandent de résoudre ou les fonctionnalités supplémentaires qu'ils souhaitent que nous développions. Nos coûts de développement apparaissent sous deux formes, la production et la retouche : le coût nécessaire du développement (heures de codage, mais aussi coût d'autres fournisseurs, comme le fournisseur de cloud) plus les coûts exceptionnels de développement (heures de débogage et de re-codage).

Au fil de ses lectures Lean, l'un d'entre nous (Woody) est tombé sur le *dantotsu*, une approche menant à une amélioration radicale de la qualité, qui a été rendue populaire par le vétéran de Toyota Sada Nomura. De ce bouquin, Woody a pris à cœur la notion de management des points faibles : comment isoler des problèmes qualité récurrents et les résoudre jusqu'à ce qu'ils ne réapparaissent plus.



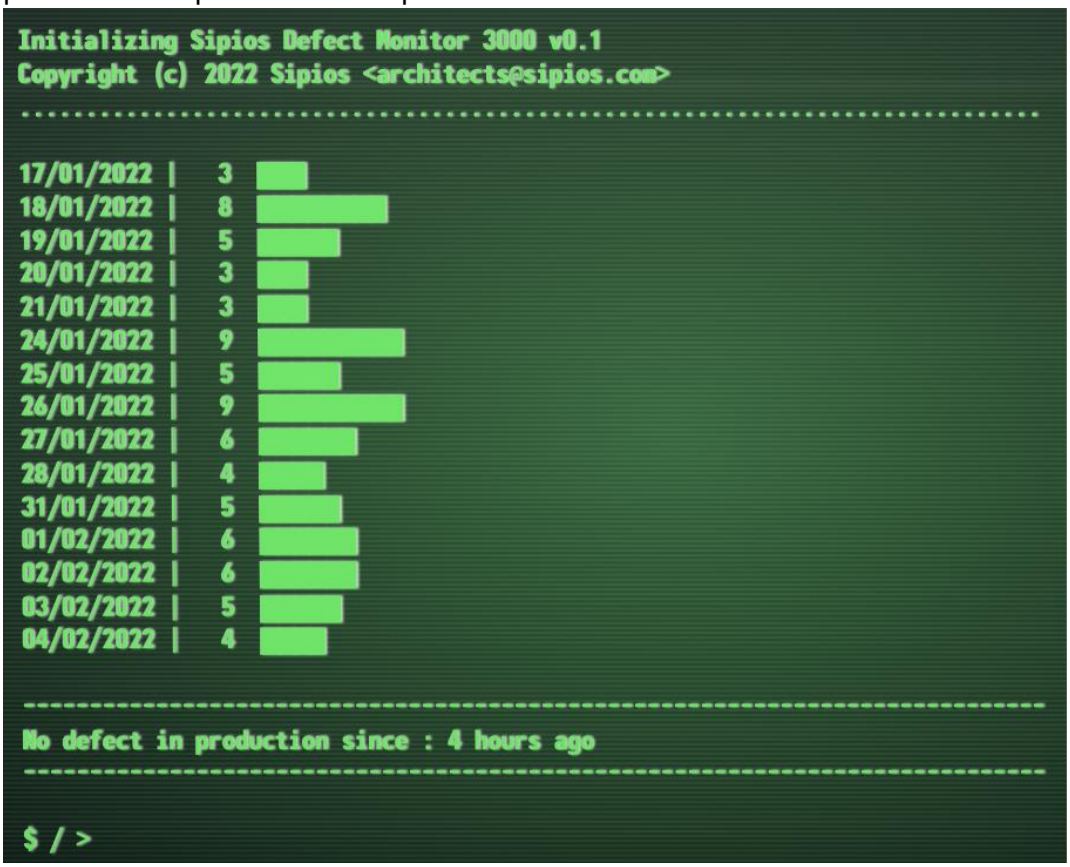
Management visuel dans l'open space de l'équipe, montrant le nombre de bugs détectés par l'utilisateur et pendant la phase d'acceptation du logiciel.

