ÉTUDES, DÉVELOPPEMENT & INTÉGRATION

LE STORY MAPPING

Visualisez vos user stories pour développer le bon produit

Jeff Patton

Préface de Claude Aubry

DUNOD

LE STORY MAPPING

Visualisez vos user stories pour développer le bon produit

Jeff Patton
Consultant et formateur,
spécialiste des méthodes agiles

Préface de Claude Aubry

Traduit de l'américain
par Dominique Maniez

DUNOD

L'édition originale de ce livre a été publiée en anglais par O'Reilly Media Inc.
sous le titre *User Story Mapping*, ISBN 9781491904909.
Copyright © 2014 Jeff Patton.

This translation is published and sold by permission of O'Reilly Media Inc.,
which owns or controls all rights to publish and sell the same.

Toutes les marques citées dans cet ouvrage sont des
marques déposées par leurs propriétaires respectifs.

Adaptation française des illustrations : WIP-Design

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.
Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).

DANGER
LE PHOTOCOPILLAGE TUE LE LIVRE

© Dunod, 2015
5 rue Laromiguière, 75005 Paris
www.dunod.com

ISBN 978-2-10-074407-7

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Préface

Je n'ai jamais rencontré Jeff Patton. Je ne sais même pas s'il est de la famille du général. Mais je l'aime bien.

Ce n'est pas parce que j'ai échangé quelques tweets et mails avec lui, car cela ne remplace pas une conversation directe, ce que vous comprendrez comme jamais en lisant cet ouvrage. C'est la lecture de ses articles, puis de son livre, qui m'ont donné à croire que c'était un bon gars, ce Jeff.

Mon hypothèse se base, notamment, sur le fait qu'il cite volontiers le rock'n roll. Par exemple, dans l'article « Don't Know What I Want, But I Know How to Get It », il a réussi à mentionner les Sex Pistols et les Rolling Stones. Bon, dans une vidéo, il y avait aussi les Spice Girls, ce qui montre que Jeff a ses faiblesses, mais il le reconnait volontiers.

Il sera beaucoup question de conversations et d'hypothèses dans *Le story mapping*, mais d'abord de la pratique qui a donné le titre. Le story mapping est devenu célèbre suite à un seul article de Jeff, écrit en 2008 « *The New User Story Backlog is a Map* ».

Cela a suffi pour convaincre les agilistes que cette pratique serait un bon complément au « backlog Scrum ». Avec une liste d'éléments, tous au même niveau, le backlog est *plat*, tandis que la carte apporte, avec une nouvelle dimension, la « big picture » nécessaire aux personnes impliquées dans le développement.

Le story mapping apporte la carte et revisite la notion de story, centrale dans le développement agile.

Un jour, une stagiaire m'avait envoyé son mémoire sur les méthodes agiles. Le backlog y était présenté comme « *l'ensemble des stories rédigées par le Product Owner qui constitue la spécification* ». C'est un peu l'idée que peuvent en avoir ceux qui n'ont pas bien compris les notions de backlog et de stories. C'est d'abord à eux, tous ces gens qui sont perdus avec des centaines de stories, ceux qui pensent qu'elles doivent être rédigées, ceux qui pensent que c'est le Product Owner seul qui s'y colle, que s'adresse ce livre.

Patton y démonte les anciennes croyances des spécifications complètes, rédigées et figées dès le début, croyances qui perdurent, parfois même dans des organisations soi-disant passées au développement agile.

© Dunod – Toute reproduction non autorisée est un délit.

Dans ces organisations qui ont adopté Scrum, l'équipe multidisciplinaire mise en place est censée contenir toutes les personnes qui contribuent au développement du produit. Dans la pratique, le rôle de Product Owner s'avère délicat à jouer et les contours de ses responsabilités ne sont pas toujours bien définis. La place des autres gens côté métier : analystes métier, experts du domaine, designers d'interface, etc. est encore moins évidente. Ils ne sont généralement pas dans l'équipe Scrum, et leurs interactions avec les développeurs sont sources de difficultés.

Le livre de Jeff Patton leur apprend à collaborer véritablement, pour faire émerger les bonnes idées et une compréhension partagée du produit souhaité.

Les conversations, dont il est question depuis l'invention des user stories par Kent Beck, ont toujours fait sourire quand on explique que c'est cela la base d'une story. Avec ce livre truffé de retours d'expérience, les équipes sauront éviter qu'une conversation ne dégénère en une discussion de comptoir, et en tirer profit pour acquérir une perception commune du produit souhaité.

C'est en relisant l'ouvrage de Jeff que j'ai acquis la conviction de changer le titre d'un chapitre de mon livre sur Scrum. Je l'avais appelé « Définir le produit ». Mais la lecture de la prose pattonnienne m'a convaincu qu'un produit ne définit pas. Il se découvre.

La découverte par l'apprentissage validé, c'est l'occasion pour Jeff Patton de revisiter le *Lean Startup*. Le Lean Startup visant les ...start up, le livre montre comment utiliser ses idées quand on n'est pas une start up.

Il explique qu'une idée n'est qu'une hypothèse, tant qu'elle n'a pas été confrontée à ceux à qui elle est destinée, et validée par eux. Il montre comment exécuter cette boucle rapidement, pour apprendre, afin de décider quoi faire de l'idée de départ.

À cet égard, il intéressera toutes les personnes impliquées dans l'innovation.

En conclusion, vous l'aurez compris, ce livre va bien au-delà du story mapping, car il aborde toutes les facettes de la définition, pardon de la découverte d'un produit.

Avec ou sans ses cartes, Jeff raconte de bien belles histoires, qui plairont à tous ceux qui travaillent sur des produits. Il leur apporte des outils simples et efficaces et surtout leur ouvre l'accès à une nouvelle culture.

Le livre, dans sa version anglaise, a fait l'objet d'une session du club de lecture d'Agile Toulouse. Voici le résumé élaboré en commun par les participants :

« Jeff Patton nous donne à lire un livre d'une grande richesse. C'est indéniablement un ouvrage à mettre dans les mains de tout Product Owner. Ainsi ils pourront changer le monde, comme le suggère Jeff, ou à défaut appréhender pleinement leur rôle. »

Non Jef(f), t'es pas tout seul.

Claude AUBRY
Président de l'association Agile Toulouse,
Product Owner d'iceScrum,
auteur du blog Scrum, Agilité... et Rock'n'roll

© Dunod – Toute reproduction non autorisée est un délit.

Table des matières

Préface III

Avant-propos XI

À lire en premier 1

Chapitre 1 – Vision d'ensemble 15

1.1 Le mot « A » 15
1.2 Raconter des stories, mais ne pas les écrire 16
1.3 Raconter toute l'histoire 17
1.4 Gary et la tragédie du backlog plat 18
1.5 Parler et documenter 19
1.6 Formulez votre idée 21
1.7 Décrivez vos clients et vos utilisateurs 21
1.8 Raconter les stories de vos utilisateurs 22
1.9 Explorez les détails et les options 25

Chapitre 2 – Prévoir de moins créer 31

2.1 Le story mapping aide les grands groupes à améliorer la compréhension mutuelle 32
2.2 La carte vous aide à repérer les trous dans votre story 34
2.3 Il y a toujours trop 35
2.4 Concevoir une release de produit viable minimum 37

2.5 Décomposer une roadmap en releases ... 37
2.6 Ne hiérarchisez pas les features, mais l'impact ! ... 38
2.7 C'est magique, vraiment magique ! ... 39
2.8 Pourquoi nous soutenons tellement les MVP ... 41
2.9 Le nouveau MVP n'est pas du tout un produit ! ... 42

Chapitre 3 – Prévoir d'apprendre plus rapidement ... 45
3.1 Commencer par discuter de l'opportunité ... 46
3.2 Valider le problème ... 46
3.3 Prototyper pour apprendre ... 47
3.4 Attention à l'expression des besoins des utilisateurs ... 48
3.5 Créez pour apprendre ... 49
3.6 Itérez jusqu'à la viabilité ... 51
3.7 Comment se tromper ... 51
3.8 Apprentissage validé ... 53
3.9 Réduisez autant que possible les expériences ... 54
3.10 Récapitulons ... 55

Chapitre 4 – Planifiez pour finir à temps ... 57
4.1 Communiquez avec l'équipe ... 58
4.2 Le secret d'une bonne estimation ... 59
4.3 Créez pièce par pièce ... 60
4.4 Ne livrez pas chaque tranche ... 61
4.5 L'autre secret d'une bonne estimation ... 62
4.6 Gérez votre budget ... 62
4.7 Que ferait Léonard de Vinci ? ... 64
4.8 Itératif et incrémental ... 67
4.9 Stratégies d'ouverture, de milieu et de fin de partie ... 67
4.10 Décomposez votre stratégie de développement à l'aide d'une carte ... 68
4.11 Il s'agit de risques ... 68
4.12 Et maintenant ? ... 69

Chapitre 5 – Vous savez déjà comment faire ... 71
5.1 Écrivez votre story, une étape à la fois ... 71
5.2 Organisez votre story ... 75
5.3 Explorez les stories alternatives ... 76
5.4 Extrayez l'essentiel de votre carte pour constituer une colonne vertébrale ... 78
5.5 Décomposez les tâches qui vous aident à atteindre un outcome spécifique ... 79
5.6 C'est fini ! Vous avez appris toutes les notions importantes ... 80
5.7 Essayez ceci à la maison ou au travail ... 81
5.8 Il s'agit d'une carte actuelle, pas d'une carte du futur ... 82
5.9 Essayez ceci pour de vrai ... 83
5.10 Avec les logiciels, c'est plus difficile ... 84
5.11 La carte n'est que le commencement ... 86

Chapitre 6 – La véritable histoire des stories ... 91
6.1 L'idée étonnamment simple de Kent ... 91
6.2 Simple ne veut pas dire facile ... 93
6.3 Ron Jeffries et les 3 C ... 94
6.4 Des mots et des images ... 96
6.5 C'est tout ... 96

Chapitre 7 – Racontez de meilleures stories ... 99
7.1 Le modèle de chez Connextra ... 99
7.2 Le pattern zombie et le chasse-neige ... 103
7.3 Liste de contrôle de ce dont il faut vraiment parler ... 105
7.4 Créez des photos de vacances ... 108
7.5 Il y a beaucoup de sujets de préoccupation ... 109

Chapitre 8 – Tout ne figure pas sur la fiche ... 111
8.1 Différentes personnes, différentes conversations ... 111
8.2 Il nous faut une fiche plus grande ... 112
8.3 Radiateurs et glacières ... 115

© Dunod – Toute reproduction non autorisée est un délit.

8.4 Cet outil n'est pas conçu pour cela ... 117

Chapitre 9 – La fiche n'est que le commencement ... 121

9.1 Construisez avec une image claire en tête ... 121
9.2 Construisez une tradition orale du storytelling ... 123
9.3 Inspectez les résultats de votre travail ... 124
9.4 Ce n'est pas pour vous ... 125
9.5 Développez pour apprendre ... 126
9.6 Il ne s'agit pas toujours de logiciel ... 127
9.7 Planifiez pour apprendre et apprenez à planifier ... 127

Chapitre 10 – Mitonnez vos stories comme des gâteaux ... 129

10.1 Créez une recette ... 129
10.2 Découpage d'un gros gâteau ... 131

Chapitre 11 – Cassez les blocs ... 135

11.1 La taille a toujours de l'importance ... 135
11.2 Les stories sont comme des pierres ... 137
11.3 Les epics sont de gros rochers, parfois utilisés pour frapper les gens ... 138
11.4 Les thèmes organisent des groupes de stories ... 139
11.5 Oubliez ces termes et concentrez-vous sur le storytelling ... 139
11.6 On commence par les opportunités ... 141
11.7 Découverte d'une solution viable minimum ... 141
11.8 Approfondissez les détails de chaque story au cours de la livraison ... 143
11.9 Continuez à discuter pendant que vous développez ... 144
11.10 Évaluez chaque partie ... 145
11.11 Évaluez avec les utilisateurs et les clients ... 146
11.12 Évaluez avec les parties prenantes de l'entreprise ... 147
11.13 Sortez une release et continuez à évaluer ... 148

Chapitre 12 – Les casseurs de blocs ... 151

12.1 Valable, utilisable et faisable ... 152
12.2 Une équipe de découverte a besoin de beaucoup d'autres personnes pour réussir ... 154
12.3 Les trois amigos ... 155
12.4 Le PO vu comme un producteur ... 157
12.5 C'est compliqué ... 159

Chapitre 13 – On commence par les opportunités ... 161

13.1 Tenez des conversations sur les opportunités ... 161
13.2 Approfondissez, jetez ou réfléchissez ... 162
13.3 Une opportunité ne devrait pas être un euphémisme ... 166
13.4 Story mapping et opportunités ... 167
13.5 Soyez rigoureux ... 171

Chapitre 14 – Utilisation de la découverte pour améliorer la compréhension mutuelle ... 173

14.1 La découverte ne concerne pas le développement ... 173
14.2 Quatre étapes essentielles de la découverte ... 174
14.3 Activités de découverte, discussions et artefacts ... 188
14.4 La découverte sert à ce que tout le monde se comprenne ... 189

Chapitre 15 – Utilisation de la découverte pour l'apprentissage validé ... 191

15.1 Nous avons tort la plupart du temps ... 191
15.2 La sombre époque ... 193
15.3 *Empathize*, *Focus*, *Ideate*, *Prototype*, *Test* ... 193
15.4 Comment gâcher les bonnes choses ... 197
15.5 Courtes boucles d'apprentissage validé ... 198
15.6 Comment Lean Startup modifie la conception de produit ... 199
15.7 Commencez par deviner ... 200
15.8 Donnez un nom à vos hypothèses risquées ... 201
15.9 Concevez et développez un petit test ... 201
15.10 Mesurez en exécutant votre test avec les clients et les utilisateurs ... 203
15.11 Repensez votre solution et vos hypothèses ... 203

© Dunod – Toute reproduction non autorisée est un délit.

15.12 Stories et story maps ? 204

Chapitre 16 – Affinez, définissez et développez 205
16.1 Des fiches, des conversations, encore des fiches, encore des conversations... 205
16.2 Découpage et polissage 206
16.3 Animation des ateliers de stories 206
16.4 Sprint ou planification des itérations ? 209
16.5 Les foules ne collaborent pas 212
16.6 Décomposez et amincissez 213
16.7 Utilisez votre story map pendant la livraison 218
16.8 Utilisez une carte pour visualiser les progrès 218
16.9 Utilisez des cartes simples au cours des ateliers de stories 218

Chapitre 17 – Les stories sont comme des astéroïdes 223
17.1 Réassemblage des blocs décomposés 225
17.2 N'abusez pas du mapping 227
17.3 Ne vous perdez pas dans les détails 227

Chapitre 18 – Tirez un enseignement de tout ce que vous développez 229
18.1 Débriefez en équipe 229
18.2 Débriefez avec d'autres personnes de votre organisation 232
18.3 Suffisamment de logiciel 234
18.4 Apprenez des utilisateurs 235
18.5 Apprenez des utilisateurs au moment de la release 236
18.6 L'outcome dans les délais prévus 236
18.7 Utilisez une carte pour évaluer la préparation de la release 237

Fin provisoire 239

Bibliographie 241

Index 243

© Dunod – Toute reproduction non autorisée est un délit.

Avant-propos

Live in it, swim in it, laugh in it, love in it /
Removes embarrassing stains from contour sheets, that's right /
And it entertains visiting relatives, it turns a sandwich into a banquet.
Tom Waits, « *Step Right Up* ».

Au départ, ce livre était prévu pour être une petite chose... en fait une brochure.

J'ai entrepris d'écrire un livre sur une pratique simple que j'appelle *story mapping*. Moi-même et pas mal d'autres personnes créons des cartes pour faciliter la collaboration et pour imaginer comment un produit sera utilisé.

Le story mapping permet de se concentrer sur les utilisateurs et leur expérience, ce qui améliore l'échange avec eux et, au bout du compte, engendre un meilleur produit.

La création d'une carte (story map) est vraiment très simple. En travaillant avec les autres, je décris la story d'un produit, en listant sur des Post-it® chaque grande étape que les utilisateurs identifient dans la story dans un flux qui va de la gauche vers la droite. Ensuite, on revient sur les détails de chaque étape et on en discute ; on rédige ces détails sur des Post-it® que l'on place verticalement en dessous de chaque étape.

Le résultat est une structure en forme de grille qui décrit une story, de la gauche vers la droite, et la décompose en détail, du haut vers le bas. C'est plaisant et c'est rapide. De plus, ces détails créent un *backlog des stories* de nos projets en développement agile.

Est-ce vraiment compliqué d'écrire un livre sur cette technique ?

Il s'avère néanmoins que même les choses simples peuvent être assez sophistiquées. Savoir pourquoi on veut créer une story map, anticiper sur ce qui va se passer quand on va en créer une, et connaître toutes les manières différentes dont on peut l'utiliser m'a pris beaucoup de temps et j'ai finalement rédigé plus de pages que je ne pensais sur cette pratique apparemment simple.

Si vous utilisez un processus de développement agile, vous remplissez probablement des backlogs avec des user stories. J'ai présupposé que dans la mesure où les stories sont une pratique courante pour vous, cela aurait été une perte de temps pour moi de les décrire dans cet ouvrage. Mais j'avais tort. Depuis une quinzaine d'années où les stories ont été décrites pour la première fois par Kent Beck, elles sont de plus en plus populaires, et de plus en plus mal comprises et mal utilisées. Cela m'attriste et, en plus, cela ruine tous les bénéfices que l'on retire du story mapping.

Je souhaite donc dans ce livre corriger le plus grand nombre possible d'idées fausses sur les stories et sur la manière dont elles sont utilisées dans le développement agile et lean. C'est la raison pour laquelle, pastichant la chanson de Tom Waits, j'ai transformé « ce sandwich en un banquet ».

Pourquoi moi ?

J'aime fabriquer des choses. Ce qui me motive, c'est la joie que j'éprouve à créer un bout de logiciel et à voir les gens l'utiliser et en tirer parti. Je ne suis pas un fanatique de la méthodologie, mais j'ai considéré que j'avais besoin d'apprendre à analyser ma pratique professionnelle pour l'améliorer. Et ce n'est qu'après une vingtaine d'années passées dans le développement logiciel que je commence à apprendre à enseigner le fruit de mon expérience. Et je suis conscient que ce que j'enseigne est une cible mouvante car ce que je comprends évolue toutes les semaines. Comment expliquer ces changements aussi rapides ? Toujours est-il que cela m'a empêché pendant des années d'écrire ce livre.

Mais le moment est venu.

Les stories et le story mapping sont des concepts très intéressants qui ont bénéficié à tant de gens, en améliorant leur vie et les produits qu'ils créent. Mais pourtant pendant que la vie de certains s'améliore, il y a de plus en plus de gens qui pataugent avec les stories et je souhaite que cela cesse.

Ce livre est ma contribution à cette cause et s'il améliore la vie, ne serait-ce que de quelques utilisateurs, j'en serais déjà heureux.

Ce livre est pour vous si vous pataugez avec des stories

Tant d'organisations ont adopté les processus de développement agile et lean, et le concept de story qui s'y rattache, qu'il est possible que vous tombiez dans les pièges suivants engendrés par les idées reçues sur les stories :

- Puisque les stories permettent de se concentrer sur la création de petites choses, il est facile de perdre la vision d'ensemble. On obtient alors souvent un produit monstrueux dont il apparaît clairement aux utilisateurs qu'il a été assemblé à partir de morceaux qui ne vont pas ensemble.
- Quand on crée un produit de taille significative, la création morcelée de petites choses laisse planer des doutes sur la date finale de réalisation ou sur la nature exacte du produit livré. Et si vous êtes le concepteur, vous vous posez également des questions.
- Comme les stories sont basées sur des conversations, les gens usent de ce prétexte pour éviter d'écrire quoi que ce soit. Ils oublient ensuite ce dont ils ont parlé et ce sur quoi ils se sont mis d'accord pendant leurs conversations.
- Comme les bonnes stories sont censées avoir des critères d'approbation, on se concentre sur l'obtention de critères d'approbation écrits, mais il n'y a pas toujours une compréhension mutuelle de ce qui doit être créé. Cela a pour conséquence que les équipes ne terminent pas le travail prévu dans les délais impartis.
- Comme les bonnes stories sont censées être écrites du point de vue de l'utilisateur et qu'il y a de nombreuses parties que l'utilisateur ne voit jamais, les membres des équipes prétendent que leur produit n'a pas d'utilisateurs si bien que les user stories ne peuvent pas fonctionner dans ce cas-là.

Si vous êtes déjà tombé dans l'un de ces pièges, je tenterai d'abord de chasser ces idées fausses qui vous ont induit en erreur. Vous apprendrez à prendre en considération la vision d'ensemble, à animer des conversations productives sur ce que les utilisateurs tentent de réaliser, et à concevoir des logiciels qui aident vraiment les utilisateurs.

À qui s'adresse cet ouvrage ?

La lecture de cet ouvrage est particulièrement recommandée si vous appartenez à l'une de ces catégories :

- **Les chefs de produits et les professionnels de l'expérience utilisateur** doivent lire cet ouvrage pour les aider à faire le lien, d'une part entre la réflexion sur les produits et l'expérience utilisateur, et d'autre part, la réflexion sur les projets tactiques et les éléments du backlog. Si vous avez des difficultés à passer de la vision d'ensemble que vous imaginez aux détails que vos équipes créent, le story mapping va vous aider. Si vous vous échinez à aider les autres à imaginer et à ressentir l'expérience des utilisateurs de votre produit, le story mapping vous sera utile. Si vous vous démenez pour trouver comment lier une bonne expérience utilisateur avec une pratique de conception de produit, ce livre vous facilitera la tâche. Si vous voulez intégrer une expérience de style lean start up dans vos méthodes de travail, cet ouvrage vous aidera.
- **Les Product Owners, les analystes et les chefs de projet** dans les entreprises d'informatique doivent lire ce livre pour les aider à faire le lien entre leurs utilisateurs en interne, les parties prenantes, et les développeurs. Si vous avez des difficultés à convaincre de nombreuses parties prenantes de votre entreprise

© Dunod – Toute reproduction non autorisée est un délit.

à être sur la même longueur d'onde, alors le story mapping va vous aider. Si vous vous démenez pour aider les développeurs à avoir une vision d'ensemble, les story maps vous faciliteront la tâche.

- **Les formateurs en développement agile et lean** qui ont pour objectif d'aider les individus et les équipes à s'améliorer doivent lire ce livre. Et au fur et à mesure que vous progresserez dans cet ouvrage, pensez à toutes les idées reçues que vos collègues véhiculent sur les stories. Utilisez les stories, des exercices simples et les pratiques décrites dans ce livre pour aider vos équipes à se professionnaliser.
- **Toute autre personne**. Quand on utilise un processus agile, on compte souvent sur les rôles comme celui de Product Owner ou d'analyste pour piloter une bonne partie du travail avec des stories, mais une utilisation efficace des stories nécessite que chacun comprenne les bases. Quand les gens ne maîtrisent pas les bases, on entend des plaintes sur le fait que les stories sont mal écrites, qu'elles sont trop grandes ou bien encore qu'elles n'ont pas assez de détails. Ce livre vous aidera, mais peut-être pas de la façon dont vous l'imaginez. Vous allez apprendre que les stories ne sont pas une méthode pour écrire de meilleures exigences, mais un moyen d'organiser de meilleures conversations. Ce livre va vous aider à comprendre les types de conversations que vous devez mener pour obtenir les informations dont vous avez besoin au bon moment.

J'espère que vous avez pu vous identifier à l'un de ces groupes que je viens de décrire et, si tel n'est pas le cas, donnez cet ouvrage à quelqu'un qui en fait partie.

Conventions utilisées dans ce livre

Comme je pense que ce n'est pas le premier livre sur le développement logiciel que vous lisez, vous ne devriez pas avoir de surprise.

Les **titres** à l'intérieur de chaque chapitre présentent le sujet traité : utilisez-les pour vous repérer ou bien pour sauter les passages qui ne vous intéressent pas dans l'immédiat.

Les points clés sont représentés de cette manière. Imaginez que je hausse la voix par rapport aux autres parties du texte.

Si vous faites une lecture rapide, lisez les points clés. Si vous les trouvez intéressants, lisez le texte qui les encadre, ce qui devrait les rendre plus intelligibles.

Les encadrés sont utilisés pour décrire plusieurs choses :

– Les concepts intéressants mais non critiques. Ils devraient être récréatifs, du moins telle était mon intention.

– Les recettes pour des pratiques spécifiques. Vous devriez pouvoir utiliser ces recettes pour vous assister dans l'apprentissage d'une pratique spécifique.

– Les stories et les exemples issus d'expériences d'autres utilisateurs. Cela devrait vous donner des idées que vous pourriez expérimenter dans votre organisation.

© Dunod – Toute reproduction non autorisée est un délit.

Cet ouvrage est organisé en sections spécifiques. Vous pouvez lire une section à la fois, ou bien employer les sections pour vous aider à trouver des idées pour résoudre un problème spécifique.

Structure de l'ouvrage

Il y a quelque temps, j'ai acheté une nouvelle imprimante laser couleur. En ouvrant le carton, j'ai vu qu'il y avait une brochure intitulée « *À lire en premier* » en grosses lettres rouges. Je me suis demandé si je devais vraiment lire cela avant toute chose parce qu'en général je ne respecte pas ce genre de consigne. Finalement, j'ai bien fait de lire les consignes car il y avait dans l'imprimante toute une série de pièces en plastique destinées à la protéger pendant le transport et si j'avais branché l'imprimante avant de les enlever, j'aurais pu l'endommager.

Cette anecdote peut sembler hors de propos, mais ce n'est pas le cas.

Ce livre contient un chapitre intitulé « *À lire en premier* » car j'y décris deux concepts majeurs et la terminologie associée que j'emploie tout au long de l'ouvrage. Je souhaite que vous intégriez ces concepts avant de commencer. Si vous entamez le story mapping avant de les avoir compris, je ne suis pas en mesure de garantir votre sécurité.

- **Le story mapping vu du ciel** – Dans les chapitres 1 à 4, vous allez prendre de la hauteur pour mieux appréhender le story mapping. Si vous utilisez les stories depuis un certain temps et avez déjà utilisé une story map, cette section devrait vous suffire pour aller de l'avant.
  Le chapitre 5 vous propose un chouette exercice pour vous aider à apprendre les concepts clés utilisés pour créer une belle story map. Faites-le en groupes dans votre entreprise et chaque participant comprendra de quoi il retourne. Et je vous promets que les futures cartes qu'ils vont créer pour vos produits seront bien meilleures après cet exercice.
- **Comprendre les user stories** – Les chapitres 6 à 12 racontent l'histoire qui se cache derrière les stories, la manière dont elles fonctionnent vraiment et le bon usage que l'on peut en faire dans les projets agiles et lean. Au sein des story maps, il y a beaucoup de petites stories que vous pouvez utiliser pour piloter le développement au quotidien. Même si vous êtes un vétéran du développement agile, je vous promets que vous apprendrez quelque chose sur les stories. Et si vous êtes néophyte en la matière, vous en apprendrez suffisamment pour surprendre vos collègues qui se prennent pour des gourous du développement agile.
- **Améliorer les backlogs** – Les chapitres 13 à 15 vous plongent dans le cycle de vie d'une story. Je traiterai des pratiques spécifiques qui vous aident à utiliser les stories et les story maps, en commençant par les grandes opportunités et en passant par le travail de découverte pour identifier un backlog rempli de stories qui décrivent un produit viable. Vous apprendrez à simplifier toutes les étapes du parcours grâce au story mapping et à d'autres techniques.
- **Améliorer la création** – Les chapitres 16 à 18 vous mènent au cœur de l'utilisation des stories d'un point de vue tactique, itération par itération ou sprint

par sprint. Vous apprendrez à créer des stories dans les délais, à les surveiller au cours de leur construction, à les terminer et à retirer un bénéfice intellectuel de chaque story que vous convertirez en un logiciel fonctionnel.

Je considère que les derniers chapitres de nombreux ouvrages sur le développement logiciel sont superflus et, en général, je les ignore. Malheureusement, je n'écris pas ce genre de chapitres si bien que vous allez devoir lire cet ouvrage en entier. Ma seule consolation est que chaque chapitre vous fournira des contenus utiles que vous allez pouvoir réinvestir immédiatement dans votre travail.

Note du traducteur

La traduction des termes du vocabulaire informatique de l'anglais vers le français est très souvent problématique car le traducteur doit s'inspirer de plusieurs sources qui fournissent parfois des résultats fort différents. Même si le traducteur s'appuie principalement sur l'usage des informaticiens professionnels, il doit aussi prendre en compte le travail des commissions de terminologie (www.culture.fr/franceterme), la terminologie prônée par les éditeurs de logiciels (Microsoft, par exemple, publie des glossaires multilingues pour toutes ses applications), ou bien les choix opérés par les premiers traducteurs de certaines technologies et que l'usage a consacrés.

D'une manière générale, la communauté francophone du développement agile semble avoir privilégié la conservation de la terminologie anglophone et c'est la raison pour laquelle nous avons pris le parti de ne pas traduire les termes comme *release*, *backlog*, *input*, *output* ou bien encore *outcome* (nous vous recommandons vivement à ce sujet de lire le chapitre intitulé « À *lire en premier* » où l'auteur explique les concepts d'*output*, d'*outcome* et d'*impact*). Pourtant, cette communauté n'est pas insensible, loin de là, aux problèmes de traduction et pour s'en convaincre, il suffit de fréquenter assidument le blog de Claude Aubry, éminent spécialiste de la méthode Scrum et qui a rédigé la préface de cet ouvrage. En effet, il existe plus de 150 pages sur le blog de Claude Aubry qui contiennent le mot « traduction » (pour vous en persuader, saisissez dans Google la requête suivante « traduction site:http://www.aubryconseil.com/ »), et certaines pages ont des noms évocateurs, comme « Agacement contre le franglais agile ». À la lecture de ce blog, on constatera que les spécialistes ne sont pas tous d'accord entre eux et que la traduction de certains termes varie également dans le temps...

Même si nous avons majoritairement respecté les usages du monde agile, nous avons néanmoins parfois choisi de traduire certains termes anglais en français, et nous les avons alors fait suivre du mot original en anglais mis entre parenthèses. Quand le contexte le justifiait, nous avons également inséré des notes de bas de page pour expliquer notre traduction. Nous espérons que nos choix terminologiques faciliteront la lecture de cet ouvrage à tous ceux qui sont déjà familiers des méthodes agiles.

À lire en premier

Cet ouvrage n'a pas d'introduction.

Oui, vous avez bien lu ! Vous vous demandez sans doute maintenant pourquoi le livre de Jeff n'a pas d'introduction. A-t-il oublié d'en écrire une ? Avec l'âge, est-ce qu'il commence à décliner ? Est-ce que son chien l'a mangée ?

Non, je n'ai pas oublié d'écrire une introduction pour ce livre. Et non, je ne commence pas à perdre mes facultés. Tout du moins, je n'en ai pas conscience. Et mon chien ne l'a pas avalée. C'est tout simplement que j'ai longtemps considéré que les auteurs passaient beaucoup trop de temps à me persuader que je devais lire leur livre, et que le lieu idéal de ces tentatives pour me convaincre se trouve être l'introduction. Cela a pour conséquence que ce qui est intéressant démarre dans la plupart des ouvrages à partir du chapitre 3, et je ne suis pas certain d'être le seul à avoir pris l'habitude de sauter les introductions.

En fait, cet ouvrage commence ici.

Et je ne vous autorise pas à sauter ce passage car il s'agit vraiment de la partie la plus importante. En fait, si vous ne deviez retenir de ce livre que les deux idées suivantes qui figurent dans ce chapitre, j'en serais déjà très content :"

> Le but de l'utilisation des stories n'est pas d'écrire de meilleures stories.
> Le but du développement de produits n'est pas de fabriquer des produits.

Laissez-moi vous expliquer !

Le jeu du téléphone arabe

Je suis sûr que vous vous souvenez de l'époque de votre enfance où vous jouiez à ce jeu étrange du téléphone où l'on murmure quelque chose à l'oreille de quelqu'un qui à son tour le répète en chuchotant à quelqu'un d'autre, et ainsi de suite, jusqu'à ce que la dernière personne du groupe révèle à voix haute le message complètement transformé, ce qui déclenche l'hilarité générale. Je joue encore en famille à ce jeu avec mes enfants quand nous sommes à table. Note à l'attention des parents : il s'agit d'une

© Dunod – Toute reproduction non autorisée est un délit.

excellente activité pour occuper des enfants qui meurent d'ennui à table à cause des conversations des grandes personnes.

Dans le monde des adultes, ce jeu se poursuit, sauf que l'on ne chuchote plus. On écrit de longs documents et on crée des présentations officielles que l'on transmet à quelqu'un qui en fera quelque chose de complètement différent de ce qui était prévu. Et cette personne utilise ce document pour créer d'autres documents qu'il donnera à différentes personnes. Mais à la différence du jeu auquel nous nous adonnions enfants, plus personne ne rit à la fin…

Quand les gens lisent des instructions, ils les interprètent. Si vous ne me croyez pas, laissez-moi vous montrer quelques exemples d'instructions qui sont vraiment très mal comprises.

Le décorateur du gâteau a écrit littéralement sur le gâteau la consigne qu'on lui avait donnée (*underneath that*, qui est d'ailleurs mal orthographié, signifie en dessous de ça).

Voici la couverture du livre de Jen Yates, *Cake Wrecks* (publié chez Andrews McMeel Publishing et que l'on pourrait traduire par *Naufrages en pâtisserie* ; je remercie Jen et John Yates pour m'avoir fourni ces photos). Ce livre tire son origine de son site Web qui est furieusement drôle :

http://www.cakewrecks.com/

N'y allez pas si vous n'avez pas au moins une heure à perdre ! Le site présente des photos de gâteaux décorés de manière improbable qui défient les lois de la raison, mais que Jen arrive quand même à expliquer. Un des thèmes récurrents de l'ouvrage et du site a trait aux exigences mal interprétées. Bien entendu, Jen n'emploie pas le terme **exigence** car elle n'est pas informaticienne, mais elle parle de *littéral*, car le lecteur lit et interprète littéralement ce qui est écrit. En regardant les photos, j'arrive à imaginer que quelqu'un écoute un client et retranscrit ce qu'il souhaite, puis transmet les consignes à une autre personne qui va décorer le gâteau :

CLIENT. – Bonjour, j'aimerais commander un gâteau.

EMPLOYÉ. – Qu'est-ce que vous voulez que l'on écrive dessus ?