DEFECT COUNTERMEASURE FOLLOW-UP SHEET

Item N°	Defect Info	Defect detailed description	Root cause analysis	Counter-measures	Due date	Owner	Completion status
	Detection date 14/12	Short description of defect In the web page to download documents, the customer downloaded a file with no file extension which he thus couldn't open easily	What is the main reason why the defect occurred? Antoine did not know how to process response headers (where the file extension was provided) when writing an HTTP client	Actions : Thibault writes a standard and trains the developers on writing HTTP clients			
			What are the relevant defect types? ☐ Defective specification ☐ Defective model ☐ Defective API design ☐ Security breach ☐ Poor performance ☑ Defective interface with external system ☐ Defective configuration ☑ Application fragility ☐ Defective infrastructure				
		Defect introduction pull request N° *5094*	What are the relevant workstations? ☐ Functional specification ☐ Technology choices ☐ Architecture ☐ Sequencing ☐ Implementation ☑ Assembly ☐ Deployment				
2	Product name BEL		What is the main reason why the defect was not detected earlier? There was no dataset with a document with an empty 'documentTitle' field, which prevented us from detecting the defect during the acceptance phase of the software	Actions : Thibault enriches the dataset with a new document	25/01	Thibault	
		Defect introduction specification N° *Aldrin n°820*					

Exemple d'analyse de bug produite par le tech lead, le team leader d'une équipe de développement. Il s'agit de l'un des nombreux bugs correspondant à un point faible dans notre développement de logiciel, un assemblage défectueux avec un système externe, ce qu'on appelle des APIs en jargon logiciel.

En tant que fondateur et Directeur Technique de l'entreprise, Woody a investi dans des systèmes pour détecter les bugs en temps réel (une chose étonnamment difficile à faire dans une entreprise de logiciels), comprendre la nature de ces bugs, et commencer une analyse approfondie avec les développeurs de ceux qui reviennent le plus souvent.



Le système affiche les bugs sur tous les produits que Sipios développe pour ses clients. Cette information est affichée à l'entrée de l'entreprise.

Les obstacles sont de taille, car cela doit se faire dans l'urgence de la livraison de chaque projet, avec une pénurie chronique de personnel due à une concurrence féroce pour recruter des développeurs, et une pression constante de la part des clients qui modifient souvent le déroulement des projets alors que le codage est déjà en route. La vie habituelle dans le développement de logiciels ! Pourtant, les premiers résultats sont prometteurs et nous apprenons rapidement sur les problèmes typiques liés à des technologies spécifiques.

Et nous nous sommes rendu compte lors du *gemba walk* de nos fondateurs que ce dont nous avions besoin, c'était de faire concorder les deux approches pour développer une vision de la valeur qui soit plus cohérente: vrais bénéfices pour les utilisateurs / coût total du service.

Nous avons déjà des retours positifs sur les deux volets de l'équation

- Nous discutons de plus en plus avec nos clients de ce qu'est la valeur, là où les discussions traditionnelles avec des fournisseurs d'informatique ont tendance à se concentrer sur les tarifs journaliers et les devis de logiciels. Cela a pour effet de changer la nature de notre relation, en arrêtant de tergiverser sur le prix et le délai et en s'intéressant plutôt au résultat pour le client.
- Les tech leaders ont désormais une compréhension plus pratique de notre stratégie zéro-bug – plutôt qu'un slogan d'entreprise, il s'agit de se concentrer sur des activités claires qu'ils peuvent prendre en charge et discuter.

Il est bien possible que les financiers pur sucre demandent comment cela peut se transformer en croissance et en profitabilité, et jusqu'à présent, nous n'avons d'autre réponse que notre conviction que placer la valeur au cœur de nos discussions avec les clients est la clé pour développer un nouveau type d'entreprise. Une entreprise où la technologie apporte de meilleurs résultats, pas seulement du rendement. C'est ce qui donne du sens à notre travail !

Traduction par Marc-Antoine Guichard, Nicolas Villemain et François Lopez

Les auteurs

Michael Ballé est un auteur et sensei lean, cofondateur de l'Institut Lean France.

Céleste Dhellemmes est Head of Product chez Sipios, du groupe M33 Theodo.

Woody Rousseau est CTO de Sipios.