CLIENT. – Pouvez-vous écrire « Au revoir Alice » en violet ?

EMPLOYÉ. – Absolument.

CLIENT. – Et mettre des étoiles autour ?

EMPLOYÉ. – Pas de problème. J'ai tout noté et je vais transmettre tout de suite à la personne qui va décorer le gâteau. Il sera prêt dans la matinée.

Le résultat est illustré ci-dessous.

Ici, le décorateur du gâteau a écrit littéralement « Stars » au lieu de dessiner des étoiles et il a écrit la consigne *in purple* (en violet) sur le gâteau.

En voici un autre. En développement logiciel, on appelle cela une exigence non fonctionnelle :

Dans cet exemple, le décorateur a écrit sur le gâteau que le client avait une allergie aux noisettes.

Il s'agit d'exemples amusants et on peut rire d'avoir perdu quelques dizaines d'euros pour un gâteau, mais il arrive parfois que les enjeux soient beaucoup plus importants.

© Dunod – Toute reproduction non autorisée est un délit.

Vous avez sans doute entendu parler de l'histoire du crash en 1999 de la sonde Orbiter qui a coûté 125 millions de dollars à la NASA[1]. Vous ne la connaissez peut-être pas, mais voici la chute de l'histoire : si un projet coule à cause des exigences et de sa documentation, il s'agit d'un projet de la NASA. En dépit de ses classeurs remplis d'exigences et de documentation, la sonde s'est écrasée parce que la NASA utilisait le système métrique pour ses mesures alors que les ingénieurs de chez Lockheed Martin employaient l'ancien système de mesures impérial pour développer les commandes de navigation des propulseurs du véhicule. Personne ne sait exactement où la sonde a fini sa course, mais certains pensent qu'elle tourne en orbite autour du soleil quelque part après Mars.

Paradoxalement, on s'efforce d'écrire pour communiquer plus clairement et ainsi éviter le risque de malentendu, mais trop souvent, c'est l'inverse qui se produit.

Le partage de documents ne favorise pas la compréhension mutuelle.

Arrêtez-vous une minute et écrivez cette maxime. Écrivez-la sur un Post-it® et mettez-le dans votre poche. Envisagez de vous la faire tatouer quelque part de sorte que vous puissiez la voir quand vous vous préparez le matin pour aller au travail. Quand vous la lirez, cela vous aidera à vous souvenir de ce que je suis en train de vous raconter.

La compréhension mutuelle a lieu quand deux personnes comprennent ce qu'elles imaginent ainsi que leurs motivations. À l'évidence, il n'y a pas eu de compréhension mutuelle entre plusieurs décorateurs de gâteaux et les personnes qui leur ont communiqué des instructions écrites. Et à la NASA, quelqu'un d'important n'a pas compris la même chose que les ingénieurs qui travaillaient sur le système de guidage. Je suis certain que si vous avez été impliqué dans un développement logiciel pendant un certain temps, vous n'avez pas à chercher longtemps dans votre mémoire pour vous rappeler une situation où deux personnes croyaient être d'accord sur une fonctionnalité qu'elles voulaient ajouter au programme, pour finir par se rendre compte qu'elles imaginaient chacune quelque chose de complètement différent.

Favoriser la compréhension mutuelle est étonnamment simple

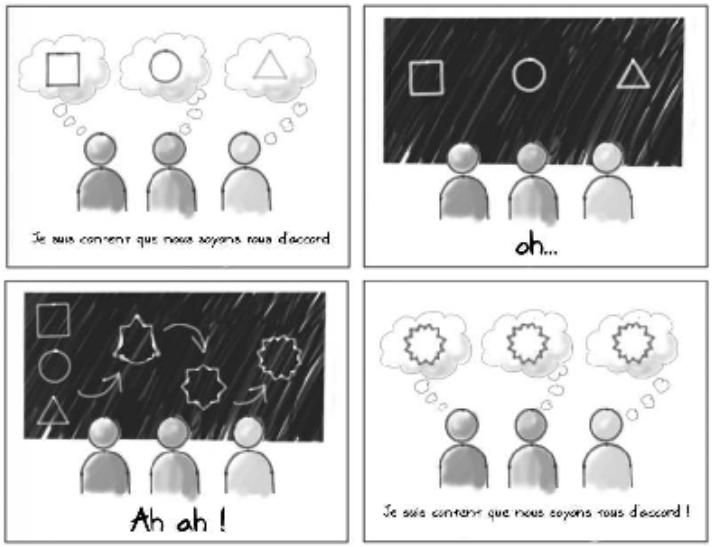
Un de mes anciens collègues, Luke Barrett, a réalisé une bande dessinée pour décrire ce problème. Pendant des années, j'ai vu Luke passer ces quatre images sous la forme de diapositives d'une présentation PowerPoint alors que je n'en voyais pas l'intérêt, les considérant comme trop évidentes. Il semblerait que je sois un peu obtus et il m'a fallu des années pour comprendre la manière dont cette bande dessinée illustre l'idée la plus importante sur l'utilisation des *stories* en développement logiciel.

Le concept est que si j'ai une idée en tête et que je la décris sous une forme écrite, il est fort possible qu'une tierce personne, quand elle va lire ce document, imagine quelque chose de différent. On peut même demander à chacun s'il est d'accord avec ce qui a été écrit et tout le monde vous répondra par l'affirmative.

1. Il y a de nombreux articles qui décrivent les causes du crash, dont notamment celui qui se trouve à www.cnn.com/TECH/space/9909/30/mars.metric.02/.

© Dunod – Toute reproduction non autorisée est un délit.

Cependant, si l'on est ensemble et que l'on se met à parler, vous pouvez me dire ce que vous pensez et je peux poser des questions. La conversation se passe encore mieux si l'on peut extérioriser nos pensées en dessinant des schémas ou en organisant nos idées à l'aide de fiches ou de Post-it®. Si nous prenons chacun le temps d'expliquer nos idées avec des mots et des images, nous favorisons la compréhension mutuelle. C'est précisément à ce moment-là que l'on se rend compte que nous avions compris les choses différemment. C'est pénible, mais maintenant on est au courant.



Il ne s'agit pas de dire qu'une personne a raison et qu'une autre a tort, mais que nous voyons les choses différemment. En combinant et en affinant nos idées différentes, nous finissons par créer une compréhension mutuelle qui inclut l'ensemble de nos meilleures idées. C'est la raison pour laquelle l'externalisation de nos idées est si importante. Nous pouvons redessiner des schémas ou déplacer des Post-it®, mais la chose la plus importante est que nos idées sont réellement en mouvement. En fait, nous faisons évoluer notre compréhension mutuelle, ce qui est extrêmement difficile à réaliser si l'on se contente de mots.

À la fin de la conversation, quand nous faisons référence à une fonctionnalité ou une amélioration, nous désignons en fait la même chose. Nous avons le sentiment d'aller de l'avant ensemble. En fait, c'est la qualité que nous gérons. Malheureusement, cela est impalpable. Il est impossible de voir ou de toucher la « compréhension mutuelle », mais vous pouvez la ressentir.

Arrêtez d'essayer d'écrire des documents parfaits

Il y a beaucoup de gens qui croient qu'il existe une façon idéale d'écrire la documentation et qui pensent que lorsqu'une personne lit des documents et parvient à une

interprétation différente, c'est soit la faute du lecteur, soit la faute du rédacteur. En fait, ce n'est ni l'un ni l'autre.

La réponse à ce problème est qu'il faut simplement arrêter.

Il faut arrêter de tenter d'écrire des documents parfaits.

Allez-y et écrivez quelque chose, n'importe quoi. Puis ayez des conversations constructives avec des mots et des images pour forger une compréhension mutuelle.

> Le véritable but de l'utilisation des stories est la compréhension mutuelle.

Les stories du développement agile tirent leur nom de la manière dont elles doivent être utilisées, et non pas de ce que vous écrivez. Si vous employez des stories dans un développement et que vous ne communiquez pas en utilisant des mots et des images, vous êtes dans le faux.

Si votre objectif en lisant ce livre est d'apprendre à écrire de meilleures stories, vous vous êtes trompé d'objectif.

Les bons documents sont comme les photos de vacances

Si je vous montre l'une de mes photos de vacances, vous verrez peut-être mes enfants sur une plage et vous ferez un commentaire poli en prétendant qu'ils sont mignons, mais quand je regarde cette photo, je me rappelle une plage particulière d'Hawaï accessible en 4 × 4 après un trajet de plus d'une heure sur une route défoncée. Je me souviens de mes enfants qui pleurnichaient sur leur triste sort je commençais d'ailleurs moi-même à me poser des questions. Nous avions finalement passé une journée merveilleuse sur une plage incroyable, quasiment déserte car très peu de gens avaient fait les efforts auxquels nous avions consenti. Cerise sur le gâteau de cette journée exceptionnelle : des tortues étaient venues se prélasser sur le sable de la plage.

Bien entendu, si vous regardez la photo, vous ne saurez pas tout ça car vous n'y étiez pas. Je me souviens de tout cela car j'étais présent.

Pour le meilleur et pour le pire, c'est la manière dont les documents fonctionnent en réalité.

© Dunod – Toute reproduction non autorisée est un délit.

Si vous participez à de nombreuses discussions sur la création d'un logiciel, puis créez un document qui synthétise le tout, il est possible que vous le transmettiez à une personne qui était présente lors des discussions. Vous serez sans doute d'accord pour dire qu'il s'agit d'un bon document, mais vous devez vous rappeler que votre compréhension mutuelle est bourrée de détails qui ne figurent pas dans le document. Un autre lecteur qui n'a pas assisté aux réunions n'en tirerait pas les mêmes informations que vous. Même s'il prétend qu'il comprend le document, vous n'avez pas à le croire. Réunissez-vous et utilisez le document pour raconter une histoire, de la même manière que je l'ai fait avec ma photo de vacances.

Les documents aident à se souvenir

J'ai entendu des gens plaisanter sur le fait qu'ils utilisaient une méthode agile parce qu'ils avaient arrêté d'écrire des documents. C'est une plaisanterie pour les initiés car un processus piloté par les stories nécessite de nombreux documents pour marcher. Mais ces documents ne ressemblent pas toujours aux traditionnels documents des spécifications fonctionnelles.

Cela comprend les conversations, les schémas, les écrits et le travail avec des Post-it® ou des fiches. Il s'agit aussi des documents de travail que nous avons surlignés et sur lesquels nous avons griffonné des notes. C'est un processus interactif et hautement dynamique. Si vous êtes assis à une table de réunion et qu'une seule personne écrit ce que vous dites dans un système de gestion de stories, vous êtes probablement en train de faire fausse route.

Quand on décrit des stories, presque tout peut être utilisé comme outil de communication. Et au fur et à mesure que nous créons ces stories, écrivons beaucoup de notes, et dessinons de nombreux schémas, nous devons conserver toutes ces traces. Nous les archivons pour les examiner plus tard, les photographier et les ressaisir dans d'autres documents.

Mais souvenez-vous que le plus important n'est pas ce qui est écrit, mais bien ce que nous nous rappelons quand on le lit. C'est le syndrome de la photo de vacances.

Parlez, dessinez, écrivez, utilisez des Post-it® et des fichiers, puis photographiez vos résultats. Encore mieux, prenez une courte vidéo de la conversation où vous décrivez ce qui est au tableau. Vous vous souviendrez ainsi précisément de nombreux détails qu'il aurait été impossible de documenter.

> Pour faciliter la mémorisation, photographiez et prenez de petites vidéos des résultats de vos conversations.

Parlez des choses importantes

Il y a beaucoup de gens qui croient que leur travail consiste à collecter et à gérer des exigences, mais ce n'est pas le cas.

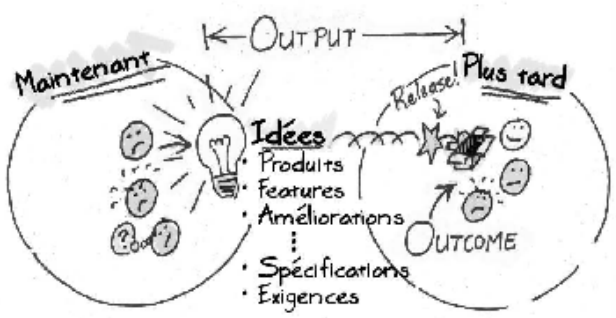
> En vérité, votre travail est de *changer le monde*.

J'ai écrit cela pour attirer votre attention, et j'ai bien conscience que cela ressemble à une hyperbole dans la mesure où ce propos est en général associé à la paix dans le monde, l'élimination de la pauvreté, voire des objectifs encore plus farfelus, comme le fait que des hommes politiques puissent s'entendre. Je suis pourtant très sérieux. Chaque grande idée que vous transformez en une solution pour un produit change le monde à plus ou moins grande échelle pour les gens qui l'utilisent. Et si ce n'est pas le cas, c'est que vous avez échoué.

Aujourd'hui et demain

Il existe un modèle simplifié de changement du monde que j'utilise personnellement et que je garde à l'esprit tout le temps, et que vous devez également avoir en tête quand vous élaborez vos stories et la compréhension mutuelle.

Je dessine ce modèle de la manière suivante :



Le modèle commence à examiner le monde tel qu'il est à présent. Quand on scrute le monde actuel, on trouve des gens qui sont malheureux, en colère, désorientés ou frustrés. Le monde étant vaste, nous allons nous concentrer en grande partie sur les personnes qui utilisent le logiciel que l'on crée ou celles dont on espère qu'elles l'emploieront. Quand vous examinez ce qu'elles font, les outils dont elles se servent et leur manière de faire les choses, il vous vient à l'esprit des idées qui peuvent tourner autour de :

- Produits entièrement nouveaux que vous pourriez créer.
- Fonctionnalités à ajouter à un produit existant.
- Améliorations à apporter à des produits que vous avez conçus.

À un moment, vous allez devoir communiquer les détails de vos idées à d'autres personnes et commencer un travail de conception et de spécification. Si vous communiquez tout ceci à quelqu'un d'autre, vous allez peut-être même appeler tous ces détails des *spécifications*. Mais il est important de se rappeler que les exigences ne sont juste qu'un autre nom pour les idées que l'on a pour venir en aide aux gens.

En fonction de ces exigences, on entre dans un processus qui implique une livraison et un logiciel qui arrivera ultérieurement. Et nous espérons sincèrement que ces personnes qui étaient au départ malheureuses, en colère, frustrées ou désorientées vont retrouver le sourire quand le logiciel débarquera. Pour le moment, ce n'est pas parce qu'ils ont vu la jolie boîte qui contient le logiciel (même si les logiciels ne sont plus livrés dans des boîtes de nos jours) qu'ils sont contents. Ce n'est pas non plus parce qu'ils ont lu les notes de la release, ou bien téléchargé l'appli sur leur mobile, qu'ils sont contents. Ils seront heureux quand ils utiliseront le logiciel, le site Web, l'appli mobile, ou votre création, car ils feront les choses différemment et c'est ce qui les rendra heureux.

La vérité est que vous ne pouvez pas satisfaire tout le monde tout le temps. Votre mère a d'ailleurs dû vous le dire. Certaines personnes seront plus contentes que d'autres de vos productions, et il y aura toujours des gens qui seront mécontents malgré tout le travail que vous aurez fourni et la qualité de votre produit.

Le logiciel n'est pas le problème

Tout ce qui se trouve entre l'idée est la livraison se nomme **output**. C'est ce que l'on crée, le résultat. Ceux qui développent en agile tendent à mesurer la rapidité de l'output et tentent de l'améliorer. Quand les gens créent du logiciel, ils se préoccupent bien entendu du coût de ce qu'ils réalisent et aussi du temps qu'ils mettent à le faire.

Mais bien que cela soit nécessaire, l'output n'est pas vraiment le problème car ce n'est pas réellement l'output qui importe, mais ce qui vient après et que l'on appelle **outcome**. L'outcome est ce qui se produit quand les choses sortent et c'est difficile à mesurer tant que les choses ne sont pas sorties. On ne mesure pas l'outcome par le nombre de *features* livrées ou bien par ce que les gens savent faire à présent. On mesure en fait ce que les gens font de manière différente pour atteindre leurs objectifs

© Dunod – Toute reproduction non autorisée est un délit.

comme une conséquence de ce que nous avons créé, et plus important encore, si nous leur facilitons la vie[1].

Voilà ! Vous avez changé le monde.

Vous avez réalisé quelque chose qui modifie la manière dont les gens peuvent atteindre leurs objectifs, et quand ils utilisent votre produit, leur monde est différent.

N'oubliez pas que votre but ne consiste pas simplement à créer un nouveau produit ou une nouvelle fonctionnalité. Quand vous avez des discussions sur cette fonctionnalité, vous allez évoquer les utilisateurs, leur travail et la manière dont il va changer à l'avenir. Cette amélioration ultérieure est vraiment ce qui intéresse les utilisateurs.

> Les bonnes conversations des stories concernent les utilisateurs et leurs motivations, et pas simplement la technique.

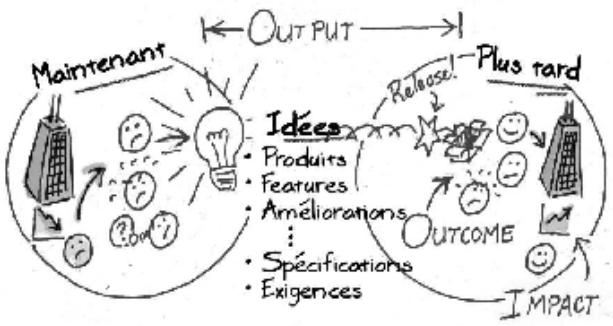
OK, il n'y a pas que les personnes qui comptent

Je m'intéresse aux gens, comme tout le monde, mais honnêtement, il ne s'agit pas que de leur bonheur. Si vous travaillez pour une entreprise qui vous rétribue, vous devez vous concentrer sur ce qui, au bout du compte, lui permet de gagner plus d'argent, de protéger et d'accroître ses parts de marché, ou de faciliter son exploitation. Car si votre entreprise n'est pas en bonne santé financière, alors vous n'aurez pas les ressources nécessaires (ou la main d'œuvre) pour aider qui que ce soit.

Il faut donc que je révise un peu ce modèle. Commençons par scruter à l'intérieur de nos organisations. Vous y trouverez encore plus de gens qui ne sont pas heureux. Et c'est en général parce que les affaires ne se déroulent pas aussi bien qu'ils l'espéraient. Pour régler ce problème, ils peuvent avoir des idées pour se consacrer à des clients ou des utilisateurs spécifiques afin d'améliorer les logiciels dont ils se servent. Finalement, vous voyez, cela concerne les gens, parce que :

> Votre entreprise ne peut pas obtenir ce qu'elle veut tant que ses clients et ses utilisateurs ne seront pas satisfaits.

Le flux se poursuit en choisissant les personnes sur lesquelles se consacrer, les problèmes à résoudre et les idées à transformer en logiciel efficace. Et à partir de là, si les clients achètent, que les utilisateurs s'en servent et que tout le monde est content, au bout du compte l'entreprise qui a financé ce développement en verra les bénéfices escomptés. Cela aura des conséquences positives sur les revenus, les coûts opérationnels, la satisfaction des clients et les parts de marché. Cela fera des heureux au sein de votre entreprise. Vous devriez également être satisfait puisque vous aurez contribué à la santé financière de votre entreprise tout en améliorant la vie des gens. C'est une situation où tout le monde est gagnant.



C'est ce bénéfice à long terme qui est la conséquence des bons résultats que j'appelle **impact**. L'outcome est souvent quelque chose que l'on peut observer juste après la livraison, mais l'impact prend plus de temps.

Créez moins

Il existe une vérité dérangeante à propos de l'industrie logicielle, et j'ai peur qu'elle s'applique à beaucoup d'autres domaines. En tant qu'informaticien, je sais que :"

> Il y a toujours plus à créer que nous avons de temps et de ressources à y consacrer.

Une erreur de conception courante dans le développement logiciel consiste à tenter de produire plus d'output plus rapidement. Cela serait logique s'il y avait trop à faire car l'accélération du travail rendrait service. Mais si vous avez bien suivi, vous allez vous rendre compte que votre travail ne consiste pas à créer *plus*, mais à créer *moins*.

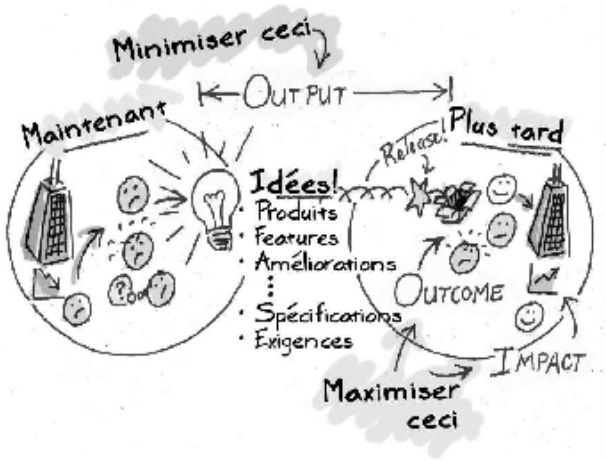
> Minimisez l'output, et maximisez l'outcome et l'impact.

L'astuce consiste à prêter une attention particulière aux personnes dont vous tentez de résoudre les problèmes. Cela inclut les gens qui vont choisir d'acheter le logiciel susceptible de résoudre les problèmes de leur entreprise, les *décideurs*, mais aussi les gens qui vont s'en servir, les *utilisateurs*, même si dans certains cas il s'agit des mêmes personnes.

Votre entreprise a beaucoup de clients et d'utilisateurs potentiels sur lesquels elle doit se focaliser. Votre stratégie d'entreprise doit vous fournir des conseils sur le choix des personnes sur lesquelles vous devez vous concentrer pour obtenir l'impact souhaité. Je peux vous garantir qu'aucune entreprise n'a les ressources pour satisfaire tout le monde car cela n'est pas possible.

1. La distinction entre *output* et *outcome* m'est apparue clairement pour la première fois lors d'une conférence de Robert Fabricant appelée « Behavior Is Our Medium » (*Le comportement est notre médium*, consultable sur http://vimeo.com/3730382).

© Dunod – Toute reproduction non autorisée est un délit.



Ne vous méprenez pas cependant ! Accélérer le développement logiciel est toujours une bonne idée, mais ce n'est jamais la solution.

Un dernier mot sur le monde redoutable des spécifications

Pendant les dix premières années de ma carrière, où je créais des logiciels pour le commerce de détail, je n'ai pratiquement jamais prononcé le terme **spécifications**. C'était un mot qui ne relevait pas de mon champ d'activité. J'avais beaucoup de clients différents qui avaient tous des idées spécifiques sur ce qui leur rendrait service. Je savais aussi que je travaillais pour une société qui devait gagner de l'argent en vendant mes produits. En fait, j'ai passé beaucoup de temps à arpenter des salons pour aider mon entreprise à vendre ses produits à une grande variété de clients. Je savais à la fin de la journée que je devrais continuer à travailler avec ces clients une fois le logiciel développé et livré, si bien que je m'appliquais dans ma tâche, pour agir dans leur intérêt. Cela impliquait que je ne pouvais pas donner à tous tout ce qu'ils voulaient, parce qu'ils désiraient des choses différentes. De plus, ma société et mon équipe n'avaient pas un temps infini si bien que je devais travailler dur pour trouver comment contenter les clients. Cela peut paraître frustrant, mais en fait c'est ce qui motive.

Comme l'entreprise fructifiait, nous avons embauché plus d'informaticiens traditionnels. À un moment donné, une collègue d'une autre équipe est venue me voir en me disant : « Jeff, j'ai besoin que tu fasses des modifications sur un de tes produits en cours de développement. »

Je lui ai répondu : « Pas de problème. Dis-moi pour qui c'est et quel problème cela va résoudre. »

Sa réponse a été : « C'est dans les specs. »

J'ai répondu : « Je comprends. Dis-moi simplement pour qui c'est, comment ils vont l'utiliser et où cela se situe dans leur processus. »

Elle m'a regardé comme si j'étais stupide et m'a répété une dernière fois avec un air qui ne souffrait aucune réplique : « Il y a des specs. »

C'est à ce moment-là que j'ai appris que le mot *spécifications* signifiait en réalité *ferme-la*.

Pour un grand nombre de personnes, c'est exactement l'effet que produisent les spécifications. Elles stoppent net leurs conversations à propos des gens dont nous résolvons les problèmes. En vérité, si vous développez une partie de ce qui est exigé, vous pourrez quand même donner entière satisfaction à vos utilisateurs[1].

N'oubliez pas qu'à la fin de la journée, votre travail n'est pas de vous conformer aux spécifications, mais de changer le monde.

Voilà, c'est terminé

Si vous ne deviez retenir que cela de ce livre, mémorisez ceci :

- Les stories ne sont pas une forme écrite des exigences ; le fait de raconter des histoires de manière collaborative avec des mots et des images est un mécanisme qui crée de la compréhension mutuelle.
- Les stories ne sont pas des exigences ; ce sont des discussions sur la résolution de problèmes pour notre entreprise, nos clients et nos utilisateurs qui mènent à des accords sur ce que l'on crée.
- Votre travail ne consiste pas à créer plus de logiciels plus rapidement : il consiste à maximiser l'outcome et l'impact que vous obtenez à partir de ce que vous avez choisi de créer.

Les stories, telles qu'elles sont conçues, sont une manière complètement différente de se représenter les défis auxquels on fait face quand on crée des logiciels, et bien d'autres choses encore. Si vous pouvez collaborer efficacement et créer des choses pour résoudre des problèmes, vous allez être le maître du monde. Ou au moins régner dans le petit secteur de vos produits.

J'espère qu'en lisant ce livre, vous reviendrez aux fondamentaux de l'utilisation des stories. Je vous souhaite de collaborer, d'écrire des stories sur vos utilisateurs et vos clients et sur la manière dont vous pouvez les aider. J'espère que vous allez dessiner des images et créer des modèles remplis de Post-it®. J'espère que vous allez vous engager et que vous serez créatif. J'espère que vous allez vous sentir différent. Parce que quand vous agissez comme il faut, vous êtes dans le vrai. Et c'est aussi beaucoup plus amusant.

Il nous faut à présent parler de la partie la plus intéressante du travail avec les stories, le moment où l'on utilise une *story map*.

© Dunod – Toute reproduction non autorisée est un délit.

1. Comme je suis parfaitement d'accord avec ce sentiment, je paraphrase la manière dont Kent Beck met en garde contre la mauvaise utilisation du terme exigence dans son ouvrage *Extreme Programming Explained* (Addison-Wesley).

1

Vision d'ensemble

« J'aime le développement agile ! Au bout de quelques semaines, on voit déjà le logiciel fonctionner. Mais c'est comme si j'avais perdu la vision d'ensemble. »

Si je recevais un euro chaque fois que j'entends ce genre de propos de la part d'un membre d'une équipe de développement agile, je serais millionnaire... Vous-même, vous vous êtes déjà peut-être fait cette réflexion. Eh bien, j'ai une bonne nouvelle pour vous. Le fait d'utiliser un processus agile et une approche pilotée par les stories ne signifie pas que vous deviez sacrifier la vision d'ensemble. Vous pouvez encore avoir des discussions intéressantes sur votre produit dans son ensemble tout en constatant les avancées régulières du logiciel.

Étant donné que vous avez patiemment lu le chapitre « À lire en premier », je vais vous faire grâce de tous les lieux communs sur les stories et vous expliquer directement comment le *story mapping* peut résoudre un des problèmes les plus importants du développement agile. Si vous êtes déjà familiarisé avec l'écriture des stories dans des projets agiles, ce chapitre peut suffire à vous aider à démarrer.

1.1 LE MOT « A »

Si vous lisez ce livre, vous savez probablement que le story mapping est une méthode de travail basée sur les user stories qui sont utilisées dans les processus agiles. C'est en général à ce moment-là que tout livre ayant quelque chose à voir avec le développement agile reproduit le *Manifeste pour le développement logiciel agile*, cette chose écrite en 2001 par 17 gars qui étaient frustrés par certaines tendances des grands processus contre-productifs de l'époque. Je suis heureux qu'ils l'aient rédigé et je suis ravi que leur travail ait eu autant d'impact sur tant de monde.

Mais je suis désolé de vous décevoir car je ne vais pas réimprimer le manifeste ni m'enthousiasmer sur son importance. Je pense que vous savez déjà pourquoi c'est

© Dunod – Toute reproduction non autorisée est un délit.

quelque chose qui compte et si vous n'avez pas encore lu le manifeste, alors vous devriez le faire.

À l'emplacement que le manifeste aurait pris dans ce chapitre, j'ai préféré inclure une image amusante de chaton[1]. Pourquoi ? Parce qu'il a été prouvé maintes et maintes fois que des photos amusantes de chaton sur Internet focalisent, au-delà de toute espérance, beaucoup plus l'attention que n'importe quel manifeste.

Vous vous demandez sans doute ce que ce chaton a à voir avec le développement agile. En fait, rien ! Mais l'agilité a certainement quelque chose à voir avec ce livre et les stories, ainsi que l'évolution du story mapping.

< *Où l'on entend une petite musique de flash-back…* >

En 2000, je travaillais dans une start-up à San Francisco, et la société a embauché Kent Beck (le gars qui a créé *Extreme Programming* et qui a été le premier à décrire le concept de story) comme consultant pour dynamiser le processus de développement logiciel. Ce qu'il faut retenir de cette histoire ancienne, c'est que le concept de story n'est pas tout neuf. Si vous commencez juste à utiliser des stories, vous n'êtes plus considéré comme un précurseur, ce qui était le cas il y a une dizaine d'années. Kent et les autres pionniers de l'*Extreme Programming* savaient que par le passé toutes ces méthodes pour traiter des exigences ne fonctionnaient pas bien. L'idée simple de Kent était qu'il fallait se réunir et raconter nos stories ; ainsi, en parlant, il serait possible de construire la compréhension mutuelle, et ensemble nous arriverions à de meilleures solutions.

.2 RACONTER DES STORIES, MAIS NE PAS LES ÉCRIRE

Quand j'ai entendu pour la première fois le terme de *story*, j'admets que cela m'a agacé. L'idée de banaliser les choses importantes que les gens voulaient en les appelant stories ne me semblait pas bonne. Mais je suis un esprit lent, comme je l'ai déjà signalé lors de la discussion sur la compréhension mutuelle. Il m'a fallu un certain temps pour véritablement comprendre ceci :

1. Cette photo, qui a été prise par Piutus, est disponible sur Flickr (https://www.flickr.com/photos/alreadytaken/2504312455/) en licence CC-BY.

© Dunod – Toute reproduction non autorisée est un délit.

> Les stories tirent leur nom de la manière dont elles doivent être utilisées, et non pas de leur contenu.

Avant même que je comprenne vraiment pourquoi les stories méritaient ce nom, je me suis rendu compte que je pouvais écrire un tas de stories, sous la forme d'une phrase ou d'un titre abrégé sur un Post-it® ou une fiche. Je pouvais les déplacer et les hiérarchiser pour décider laquelle était la plus importante. Une fois, j'ai décidé qu'une story était plus importante qu'une autre, et ensuite nous avons entamé une discussion à ce sujet. C'était génial. Pourquoi n'avais-je jamais pensé auparavant à écrire des choses sur des fiches et à les organiser de cette façon-là ?

Le problème est que cette fiche pouvait être quelque chose qui mobiliserait un développeur de logiciels quelques heures, quelques jours, voire quelques semaines. Et cela, je ne le savais pas tant que nous n'en avions pas discuté.

Lors de mon tout premier projet agile, alors que je travaillais avec des stories, j'avais commencé une conversation à propos d'une story et quand j'ai appris que ma story était trop grande, je me suis mis en colère. J'avais l'espoir que ma story soit réalisée lors de l'itération suivante, mais les développeurs avec lesquels je travaillais m'ont signifié que ce ne serait pas le cas. Je me sentais coupable d'avoir fait quelque chose de mal. Les développeurs ont alors identifié une petite partie dont nous pourrions parler et qui pourrait être accomplie dans notre prochaine itération. Mais je suis resté frustré que nous ne puissions pas parler de la vision d'ensemble. J'avais vraiment envie de comprendre combien de temps prendrait cette grande chose dont j'avais réellement besoin. J'avais espéré que ce serait l'objet de la discussion, mais cela n'a pas été le cas.

1.3 RACONTER TOUTE L'HISTOIRE

En 2001, j'ai quitté l'équipe avec laquelle je collaborais et j'ai commencé à faire les choses différemment. Avec ma nouvelle équipe, j'ai essayé une approche consistant à écrire des stories se focalisant sur la vision d'ensemble. Nous avons travaillé pour comprendre le produit que nous avions à créer tout en faisant des compromis. Nous avons utilisé toute une série de fiches avec des titres de stories pour organiser nos idées et décomposer cette vision d'ensemble en petites parties que nous pourrions créer par la suite. En 2004, j'ai écrit mon premier article sur cette idée, mais je n'ai cependant pas utilisé le terme *story mapping* avant 2007.

Il s'avère que le nom que l'on attribue à quelque chose a de l'importance. C'est après avoir donné à cette pratique un joli nom que j'ai vraiment constaté qu'elle se propageait. À l'époque, j'ai considéré qu'il s'agissait d'une grande invention, jusqu'à ce que je rencontre de plus en plus de gens qui faisaient des choses similaires, si ce n'est exactement les mêmes choses. J'avais découvert un **modèle**.

C'est mon amie Linda Rising qui m'a donné pour la première fois cette définition d'un modèle : quand vous parlez à quelqu'un d'une idée que vous jugez excellente et qu'il

vous répond : « Oui, nous aussi, on fait quelque chose comme ça ». Alors il ne s'agit pas d'une invention, mais d'un modèle.

Le story mapping est un modèle. C'est ce que les personnes raisonnables utilisent pour appréhender un produit dans son ensemble ou une *feature*. C'est ce qu'ils font pour décomposer les grandes stories en parties plus petites. Ne vous sentez pas mal à l'aise si vous n'y arrivez pas tout seul. Vous finirez par y arriver, mais la lecture de ce livre vous permettra d'économiser plusieurs semaines, voire des mois de frustration.

> Le story mapping est prévu pour décomposer les grandes stories quand vous en discutez.

Aujourd'hui, de plus en plus de sociétés ont adopté le concept de story mapping. Mon amie Martina, qui travaille chez SAP, m'a dit dans un courriel de septembre 2013 que : « *… à ce jour, nous avons plus de 120 ateliers de story mapping qui sont officiellement enregistrés. Un grand nombre de product owners adorent le concept. C'est une approche bien établie chez SAP.* »

Il ne se passe pas une semaine sans que j'entende quelqu'un d'autre me dire que le story mapping lui a permis de résoudre un problème. Aujourd'hui, j'en apprends beaucoup plus en parlant aux autres qu'en réfléchissant tout seul.

L'idée originale des stories était simple. Elle a permis de reporter notre attention des documents partagés vers la compréhension mutuelle. Une façon courante d'utiliser des stories consiste à en établir une liste, à les hiérarchiser et à commencer à en parler, puis à les transformer en logiciel, une par une. Dit comme cela, ça paraît assez raisonnable, mais cela peut pourtant engendrer de gros problèmes.

1.4 GARY ET LA TRAGÉDIE DU BACKLOG PLAT

Il y a quelques années, j'ai rencontré Gary Levitt qui était un homme d'affaires en train de lancer un nouveau produit web[1]. "Son site web, qui est à présent disponible se nomme Mad Mimi (acronyme pour *Music Industry Marketing Interface*, https://madmimi.com/). Musicien, Gary avait son propre groupe ; il gérait son groupe, aidait à gérer d'autres groupes et était aussi un musicien de studio qui créait des enregistrements pour ses clients.

Le jour où j'ai rencontré Gary, il avait une commande pour le show d'Oprah Winfrey et devait créer des dizaines de jingles. Les producteurs d'émissions de télévision achètent des jingles de la même façon que les personnes publiant une newsletter achètent des cliparts, un jingle étant finalement une sorte de clipart audio. Gary a eu l'idée d'une application professionnelle qui aiderait les musiciens comme lui et les gens qu'il connaissait à collaborer sur des projets comme celui sur lequel il travaillait ; cette application prenait en charge aussi beaucoup d'autres choses qu'un manager de groupe et un musicien doivent effectuer pour gérer et promouvoir leur groupe.

Gary voulait faire développer un logiciel et il travailla avec quelqu'un qui utilisait une méthode agile. Cette personne a dit à Gary d'écrire une liste de toutes les choses qu'il voulait, de hiérarchiser la liste, puis ils discuteraient des fonctionnalités les plus importantes et commenceraient à les développer une à une. Cette liste de choses est ce que l'on nomme **backlog** dans les processus agiles, et il semblait logique à Gary de créer la liste, puis de commencer par les choses les plus importantes. C'est ce qu'il fit.

Gary a écrit son backlog et l'équipe de développement a commencé à créer des choses petit à petit. Pendant ce temps, Gary finançait chaque partie du logiciel qui était développée. Le logiciel prenait forme lentement, mais Gary anticipait qu'il faudrait beaucoup plus de temps pour qu'il corresponde à sa vision des choses et il allait manquer de trésorerie bien avant la fin du développement.

Je connaissais la personne qui travaillait avec Gary. Cet ami savait que Gary était stressé et il voulait l'aider. Mon ami m'a demandé si je pouvais avoir une conversation avec Gary afin de l'aider à organiser ses idées. J'ai contacté Gary et je l'ai rencontré dans son bureau de Manhattan.

1.5 PARLER ET DOCUMENTER

Gary et moi avons commencé à parler. Et au fur et à mesure qu'il parlait, je rédigeais des fiches avec des points clés de ce qu'il disait. Il y a un mantra que j'aime bien quand je construis des story maps. J'emploie l'expression « parler et documenter », ce qui fondamentalement est un moyen pour que les paroles ne s'envolent pas. Écrivez-les sur des fiches de manière à pouvoir vous y référer plus tard. Vous remarquerez que le fait de montrer du doigt quelques mots sur une fiche vous aide à vous rappeler de la conversation que vous avez eue à ce sujet. On peut les disposer sur la table afin de pouvoir les réorganiser. On commence à utiliser des mots utiles comme *ceci* et *cela* quand on pointe les fiches. Cela économise beaucoup de temps. Aider Gary à extérioriser ses pensées était essentiel pour obtenir une compréhension mutuelle. Ce n'était pas une habitude pour lui, et c'était moi qui écrivais les fiches pendant qu'il parlait.

> Parlez et documentez : écrivez sur des fiches ou des Post-it® pour extérioriser votre façon de penser quand vous raconterez des stories.

1. Si l'histoire de Gary vous intéresse, vous pourrez lire un article dans *Business Insider* et découvrir comment on peut créer une entreprise florissante sans capital risque : http://read.bi/UtcIIE.

© Dunod – Toute reproduction non autorisée est un délit.

Nous avons commencé en plaçant les fiches sur une table, mais rapidement on a manqué d'espace. Gary déménageait ses bureaux le jour où je lui ai rendu visite, et une grande partie de son mobilier n'était pas installée. Nous avons donc déplacé notre carte en construction sur le plancher.

À la fin de la journée, le plancher ressemblait à ceci :

Penser - Écrire - Expliquer - Placer

Quand vous travaillez avec une équipe pour construire une story map, ou quand vous avez des discussions sur un sujet quelconque, créez une visualisation simple pour prendre en charge votre discussion. Une des erreurs à éviter consiste à ne pas laisser s'envoler les idées, ce qui se produit quand on évoque des idées et que les gens hochent la tête comme s'ils avaient entendu. Les idées ne sont pas écrites et personne n'y fait référence, puis, plus tard dans la conversation, les idées refont surface et malheureusement il faut expliquer à nouveau parce que les gens n'ont pas vraiment écouté ou ont oublié.

Prenez l'habitude d'écrire succinctement votre idée avant de l'expliquer.

– Si vous utilisez des fiches ou des Post-it®, **écrivez** quelques mots sur votre idée immédiatement *après y avoir pensé*.

– **Expliquez** votre idée aux autres quand vous pointez le Post-it® ou la fiche. Faites de grands gestes, faites des dessins, racontez des histoires.

– **Placez** les fiches ou les Post-it® dans un espace de travail partagé où tout le monde puisse les voir, les déplacer ou en ajouter. Si tout se passe bien, il y aura beaucoup d'autres idées qui viendront enrichir la carte.

Je trouve que lorsque je fais de mon mieux pour écouter les autres, ce qu'ils disent provoque en moi de nouvelles idées. J'avais l'habitude de tenter de mémoriser ces idées et d'attendre un moment pour les introduire dans la conversation, quitte à l'interrompre purement et simplement si l'occasion ne venait pas assez tôt. Mais je me suis rendu compte que j'arrêtais d'écouter la personne qui parlait, car mon intelligence limitée se concentrait sur la mémorisation de mon idée géniale. Aujourd'hui, je me contente de griffonner mon idée sur un Post-it®, de le mettre de côté et d'attendre le moment opportun de la conversation pour évoquer mon idée. Le fait d'écrire mon idée l'évacue en quelque sorte de ma tête si bien que je peux me concentrer sur ce que j'entends. Et la lecture du Post-it® plus tard m'aide à me souvenir de mon idée et à l'expliquer.

Mais je n'étais pas venu pour enregistrer les exigences de Gary, et la première chose dont nous avons parlé n'était pas cette liste de fonctionnalités car il nous fallait commencer par le début.

1.6 FORMULEZ VOTRE IDÉE

Notre première conversation a porté sur la formulation de l'idée de son produit. Nous avons parlé de son métier et de ses objectifs : « *Pourquoi créez-vous cela ? Parlez-moi des avantages pour vous et pour les gens qui l'utiliseront. Quels problèmes résout-il pour ces gens et pour vous-même ?* ».

Au fur et à mesure que vous lisez ceci, vous allez peut-être vous rendre compte que j'ai en tête un modèle qui prend en compte le présent et le futur. J'essaye de comprendre l'outcome que Gary recherche, et non pas l'output qu'il veut créer.

Si je place deux fiches, l'une sur l'autre, les gens supposent alors que celle qui est au-dessus est plus importante. Sans dire un mot, si je fais simplement glisser une carte au-dessus de l'autre, j'indique quelque chose au sujet de son importance. Essayez cela avec une liste d'objectifs. Volontairement, mettez-les dans le mauvais ordre et regardez la personne avec qui vous travaillez tendre la main pour les réorganiser. J'ai agi de la sorte avec Gary et ses objectifs, et cela l'a aidé à exprimer ce qui était plus important pour lui.

1.7 DÉCRIVEZ VOS CLIENTS ET VOS UTILISATEURS

Gary et moi avons continué à parler et à documenter. La conversation suivante concerna les clients qui achèteraient et ceux qui utiliseraient son logiciel. Nous avons listé les différents types d'utilisateurs. Nous avons parlé des avantages qu'ils en retireraient et nous nous sommes demandés pourquoi ils utiliseraient le produit, puis nous avons imaginé ce qu'ils en feraient. Nous avons construit une grosse pile de fiches. Les cartes semblaient naturellement s'organiser dans la pile, les utilisateurs les plus importants se situant au sommet. Il est amusant de voir comment ça fonctionne ainsi sans qu'une décision explicite ne soit prise.

© Dunod – Toute reproduction non autorisée est un délit.

Avant d'être entré dans les détails, je pouvais déjà voir que le projet de Gary était conséquent. Une des réalités cruelles du développement logiciel est qu'il y a toujours plus à créer qu'il n'y a de temps et d'argent pour le faire. C'est pour cette raison que l'objectif ne doit **jamais** être de tout réaliser. Le but est de minimiser la quantité à programmer. Ainsi la première question que j'ai posée à Gary était : « *De tous ces utilisateurs différents et des choses qu'ils veulent faire, si on devait se concentrer uniquement sur un seul de ces utilisateurs, ce serait lequel ?* ».

Gary en a choisi un et nous avons vraiment commencé à raconter des stories.

Types d'utilisateurs du site Mad Mimi

Voici les différents types d'utilisateurs de l'application Mad Mimi que Gary a décrits.

Le simple fait de les nommer et de rédiger sommairement ce qu'ils veulent nous a aidés à voir qu'il y en avait beaucoup. Même avant de discuter des caractéristiques de l'application, nous avions décidé de reporter la création de programmes pour certains types d'utilisateurs.

1.8 RACONTER LES STORIES DE VOS UTILISATEURS

J'ai ensuite dit : « *OK, nous allons imaginer l'avenir. Supposons pendant une minute que ce produit existe et parlons du quotidien d'une personne qui l'utilise et commençons à raconter la story. Tout d'abord, il va faire ceci, puis cela, et ainsi de suite.* » Et nous avons raconté la story dans un flux qui allait de la gauche vers la droite. Parfois, nous avons fait marche arrière et nous avons replacé des choses à gauche d'autres choses, et comme elles étaient inscrites sur des fiches, il était facile de les réorganiser.

L'autre chose intéressante qui se produit naturellement lorsqu'on travaille avec des fiches est que si j'en mets une à gauche et l'autre à droite, j'indique une **séquence** sans prononcer un seul mot. Pour moi, cela relève de la magie, mais j'avoue que je suis bon public et que je m'émerveille que l'on puisse si bien communiquer sans dire un mot.

> La réorganisation collaborative des fiches permet de communiquer sans dire un mot.

Au fur et à mesure que nous parlons et documentons, et pendant que je retranscris notre conversation, nous construisons quelque chose de vraiment important. Non, il ne s'agit pas de ce tas de fiches sur le sol. La chose vraiment importante est la **compréhension mutuelle**. Nous sommes sur la même longueur d'onde. C'est quelque chose que Gary n'avait jamais fait avec qui que ce soit à propos de son idée de produit, du moins pas à ce niveau de détail. Il n'y avait même jamais autant pensé tout seul. Les idées-force étaient dans sa tête, un peu comme les scènes d'action que vous verriez dans la bande-annonce d'un film.

Avant cela, Gary avait fait ce qu'on lui avait demandé de faire. Il avait écrit un tas de titres de stories, en avait fait une liste et avait parlé de chaque élément à tour de rôle. Les conversations avaient davantage porté sur les détails à programmer que sur la vision d'ensemble. Et il y avait beaucoup de trous dans la vision d'ensemble de Gary. Vous constaterez que même si vous êtes parfaitement clair sur votre story, le fait d'en parler en manipulant des fiches vous aide à découvrir les trous dans votre propre réflexion.

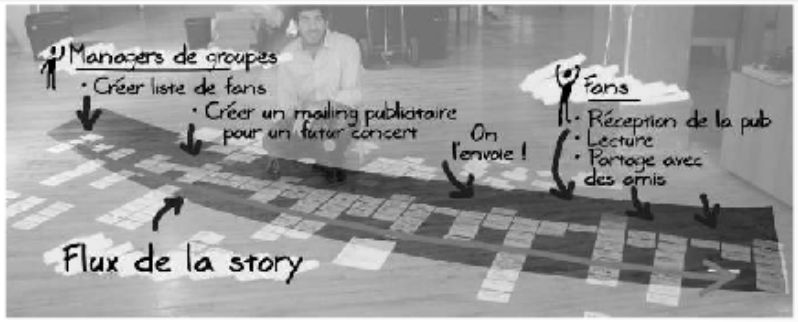
> Le story mapping vous aide à détecter les trous dans votre pensée.

Au fur et à mesure que nous creusions plus profond, nous nous sommes aperçus que la story ne concernait pas qu'un seul utilisateur. Gary a commencé par un **manager de groupe** qui voulait promouvoir son groupe et le travail qu'il accomplissait afin d'en assurer la publicité en envoyant des courriels aux fans. Puis nous avons dû rapidement parler du **fan du groupe** et raconter sa story à propos de la réception du mailing et de son projet d'assister à un spectacle.

Ensuite, si l'on assurait la promotion du groupe quelque part, il faudrait raconter la story du **directeur de la salle de concert** et des informations qu'il aimerait connaître sur le groupe. À ce moment-là, notre cartographie est devenue si large que nous avons buté dans le mur, et nous avons donc dû continuer la story à un autre niveau, sous le premier. C'est la raison pour laquelle la cartographie illustrée sur la photographie comporte deux niveaux.

Au cours de la séance, Gary s'enflammait parfois et commençait à décrire beaucoup de détails. Une fiche au-dessus de l'autre peut indiquer la priorité. Mais il peut aussi

© Dunod – Toute reproduction non autorisée est un délit.



signifier une *décomposition*, qui est juste un mot compliqué pour désigner les plus petits détails qui font partie d'une plus grande chose. Au fur et à mesure que Gary décrivait les détails, je les enregistrais sur une fiche et la plaçais au-dessous de l'étape précédente où l'on avait défini l'utilisateur principal. Par exemple, lorsque Gary a décrit la création du flyer qu'utiliseraient les managers de groupes pour promouvoir leurs concerts, il était passionné et fournissait beaucoup de détails.

Gary a vécu à New York City, et lorsque les groupes doivent créer des prospectus, il imagine toutes sortes de choses originales à partir de ce qu'il a vu collé sur les murs et les lampadaires de New York. Certains peuvent ressembler à des assemblages grossiers photocopiés à la va-vite, mais d'autres sont vraiment élégants et font preuve d'un grand sens artistique. Après avoir enregistré un grand nombre de détails, j'ai dit « *Revenons en arrière et nous aborderons les détails plus tard. Nous allons continuer et faire avancer cette story* ». Il est facile de se perdre dans les détails, en particulier ceux qui vous passionnent. Mais, quand nous essayons d'obtenir une vision d'ensemble, il est important d'aboutir à la fin de l'histoire avant d'enregistrer tous ces détails. J'utilise aussi un autre mantra lors du story mapping, tout au moins à ce stade, qui est « *votre pensée doit avoir un kilomètre de large, mais un centimètre de profondeur* ». Écrivez la fin de la story avant de vous perdre dans les détails.

Focalisez-vous sur la largeur de la story avant de vous consacrer à sa profondeur.

Finalement, nous sommes arrivés à finaliser la story de Gary. Le manager du groupe avait réussi à adresser la publicité d'un concert à des milliers de fans qui avaient relayé le message, et le spectacle avait été un franc succès. La vision du produit était jusqu'à présent très claire dans nos têtes. J'ai alors dit : « *Revenons maintenant en arrière et décrivons les détails, puis nous envisagerons quelques-unes des alternatives* ».

Grande story de Mimi

Si vous parcourez le sommet de la story map de Gary, vous y verrez de grandes activités comme celles-ci :

– Inscription
– Modification de mon service
– Visualisation des stats de mon groupe
– Planification de mes spectacles
– Analyse de l'audience du site web
– Publicité pour un spectacle
– Inscription à la mailing-list d'un groupe
– Affichage des promotions en ligne

Il y avait beaucoup d'autres activités en haut de la story map, mais il s'agit d'un sous-ensemble représentatif qui peut vous donner une idée de ce que l'on peut écrire sur une fiche. Vous noterez que l'on peut déterminer qui fait quoi. Quand Gary a dit : « *Publicité pour un spectacle* », il savait qu'il parlait du manager du groupe. Quand j'ai dit « *Inscription à la mailing liste d'un groupe* », Gary savait que je parlais des fans du groupe. Ces fiches étaient proches de nous et faciles à montrer du doigt durant notre conversation.

« *Publicité pour un spectacle* » est une activité importante. Nous l'avons décomposée en plusieurs étapes disposées de gauche à droite sous la fiche « *Publicité pour un spectacle* ».

– Lancer une campagne de promotion pour un spectacle.
– Repartir du dépliant publicitaire créé pour Mimi.
– Personnaliser le flyer publicitaire.
– Prévisualiser le flyer publicitaire créé.

Vous noterez que ce que nous avons écrit sur toutes les fiches sont des phrases verbales courtes qui décrivent ce que le type d'utilisateur spécifique veut faire. Les écrire de cette façon nous a aidés à raconter la story : « *le manager du groupe ferait ensuite de la publicité pour le spectacle. Pour ce faire, il lancerait une campagne de promotion, puis il repartirait du flyer créé pour Mimi, et le personnaliserait, et puis...* ». Vous remarquerez qu'en plaçant des « *et puis* » entre ce qui est écrit sur chaque fiche, on obtient une belle story.

1.9 EXPLOREZ LES DÉTAILS ET LES OPTIONS

Une fois que la largeur de la story map est en place, il faut commencer son épaississement. Les fiches que nous avons placées en haut de chacune des colonnes de la story map sont devenues les choses importantes, puis les détails sont énumérés en dessous. On s'arrête à chaque étape de la user story et on demande :

- *Quelles sont les choses spécifiques qu'ils feraient ici ?*
- *Quelles sont les autres choses qu'ils pourraient faire ?*
- *Qu'est-ce qui rendrait l'application vraiment cool ?*
- *Que se passera-t-il quand les choses tourneront mal ?* »

À la fin de ce processus, nous avons effectué un retour en arrière et rempli beaucoup de détails. Le résultat est que nous avons raconté la journée ordinaire de la vie d'un manager du groupe, et décrit les autres personnes importantes pour sa réussite : les fans et les directeurs de salles de spectacle.

© Dunod – Toute reproduction non autorisée est un délit.

Les détails

Si vous regardez à l'intérieur d'une étape de la story, comme « *Personnaliser le flyer publicitaire* », vous verriez des détails suivants :

– Télécharger une image.
– Joindre un fichier audio.
– Intégrer une vidéo.
– Ajouter du texte libre.
– Modifier la mise en page.
– Repartir d'une campagne publicitaire que j'ai utilisée auparavant.

Vous pouvez voir que même ces petites étapes auront besoin de beaucoup plus de discussions pour décrire les détails. Mais au moins nous pourrions commencer par tous les nommer.

Notez comme ce qui est écrit sur les fiches sont également ces phrases verbales courtes qui vous aident à raconter des stories. On peut relier tout cela avec des phrases comme « *ou il pourrait* » comme dans l'exemple suivant : « *pour personnaliser le flyer, le manager du groupe pourrait télécharger une image, ou il pourrait joindre un fichier audio ou intégrer une vidéo, ou...* ».

J'ai demandé à Gary : « *Que fait-on maintenant ? Nous avons tous ces autres utilisateurs avec d'autres choses qu'ils veulent faire : voulez-vous qu'on en parle ? Vous pouvez constater que si nous continuons à parler, nous aurons besoin d'une pièce plus grande. Et, Gary, si vous réalisez tout ça, il vous faudra beaucoup d'argent pour créer ce produit. Nous pourrions évoquer le reste de ces activités, mais si lancez votre produit en vous contentant de réaliser ce dont nous avons discuté, ce serait déjà un très bon produit.* »

Gary était d'accord et il a dit « *Je vais arrêter là* ».

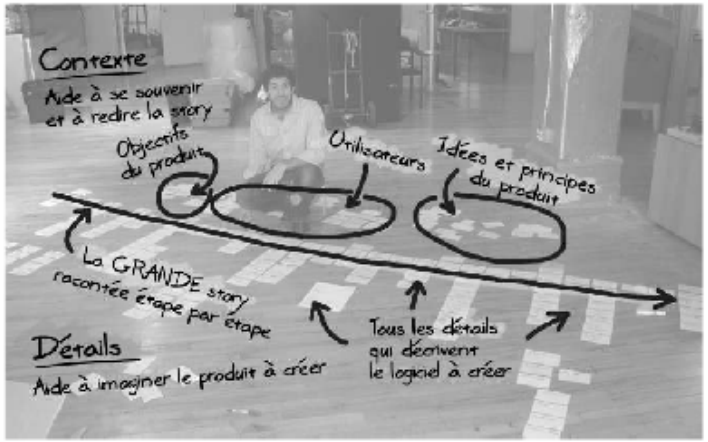
Le plus triste de cette histoire est que j'ai demandé à Gary : « *Vous avez déjà développé une bonne partie du logiciel, mais combien des programmes que vous avez écrits figurent sur cette story map ?* ».

Gary a répondu : « *Presque rien de tout cela, parce que quand j'ai créé ma liste et hiérarchisé les choses, j'ai supposé qu'il fallait commencer par une autre partie. Je pensais à la vision d'ensemble de la chose, un objectif que je mettrai des années à atteindre et maintenant que nous avons eu cette discussion, je me rends compte que je n'aurais jamais dû commencer par là.* »

© Dunod – Toute reproduction non autorisée est un délit.

Le story mapping consiste avant tout à avoir une bonne vieille conversation, puis à l'organiser sous la forme d'une carte. La partie que la plupart des gens regardent est la carte, qui se lit de gauche à droite avec les étapes que les gens suivent pour raconter une histoire. Ce qui va du haut vers le bas concerne les détails. Mais les parties critiques qui définissent le produit et fournissent davantage de contexte sont souvent localisées au-dessus de la carte et à sa périphérie. Ce sont les objectifs du produit et les informations sur ses clients et ses utilisateurs. Si vous conservez une carte accrochée sur un mur, vous trouverez qu'il est judicieux d'ajouter des sketchs d'interface utilisateur (UI) et d'autres notes autour de la carte.

En une seule journée de collaboration, Gary et moi avons construit une compréhension mutuelle autour du produit qu'il voulait créer. Mais il y avait une ombre au tableau et nous le savions. À l'intérieur de chacune de ces fiches que nous avions écrites, il y avait en fait beaucoup de détails qui impliquaient beaucoup plus de discussions. Et, pour Gary, tous ces détails et toutes ces discussions équivalaient à de l'argent dont il aurait besoin pour créer son application, argent qu'il n'avait pas... Il venait d'apprendre une des vérités fondamentales du développement de logiciel : il y a toujours plus à créer que de temps à y consacrer.



Gary faisait beaucoup d'autres grandes hypothèses sur les gens qui utiliseraient son produit (*voudraient-ils vraiment s'en servir et allaient-ils l'employer comme il l'avait imaginé ?*). Mais, à ce moment-là, toutes ces choses n'étaient pas sa plus grande préoccupation. Il lui fallait redoubler d'efforts pour réduire au minimum son idée de produit pour le transformer en quelque chose qui serait d'abord faisable.

L'histoire de Gary s'est finalement bien terminée et, dans le prochain chapitre, je vais raconter l'histoire d'un autre organisme qui a appris à utiliser une story map pour trouver une solution viable quand on a trop de choses à créer.

Artgility – Où la créativité artistique rencontre la créativité informatique

Ceedee (Clare) Doyle
chef de projet agile et formateur, Assurity Ltd, Wellington, Nouvelle-Zélande.

Contexte

The Learning Connection (TLC) est une école d'art située à Wellington, en Nouvelle-Zélande, qui enseigne l'art et la créativité. Les programmes de la TLC sont uniques car ils sont basés sur l'apprentissage par la pratique, ce qui signifie que la pratique est la théorie. En collaboration avec les professeurs, les étudiants développent les idées qu'ils choisissent d'explorer et rédigent des mémoires.

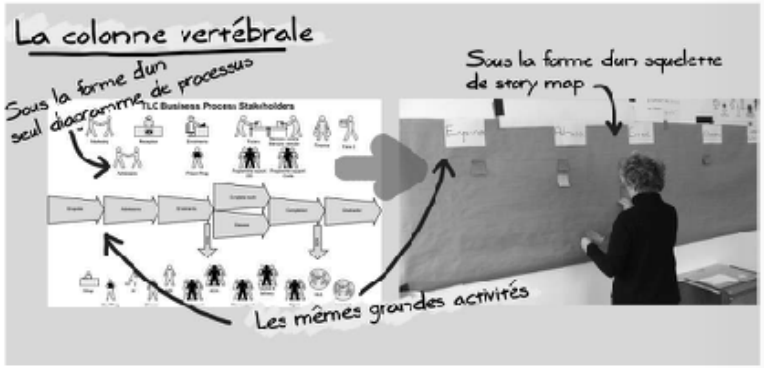
La TLC était un organisme de taille moyenne qui a développé des systèmes informatiques spécifiques au fur et à mesure de ses besoins. Les informations de scolarité étaient saisies dans cinq endroits différents et bien entendu chaque système comportait des données différentes ! La TLC avait besoin de gérer les étudiants qui travaillaient pour elle, situation qui diffère de la plupart des établissements d'enseignement.

La TLC n'avait aucune expérience des projets informatiques. Chaque petite application qui avait été développée l'avait été par quelqu'un (un frère, un ami, un colocataire...) qui avait employé des technologies simples comme Microsoft Excel et Access. La seule application commerciale (utilisée pour les déclarations réglementaires) gérait les données dupliquées provenant des quatre autres sources.

En tant qu'ancien étudiant, j'étais resté en contact avec l'équipe informatique et quand ils ont eu besoin d'aide, ils m'ont appelé. En 2009, cela faisait neuf ans que j'étais dans le secteur informatique et depuis que j'en avais entendu parler, cela faisait trois ans que je voulais faire un projet agile. C'était le bon endroit, le bon projet et le bon moment pour réaliser mon souhait !

Projet Phoenix

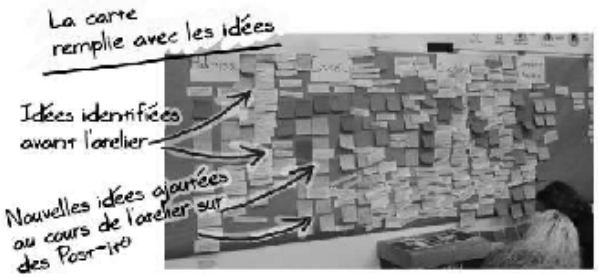
Les premiers ateliers avec le personnel d'encadrement étaient prévus pour durer deux demi-journées. Je travaillais avec un groupe vaste et diversifié, et mon objectif était de développer la compréhension mutuelle. J'ai commencé par une introduction du fonctionnement du story mapping ainsi qu'un aperçu des grandes étapes du processus de gestion des étudiants de l'école.



© Dunod – Toute reproduction non autorisée est un délit.

Jusqu'à ce que je leur montre cette photo (la colonne vertébrale de la story map), chaque membre de l'équipe avait une idée de ce qu'*ils* faisaient (tous ensemble), mais, comme Alice, le promoteur du projet l'a dit, c'était probablement la première fois qu'ils avaient tous une vision claire de leur propre processus métier et de la façon dont toutes les étapes interagissaient.

À partir de là, nous avons réfléchi à ce que les gens voulaient que le système fasse. L'étendue du projet était vaste et les stories nombreuses.



L'intérêt était qu'il s'agissait de gens créatifs qui étaient habitués à la méthode « d'enquête appréciative », si bien que le déballage de toutes leurs idées sur ce que le système devait faire leur était parfaitement naturel.

Les principaux titres (extraits du diagramme) étaient :

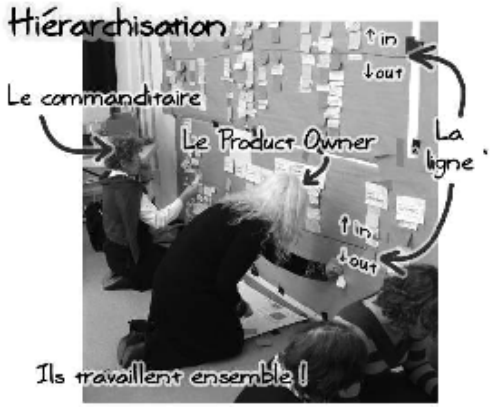
Renseignements → Admissions → Inscriptions → Cours → Réalisation des travaux → Achèvement → Remise des diplômes

En utilisant les directives du story mapping, nous avons ensuite passé en revue chaque section afin de nous assurer qu'elle faisait sens et nous avons obtenu le flux d'un étudiant à travers chaque étape du processus.

Plusieurs personnes comprenaient d'un seul coup pourquoi elles devaient réaliser certaines tâches quand elles se rendaient compte de l'endroit où elles s'inséraient dans l'ensemble du processus ; en revanche, d'autres se rendaient compte qu'elles étaient écartées de certaines étapes, ce qui n'était pas sans conséquence sur leur travail.

En progressant dans la story map et grâce à mon insistance sur le fait de concevoir des stories verticalement, cela a permis d'identifier les endroits où ils pourraient mieux collaborer et les étapes qui étaient en double.

Avant cela, l'équipe avait une vision étroite de ce que chacun faisait, mais ils ont soudainement développé une compréhension mutuelle de la façon dont l'ensemble du processus fonctionnait et ils ont adopté un lexique commun. Par exemple, *Cours* a été renommé *Prestations* parce que tous les étudiants ne suivaient pas des cours.



Quand est venu le moment de hiérarchiser, il n'était pas réaliste d'identifier différents degrés avec des étiquettes comme incontournable, obligatoire ou souhaitable, si bien que nous avons choisi deux états : dedans ou dehors.

C'était très simple : « *nous ne pouvons pas nous passer de* » au-dessus de la ligne et tout le reste en dessous. Après avoir parcouru l'étape Renseignements, l'équipe avait compris et passa la fin de la journée à faire le reste. Le personnel a poursuivi et a commencé à ajouter des sous-titres pour mieux décrire que *« ces choses doivent se produire en même temps, et ensuite ces choses »*. Ainsi, à la fin, ils avaient créé de manière collaborative une vision d'ensemble des étapes qu'un étudiant parcourt, de sa demande initiale de renseignements jusqu'à l'obtention du diplôme.

Ce qui devait durer deux demi-journées s'est transformé en trois jours d'ateliers où les gens allaient et venaient au gré des besoins (ils devaient enseigner, etc.). La flexibilité du lieu de travail signifie que presque tout le personnel est passé par la pièce dédiée au projet Phoenix et y a apporté sa contribution. Ils ont trouvé le processus vraiment utile pour obtenir une vision d'ensemble et pour que les désirs de chacun puissent être inclus. Cela a également permis d'identifier les endroits où il y avait des lacunes et cela a facilité la définition de ce qui était vraiment indispensable. À la fin, nous avions une vision claire de ce qui devait être réalisé dans la première tranche du logiciel.

2

Prévoir de moins créer

Il y a toujours plus à créer qu'il n'y a de personnes disponibles, de temps et d'argent à y consacrer.

Le fait d'employer des formules catégoriques comme « toujours » et « jamais » me met mal à l'aise, mais honnêtement, je n'ai aucun souvenir de situation où cette maxime n'ait pas été vérifiée, bien que je ne possède pas de données statistiques pour étayer cette thèse. Je n'ai jamais rencontré quelqu'un qui me dise qu'on lui avait demandé d'ajouter une nouvelle fonctionnalité, et que ça tombait bien, il disposait justement de temps disponible.

L'une des choses les plus intéressantes dans l'utilisation d'une story map, c'est que cela vous procure, ainsi qu'à vos collaborateurs, un espace pour réfléchir à des solutions de rechange et un moyen d'obtenir des bons résultats dans le temps dont vous disposez.

Prenez une tasse de café et installez-vous confortablement dans un fauteuil car je vais vous raconter une histoire.

Cette histoire est celle de mes amis qui travaillent chez Globo.com, la plus grande entreprise de médias du Brésil. Globo.com possède des chaînes de télévision et des radios, produit des téléfilms et des programmes originaux, et publie des journaux. C'est un géant des médias au Brésil et la plus grande entreprise de médias en langue portugaise au monde.

Globo.com sait mieux que la plupart des entreprises de la planète ce que sont des échéances que l'on ne peut vraiment pas déplacer. Par exemple, elle produit un programme pour la Coupe du monde de football. Si Globo.com est en retard dans ce développement, elle n'a pas le loisir de changer la date de sortie. Pourquoi ? Tout simplement parce que le reste du monde ne va pas modifier les dates de la Coupe du monde de foot. Globo.com va produire du contenu pour les Jeux olympiques que

© Dunod – Toute reproduction non autorisée est un délit.

le Brésil accueillera en 2016, et je peux vous garantir que les délais seront respectés car c'est une obligation. Et elle va aussi produire du contenu pour un grand nombre de nouveaux programmes de télévision et d'émissions de téléréalité. Aucune de ces émissions ne peut être reportée si Globo.com est en retard. Globo.com doit *toujours* finir à temps. Et comme c'est la réalité de son activité, Globo.com est experte en la matière. Ce n'est pas parce que l'entreprise est plus rapide que les autres (bien entendu, elle est rapide, mais pas tant que cela), mais c'est parce qu'elle a compris comment en faire moins.

2.1 LE STORY MAPPING AIDE LES GRANDS GROUPES À AMÉLIORER LA COMPRÉHENSION MUTUELLE

Examinez ceci :

Il s'agit juste d'une partie d'une assez grande carte, élaborée par les dirigeants de huit équipes de trois groupes différents de chez Globo.com qui travaillent ensemble. Les équipes des sports, de l'information et du divertissement ont construit ensemble cette carte pour réfléchir à ce dont ils avaient besoin pour réorganiser et rénover leur système de gestion de contenu. C'est ce système qui pilote tous les sites web d'information, les sites sportifs, les sites des feuilletons, les sites qui permettent de recruter les candidats aux émissions de téléréalité et bien d'autres choses encore. Il doit être capable de gérer de grandes quantités de flux vidéo, des scores en temps réel et les résultats des élections, des photographies, des reportages de dernière minute, etc. Ce système a donc beaucoup de données à gérer et il doit être performant.

Quand je suis arrivé dans les bureaux de Globo.com le jour où ils ont construit cette carte, les équipes qui travaillaient ensemble étaient sur le point de tomber dans le piège du backlog plat. Les équipes avaient, chacune, préparé leurs backlogs priorisés. Il était déjà clair qu'il y avait une énorme quantité de travail à réaliser et que chaque équipe dépendait de l'autre. Par exemple, pour obtenir un bon site d'information, il faudrait non seulement une équipe de journalistes, mais aussi compter sur toutes les autres équipes qui créent les briques de base permettant au site d'actualité d'utiliser des photos, des vidéos, des données en temps réel et bien d'autres choses.

© Dunod – Toute reproduction non autorisée est un délit.

Je me suis assis avec eux et je leur ai rappelé une chose qu'ils savaient déjà : « *Je sais que vous êtes des équipes différentes car vos préoccupations sont différentes, mais on parle ici d'un seul système de gestion de contenu. Cela signifie que vous allez tous devoir être prêts ensemble. Vous ne pouvez pas planifier de release tant que vous ne pourrez pas la visualiser tous ensemble. Vous devez identifier toutes ces dépendances.* » Ils en ont convenu et se sont rapidement remis au travail pour réorganiser leurs backlogs individuels sous la forme d'une carte. En quelques heures, ils ont construit une carte sur le mur à l'aide de Post-it® qui décrivait la story de leur système de gestion de contenu.

Je n'étais pas présent dans la pièce quand les membres des équipes ont travaillé ensemble pour construire la carte. Mais quand j'y suis revenu plus tard dans la journée, j'ai été impressionné de voir la vitesse à laquelle ils l'avaient construite. Leur satisfaction était légitime. Ils avaient réorganisé plusieurs backlogs complexes, et en avaient fait une story map de produit cohérente. À présent, chaque équipe pouvait voir où ses travaux s'inséraient dans la vision d'ensemble.

Une story map pour la version d'un produit impliquant plusieurs équipes permet de visualiser les dépendances.

Anatomie d'une grande carte

La carte de Globo est un bon exemple de l'aspect d'une carte typique *après* les phases de formulation, de cartographie et d'exploration des nombreux détails.

La colonne vertébrale organise la carte

En haut de la carte se trouve la colonne vertébrale, qui a parfois plusieurs niveaux. On peut commencer par le flux de base de la story, qui est le premier niveau. Mais, quand cela devient vraiment long, il est utile d'avoir un niveau supplémentaire pour synthétiser les choses. Plus tard, j'ajouterai certaines précisions terminologiques sur ce que je souhaite placer à chaque niveau, mais je me souviens d'un de mes amis qui me disait d'arrêter d'essayer d'avoir un langage précis et m'assénait cette vérité première : « Il s'agit juste de grandes choses et de petites choses ».

Le résultat est que l'on dirait que vous avez extrait la colonne vertébrale d'un animal étrange. Il y a cette longue dorsale en haut avec de nombreuses vertèbres irrégulièrement espacées d'où pendent de longues côtes de longueur différente.

Carte dans des releases d'ensemble

Cette carte a été construite par plusieurs équipes de chez Globo. Il y a des équipes de développement responsables de la vidéo et des équipes chargées de créer des choses sur le back-end qu'utilisent les éditeurs pour créer et gérer le contenu. Il y a des équipes responsables de certaines métadonnées et des associations entre les données (tout ce foutu balisage sémantique que je n'arrive jamais tout à fait à comprendre). Et il y a des gens qui s'occupent de l'habillage extérieur et de l'aspect général que perçoivent les utilisateurs et les consommateurs. Et il y a encore d'autres personnes qui s'occupent de fonctionnalités spécifiques pour les informations, les sports ou le divertissement.

Plusieurs équipes devaient travailler ensemble sur cette carte, parce que, pour cette révision majeure, aucune équipe ne pouvait produire sa propre release sans les autres. Les équipes n'ont créé qu'une seule carte car elles avaient besoin de penser la release de manière holistique.

Carte dans un flux narratif pour de nombreux utilisateurs et systèmes

La carte a commencé avec les gens sur la gauche et les choses qu'ils avaient à faire pour mettre en place les widgets de base pour les écrans qui contenaient les reportages, les photos et les vidéos. Il existait d'autres types de personnes qui combinaient ensuite ces contenus sur les pages des sites d'information ou des sites de feuilletons. Puis il y avait des éditeurs qui ajoutaient du contenu aux pages. Cette colonne vertébrale raconte la multiplicité des personnes différentes chez Globo.com qui construisent et gèrent le contenu de son site web.

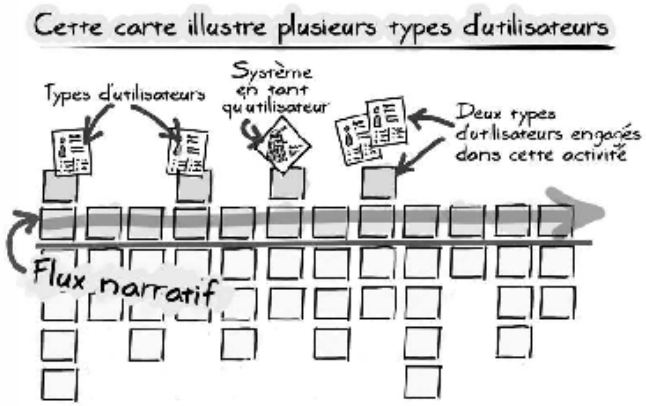
Lorsque vous lisez la colonne vertébrale de la carte de gauche à droite, cela raconte l'histoire avec toutes les personnes qui utilisent le système et sur ce qu'ils font pour créer et gérer des sites et du contenu. L'ordre de gauche à droite, c'est ce que j'appelle un flux narratif, terme pompeux pour dire l'ordre dans lequel on raconterait l'histoire. Bien sûr, tous ces gens font tout en même temps et parfois les choses ne bougent pas dans un ordre parfait, mais nous en sommes conscients. On détermine simplement un ordre qui nous aide à raconter l'histoire.

Dans ce grand système, le flux narratif doit être divisé en plusieurs stories différentes d'utilisateurs et de systèmes. J'aime placer de simples Post-it ® ou des vignettes de *personas* au-dessus de la colonne vertébrale de telle sorte que je puisse voir de qui l'on parle à certains moments de la story. Et cela ne me gêne pas de faire de l'anthropomorphisme pour les services de back-end ou certaines choses complexes du système. Mes amis de chez SAP créent bien des personnages fictifs pour leurs systèmes et utilisent des images de R2D2 ou de C3PO tirées de *Star Wars*.

2.2 LA CARTE VOUS AIDE À REPÉRER LES TROUS DANS VOTRE STORY

Quand je parle avec des gens qui font du story mapping, ils me disent : « *Chaque fois que nous créons des cartes, nous trouvons des trous. Nous trouvons des choses dont on était persuadé qu'une autre équipe devait s'occuper, mais elle l'ignorait. Nous trouvons les choses nécessaires entre les grandes releases importantes dont on avait oublié de parler* ». En créant une carte de manière collaborative, Globo.com avait aussi repéré des trous.

© Dunod – Toute reproduction non autorisée est un délit.



Après avoir imaginé le produit dans son ensemble ou la feature, il est plus facile de commencer à jouer au jeu du « Que se passe-t-il ». C'est le moment où l'on commence à poser la question « Que se passe-t-il quand une erreur se produit » ou bien « Que se passe-t-il avec ces autres utilisateurs ? ». Jouez au jeu du « Que se passe-t-il » avec tout ce qui vous préoccupe et ajoutez des Post-it ® au corps de la carte pour des idées de fonctionnalité dont vous aurez besoin pour traiter ces choses-là dans votre logiciel. Dans le premier chapitre, Gary a joué au jeu du « Que se passe-t-il » pour examiner les options et les solutions de rechange. Lorsque vous ferez cela avec d'autres équipes, vous trouverez qu'elles sont formidables pour repérer les problèmes qui pourraient survenir lorsque différents systèmes se connectent entre eux.

Une des critiques que les gens adressent parfois au story mapping, c'est que chaque fois qu'ils s'asseyent et créent des story maps, ils se retrouvent avec beaucoup trop de choses. Mais je suis convaincu qu'ils trouvent à ce moment-là des éléments qui les auraient gênés par la suite, ce qui est une bonne chose.

Dans l'approche ancienne du développement logiciel, quand on découvrait ce genre de choses ultérieurement, après avoir déjà estimé le délai de développement et s'être engagé sur une date de livraison, on appelait ces nouveaux éléments *l'inflation du périmètre (scope creep)*. Personnellement, je pense que ce n'est pas l'étendue qui augmente, mais la compréhension. Et que se passe-t-il quand les gens créent une story map et découvrent des trous dans leur compréhension ?

> Il n'y a pas d'inflation du périmètre, mais une augmentation de la compréhension.

2.3 IL Y A TOUJOURS TROP

Quand j'ai quitté les équipes de gestion de contenu de chez Globo.com, tout était merveilleux : la compréhension était totale et les équipes savaient quoi faire.

Cependant, quand je suis revenu pour vérifier avec eux quelques jours plus tard, ils se démenaient encore parce qu'ils s'étaient rendu compte qu'il y avait tellement de travail à faire, qu'il faudrait sans doute plus d'un an pour accomplir tout ce qui se trouvait sur la carte. Et, bien entendu, comme les lecteurs avertis le savent, quand des développeurs de logiciels disent que ça va prendre un an pour réaliser quelque chose, ils veulent en fait dire qu'il en faudra deux. Ce n'est pas parce qu'ils sont incompétents, ou qu'ils remettent en cause le calendrier, c'est simplement qu'estimer le temps pour réaliser quelque chose qui n'a jamais été fait auparavant est quelque chose de difficile. Et, par nature, nous sommes souvent des animaux optimistes.

Ils m'ont dit : « Il y a beaucoup trop à faire et ça va prendre du temps».

J'ai demandé : « Est-ce que vous devez tout réaliser ? »

Bien entendu, ils ont répondu : « Oui, parce que cela fait partie d'un seul grand système de gestion de contenu ».

« Mais les projets ne durent pas aussi longtemps ici » répondis-je. « Je connais votre PDG et il va vouloir des résultats beaucoup plus rapides, n'est-ce pas ? »

« Oui », ont-ils confirmé, « il veut voir quelque chose de fonctionnel pour les prochaines élections brésiliennes dans quelques mois ».

« Avez-vous besoin que *tout* cela soit en place pour l'élection ? » ai-je demandé.

Dès que j'ai posé la question, j'ai vu leurs visages s'illuminer. Bien sûr, ils n'avaient pas besoin de *tout*. Jusqu'à présent, ils s'étaient concentrés sur l'identification des séquences et sur les dépendances, en supposant qu'ils auraient besoin de tout créer. En fait, leur problème était une question de temporalité. Ils ont modifié leur façon de penser et ont mis l'accent sur l'impact.

> Il faut se concentrer sur ce que vous espérez voir se passer en dehors du système pour prendre des décisions sur ce qui vous allez mettre à l'intérieur du système.

Globo.com s'est concentrée sur les élections brésiliennes. Les équipes pensaient en particulier au grand *impact* sous la forme de visiteurs favorablement impressionnés, d'annonceurs et de la maison mère de Globo avec des contenus interactifs plus modernes pendant l'élection. S'ils y arrivaient, ce serait un grand succès.

Ce n'était pas la première fois qu'ils étaient confrontés à une date de livraison impossible. Ils y ont réfléchi pendant un moment et se sont rendu compte qu'ils avaient absolument besoin de mettre en ligne le site web d'information et potentiellement d'autres choses qui le prenaient en charge parce que c'était le lieu où les visiteurs se rendraient pour suivre les élections brésiliennes. Le fait de se concentrer sur le site d'actualité signifiait que l'on reportait son attention sur des moyens innovants pour afficher plus rapidement des données électorales en temps réel et des reportages. Et, bien entendu, il y avait un habillage visuel plus moderne qui venait coiffer le tout.

2.4 CONCEVOIR UNE RELEASE DE PRODUIT VIABLE MINIMUM

Les équipes ont pris un rouleau de scotch bleu et ont tracé des lignes sur la carte de gauche à droite pour faire des tranches horizontales. Ils ont ensuite travaillé au déplacement des fiches sur l'axe vertical, pour placer au-dessus des lignes bleues les choses qui devaient être faites dans la première tranche, et en dessous des lignes bleues ce qui pourrait attendre.

L'idée était un peu la suivante : *si nous couvrons les élections brésiliennes, beaucoup de gens au Brésil verront cela. Cela va faire sensation et va influer sur nos sites web, ce qui est bon pour nous. Tout ce qui se trouve dans cette tranche est ce que les utilisateurs devront être capables de faire après la release du logiciel afin que le succès puisse se confirmer.*

> Mettez l'accent sur l'impact (ce que les utilisateurs pourront faire et voir lorsque le système sortira), et décomposez les releases pour que vous obteniez cet impact.

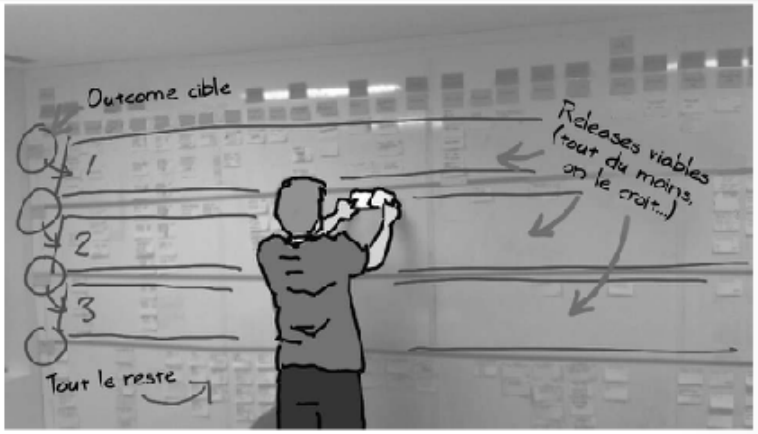
2.5 DÉCOMPOSER UNE ROADMAP EN RELEASES

La carte contenait des choses innovantes qui amélioreraient toutes les propriétés web de Globo.com, mais cela allait vraiment prendre beaucoup de temps pour que tout soit réalisé. La possibilité d'atteindre le marché créé par les élections était également une trop belle occasion à ne pas manquer. En se concentrant sur cela, Globo a pu identifier la première release.

Les équipes ont ensuite travaillé sur les types de propriétés web et les événements liés au marché qu'ils devraient intégrer dans les releases suivantes. Ils ont collé des Post-it® à gauche de chaque tranche avec quelques mots décrivant leur intention pour chaque tranche de release (l'impact cible). Puis, ils ont continué à déplacer des fiches pour les mettre dans les bonnes tranches.

À la fin du processus, ils avaient une stratégie de releases incrémentales qui leur permettait de s'attaquer à tout le travail qu'ils devaient réaliser pour remplacer au fil du temps l'ensemble du système de gestion de contenu, et de telle manière qu'ils pourraient tirer parti de réels bénéfices à chaque nouvelle release. Quand ils lisaient

© Dunod – Toute reproduction non autorisée est un délit.



le bord gauche de la carte, ils avaient une liste des noms des releases, chacune avec un impact cible spécifique. C'est ce que l'on appelle une roadmap.

Vous noterez que cette liste n'est pas un empilement de fonctionnalités, mais bien une liste des avantages réels (n'oubliez pas que votre travail n'est pas de créer un logiciel, mais de changer le monde). La partie la plus difficile est de choisir les personnes pour lesquelles vous voulez changer le monde et de quelle façon.

> Se concentrer sur l'impact cible spécifique est le secret pour hiérarchiser le travail de développement.

L'inverse est également vrai ; autrement dit, si vous ne savez pas ce qu'est votre impact cible (les avantages spécifiques que vous essayez d'obtenir), alors la hiérarchisation est presque impossible.

2.6 NE HIÉRARCHISEZ PAS LES FEATURES, MAIS L'IMPACT !

Vous noterez également comment les équipes de Globo ont démarré avec l'objectif ambitieux de remplacer la totalité de leur système de gestion de contenu. Ce remplacement constitue l'impact, les choses qu'ils allaient devoir livrer. Cela se traduirait par de nombreux impacts positifs. Le secret pour décomposer ce très gros morceau d'output a été de se concentrer sur un impact spécifique, plus petit.

N'oubliez pas que derrière l'impact il y a des changements de comportement spécifique pour des personnes spécifiques engagées dans des activités spécifiques. En mettant l'accent sur les prochaines élections brésiliennes, Globo avait choisi de se concentrer sur les gens qui sont accros aux informations, en particulier ceux qui cherchent des détails de dernière minute sur les élections. Mais, en se concentrant sur ces personnes, elle a laissé tomber les passionnés de feuilletons, les amateurs de sport et beaucoup d'autres types d'utilisateurs. Les autres personnes allaient devoir se satisfaire du site actuel pendant un peu plus longtemps. N'oubliez pas que l'on ne peut pas plaire à tout le monde tout le temps.

2.7 C'EST MAGIQUE, VRAIMENT MAGIQUE !

Je m'enthousiasme facilement, mais la décomposition est vraiment l'une des choses les plus intéressantes quand on organise les idées d'un programme dans une story map.

Le scénario suivant s'est reproduit de nombreuses fois : mon équipe et moi-même plaçons toutes nos idées à propos du produit idéal sur une carte et nous sommes submergés par la quantité de travail à réaliser si l'on veut tout développer. *Tout* semble important. On prend alors du recul et on réfléchit aux personnes spécifiques qui vont utiliser notre produit, et à ce qu'elles auront besoin d'accomplir pour faire leur travail. Nous résumons cela en une ou deux phrases. Puis nous écartons tout ce dont nous n'avons pas besoin, et nous sommes stupéfaits de la taille vraiment réduite de notre solution viable. C'est *magique*.

Gary (voir le chapitre 1) est passé par un processus similaire avec son produit. Il a finalement choisi de se concentrer uniquement sur les managers de groupes, les fans et les administrateurs internes du site Mimi, parce qu'il faut conserver l'exploitation du site. Gary a choisi de laisser de côté les directeurs de salle et les musiciens de production. Il a fini par se rendre compte qu'en se concentrant juste sur ces quelques personnes et sur l'activité de promotion, il aboutirait à une bonne plateforme d'e-mailing publicitaire.

L'extériorisation de notre façon de penser sur une grande carte visible facilite toutes ces étapes et permet à de nombreuses personnes de collaborer à cette opération.

Trouver une release viable plus petite

Chris Shinkle, SEP (Software Engineering Professionals)

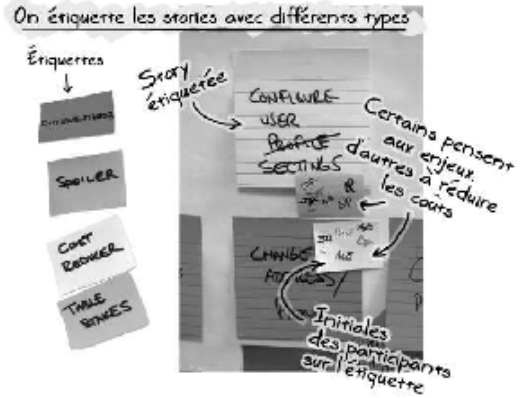
FORUM Credit Union est l'une des coopératives de crédit les plus importantes du pays et l'une des plus avancées technologiquement. Bien que la société dispose d'une bonne culture de développement, elle a contacté la SEP pour créer un nouveau système de services bancaires en ligne, capable de rivaliser avec les solutions commerciales existantes. Parmi ses objectifs figuraient l'ajout de fonctionnalités pour ses services bancaires mobiles, la gestion des finances personnelles et des services de SMS.

La SEP a débuté la prestation par une séance de découverte collaborative du story mapping sur deux jours, qui incluait l'impact, les *personas* et le story mapping. La session a facilité une conversation structurée pour hiérarchiser un grand nombre d'idées sur les features. L'impact et les personas ne suffisaient cependant pas pour hiérarchiser les stories. Au bout de deux jours, la story map couvrait presque deux murs d'une pièce d'une centaine de mètres carrés !

Une fois la carte créée, la SEP a conseillé les intervenants du FORUM grâce à un modèle simple de priorisation :

© Dunod – Toute reproduction non autorisée est un délit.

Facteur de différenciation : feature qui permet de se distinguer des concurrents
Facteur d'interférence (spoiler) : feature qui évolue dans le facteur de différenciation de quelqu'un d'autre
Réducteur de coûts : feature qui réduit les coûts d'organisation
Enjeux : feature nécessaire pour être compétitif sur le marché
La SEP a indiqué chaque catégorie de story en utilisant des Post-it ® de couleur différente. Il s'est ensuivi une intéressante discussion ; en fait, les facteurs de différenciation de certaines parties prenantes étaient les enjeux d'autres parties prenantes. Il était clair qu'un grand nombre de ces conversations avaient lieu pour la première fois ! La story map avec le modèle de hiérarchisation a permis l'émergence de ces conversations. Elle a aidé l'équipe à parvenir à une compréhension mutuelle autour des priorités.



Après l'étiquetage des stories, la SEP a utilisé un système de vote pour aider les intervenants à former et à enchaîner leurs idées afin d'arriver à un ensemble plus significatif de features centrées sur l'impact. À la surprise générale, plusieurs stories ont été jugées inutiles ou pouvant être reportées. Des calculs rapides ont révélé que cela économiserait plusieurs centaines de milliers de dollars, avant même qu'une seule ligne de code ait été écrite.
Interrogé sur l'utilisation du story mapping au démarrage du projet, Doug True, le PDG du FORUM, a déclaré : « Lorsque nous avons lancé ce projet avec un processus de story mapping incluant l'utilisation des personas, j'étais sceptique. Plus précisément, j'étais inquiet du temps investi dans cet aspect annexe du projet. Lors du deuxième jour, il y a eu un déclic et la valeur du temps s'est matérialisée. En fait, maintenant, je ne pourrais pas imaginer de poursuivre un projet de cette ampleur et d'un tel impact sans un tel processus. »

2.8 POURQUOI NOUS SOUTENONS TELLEMENT LES MVP

Il y a un terme qui est utilisé depuis un certain temps dans l'industrie du développement logiciel : MVP (pour *minimum viable product*, ou produit viable minimum). On attribue la paternité de ce concept à Frank Robinson, mais ce sont les définitions actuelles d'Eric Ries et Steve Blank qui prédominent. Même si plusieurs personnes intelligentes ont tenté de définir ce terme, la confusion règne encore, y compris chez l'auteur de ces lignes. Toutes les organisations que j'ai côtoyées et qui utilisaient ce terme avaient chacune leur acception, légèrement différente de celle des autres. Même les gens au sein d'une même organisation ou bien au cours d'une même conversation voulaient souvent dire des choses différentes...

Comme la plupart des mots dans le dictionnaire, ce terme a plusieurs significations. Je vais vous en donner trois définitions : une mauvaise et deux bonnes.

Voici la mauvaise :

> Un produit viable minimum n'est pas le produit le moins mauvais que vous puissiez vous permettre de livrer.

Et le MVP n'*est pas* le produit que vos utilisateurs peuvent utiliser, mais seulement celui qui fonctionnera dans les cas les plus simples et seulement s'ils ont un niveau élevé de tolérance à la douleur. Je vois souvent des entreprises justifier leurs mauvaises décisions avec l'argument que quelqu'un pourrait quand même utiliser le produit, lorsqu'il est clair pour tout le monde que ce ne serait pas le cas.

Si Globo.com avait utilisé cette définition pour décomposer sa première release, elle aurait obtenu un impact négatif. Personne n'aurait été impressionné et la réputation de la marque aurait été ternie, la société se retrouvant finalement plus mal en point que si elle n'avait rien livré du tout.

Quand nous utilisons le terme viable en parlant d'un organisme vivant, cela signifie qu'un organisme peut survivre par ses propres moyens sans mourir. Et nous voulons dire la même chose lorsque nous parlons de logiciel.

© Dunod – Toute reproduction non autorisée est un délit.

Le produit viable minimum est la plus petite release de produit qui réalise avec succès l'impact souhaité.

Je préfère cette définition. *Minimum* est un terme subjectif si bien qu'il convient d'être spécifique sur ce qui est subjectif car il ne s'agit pas de vous. Soyez précis sur ce que sont vos clients et vos utilisateurs, et sur ce qu'ils doivent accomplir. Qu'est-ce que *minimum* signifie pour eux ? Je vous promets que ça va faciliter énormément la conversation, même si elle reste compliquée. Vous pouvez aussi employer la méthode HiPPO (pour *highest paid person's opinion*, l'opinion de la personne la mieux payée). Celle-là est le pire de toutes.

L'expression que je préfère actuellement est *solution viable minimum*. La plupart de mes missions ne concernent pas de nouveaux produits. Il s'agit de nouvelles fonctionnalités ou d'améliorations de fonctionnalités de produits qui existent déjà. Le terme *solution* semble donc plus logique si bien que je vais réviser ma définition :

La solution viable minimum est la plus petite version qui procure l'impact souhaité.

Voici la partie la plus difficile...

On se contente d'émettre des hypothèses.

Quand on décompose un tas de fonctionnalités logicielles et que l'on appelle cela une solution viable minimum, on ne sait pas vraiment si c'est le cas.

Le problème avec l'impact, c'est que vous ne pouvez pas vraiment l'observer tant que les choses ne sont pas sorties. Lorsque vous décomposez une release, vous êtes obligé de faire des hypothèses sur ce qui va se passer. Vous devrez peut-être deviner quels clients achèteront votre produit, quels utilisateurs choisiront de s'en servir, et ce qu'il sera possible de créer dans le temps imparti. Vous êtes obligé de deviner leur degré de satisfaction. Cela fait beaucoup d'hypothèses.

C'est compliqué, parce que si votre hypothèse est trop basse, elle sera inférieure au minimum, et vous aurez échoué. Si en revanche elle est trop élevée, ce que beaucoup de gens font pour se couvrir, alors vous avez dépensé trop d'argent et vous courrez souvent le risque que les choses n'avancent pas du tout. Pire encore, vous pouvez vous planter totalement et rien de ce que vous livrerez ne conviendra.

Il n'est pas étonnant que la définition de « produit pourri que l'on pourrait éventuellement livrer » prospère toujours car c'est celle où l'on n'a pas besoin de faire d'hypothèses.

2.9 LE NOUVEAU MVP N'EST PAS DU TOUT UN PRODUIT !

Je suis bien conscient que certains d'entre vous peuvent ressentir une certaine crispation à la lecture des deux derniers chapitres. Vous avez peut-être pensé en vous-même que *Jeff néglige la chose la plus importante de toutes !* Et vous pourriez avoir raison. Certaines des choses les plus importantes dont vous pouvez discuter au cours du story mapping sont :

- Quelles sont nos plus grandes hypothèses et les plus risquées ? Où se situe l'incertitude ?
- Que puis-je faire pour apprendre quelque chose qui remplacerait les risques ou les hypothèses par des informations réelles ?

Cela m'amène à ma troisième définition du MVP, qui a été popularisée par Eric Ries dans son livre *The Lean Startup* (Crown Business). Eric a appris à la dure, comme la plupart d'entre nous, que nous nous contentons de faire des hypothèses. Eric a travaillé pour une entreprise qui propose un produit qu'elle pensait être viable, mais elle se trompait. Intelligemment, il a changé sa stratégie pour se concentrer sur l'apprentissage, en mettant l'accent sur la validation de toutes les hypothèses que la société avait faites dans sa première release de MVP. Eric juge capital de créer de petites expériences, des prototypes qui testent notre hypothèse sur ce qui est minimal et viable. Et si vous adoptez le point de vue d'Eric, ce que vous devriez faire, votre premier produit devrait vraiment être une expérience, et celui d'après aussi, et le suivant également, jusqu'à ce que vous puissiez vraiment prouver que vous avez le bon produit.

Un produit viable minimum est également la plus petite chose que vous pouvez créer ou faire pour prouver ou réfuter une hypothèse.

Même si c'était plutôt une bonne chose que les gens de chez Globo.com arrivent à formaliser un projet pour créer moins, ils ne se faisaient pas d'illusions. Ils savaient qu'il restait beaucoup à apprendre pour prouver que leurs hypothèses étaient bonnes. À partir de là, ils avaient besoin d'échafauder un plan pour en apprendre plus. Et c'est là où nous reprendrons notre histoire dans le prochain chapitre.

3

Prévoir d'apprendre plus rapidement

Voici mon ami Eric, debout devant son backlog et son tableau des tâches. Il est un Product Owner (PO) et travaille dur avec son équipe pour créer un produit de qualité, mais en ce moment ce n'est pas une grande réussite. Cela n'inquiète pas Eric, car il a une stratégie pour améliorer son produit, et jusqu'à maintenant, elle a fonctionné.

Eric travaille pour une société appelée Liquidnet qui est un réseau commercial mondial pour les investisseurs institutionnels. Bien avant qu'Eric ne vienne poser devant le tableau pour la photo, un de ses collègues a identifié un groupe de clients que Liquidnet pourrait mieux servir et a émis quelques idées pour y parvenir. Eric fait partie d'une équipe qui a pris ces idées et les a exploitées. C'est ce que font les PO, et si vous pensez qu'ils sont toujours les auteurs des idées sur lesquelles ils planchent,

© Dunod – Toute reproduction non autorisée est un délit.

vous vous trompez. Un des aspects difficiles du métier de Product Owner, c'est de s'approprier l'idée de quelqu'un d'autre et de l'aider à la transformer en succès ou bien de lui prouver que ce sera un échec. Les meilleurs PO, comme Eric, aident toute leur équipe à s'approprier le produit.

3.1 COMMENCER PAR DISCUTER DE L'OPPORTUNITÉ

Eric n'a pas commencé son travail en créant un backlog de user stories. Il a débuté avec l'idée brillante de quelqu'un d'autre et l'a traitée, à juste titre, comme une opportunité pour sa société. Il a eu des conversations avec les dirigeants de son entreprise pour mieux comprendre et la discussion a porté sur les points suivants :

- *C'est quoi* cette grande idée ?
- *Qui sont les clients ?* Quelles sont les entreprises qui pourraient acheter le produit ?
- *Qui sont les utilisateurs ?* Quels types de personnes à l'intérieur de ces sociétés pourraient utiliser le produit et qu'en feraient-ils ?
- *Pourquoi en voudraient-ils ?* Quels problèmes ce produit résoudrait-il pour les clients et les utilisateurs qu'ils sont incapables de régler aujourd'hui ? Quel bénéfice les clients retireraient-ils à acheter et à utiliser notre produit ?
- *Pourquoi le créer ?* Si nous créons ce produit et qu'il réussisse, en quoi cela va-t-il nous aider ?

Eric avait besoin de forger une compréhension mutuelle avec ses collègues avant de pouvoir s'approprier l'opportunité. Il savait qu'il allait devoir raconter la story de ce produit plusieurs fois dans les mois à venir, si bien qu'il était préférable d'en comprendre les grandes lignes dès maintenant.

> Votre première discussion à propos de la story a pour but de formuler l'opportunité.

3.2 VALIDER LE PROBLÈME

Eric a confiance dans l'intuition de sa hiérarchie, mais il sait que cette grande idée est une hypothèse et que le seul moyen d'être sûr que cette idée va réussir est de constater son réel succès.

Il a d'abord passé du temps à parler directement aux clients et aux utilisateurs pour apprendre à les connaître. Il a ainsi validé l'hypothèse qu'ils avaient vraiment un problème et qu'ils étaient intéressés par l'achat d'une solution. Eric a parlé aux gens susceptibles d'utiliser le produit. Ils n'avaient pas le produit pour l'instant et n'avaient que de piètres solutions de contournement pour résoudre les problèmes que la nouvelle idée de produit serait censée résoudre.

> Validez le fait que les problèmes que vous voulez résoudre existent vraiment.

Pendant qu'Eric discutaient avec des clients et des utilisateurs, il a constitué un pool de personnes dont il pensait qu'elles feraient de bons candidats pour tester son nouveau logiciel. Certaines entreprises désignent ces personnes sous l'appellation de *partenaires de développement client*. Il est important de garder en mémoire ce détail, parce que cela ressortira plus tard dans la story.

En fait, au cours de cette phase, Eric n'était pas tout seul. Il a travaillé avec une petite équipe qui a passé beaucoup de temps à parler aux clients, et, ce faisant, a trouvé que résoudre le problème n'était pas si facile, et qu'il y avait d'autres problèmes à régler en priorité. Vous devez retenir une chose importante : plus ils en ont appris, plus l'opportunité initiale a été modifiée, voire complètement transformée au bout du compte. On a donc de la chance qu'ils ne se soient pas mis au travail pour créer ce qui avait été demandé au départ car cela n'aurait été d'aucune utilité pour leurs clients ou leur organisation.

Pour l'instant, Eric et son équipe, après avoir parlé aux clients, ont des idées spécifiques sur le type de solution à créer pour que les utilisateurs puissent s'en servir et recueillent ainsi les bénéfices escomptés par leurs employeurs. Voici maintenant ce qu'Eric et son équipe auraient pu faire dans un scénario idéal : ils auraient pu créer un backlog de stories qui décrit leur solution, et constituer une équipe de développement. Comme ce sont des gens intelligents, ils auraient employé une story map pour passer de la grande idée aux parties spécifiques à créer. Mais comme ce sont *vraiment* des gens intelligents, la dernière chose qu'ils vont faire à ce stade, c'est créer un logiciel.

3.3 PROTOTYPER POUR APPRENDRE

C'est ici qu'Eric a commencé à agir en tant que PO. Il a tout d'abord envisagé sa solution comme une série de stories narratives simples, des *scénarios utilisateur*. Puis il a envisagé l'idée comme un simple sketch qui servirait d'armature. Enfin, il a créé un prototype haute-fidélité. Ce n'était pas un logiciel qui fonctionnait, mais un simple prototype numérique, créé avec un outil simple comme Axure, voire PowerPoint.

Pour Eric, tout ceci représente des étapes d'apprentissage qui l'aident à envisager sa solution. Au bout du compte, il veut présenter sa solution à ses utilisateurs pour voir ce qu'ils en pensent. Mais il sait qu'il a d'abord besoin de s'assurer que la solution résout les problèmes des utilisateurs avant de la leur présenter.

> Créez un sketch et un prototype afin de pouvoir envisager votre solution.

Je vous ai cependant caché un détail important. Eric était en fait un designer d'interaction. C'est le genre de designer qui est habitué à passer du temps avec les clients et les utilisateurs, et à créer ces prototypes simples. Mais, pour ce nouveau produit, il est également le PO, celui qui au bout du compte est responsable du succès du produit. Il y a d'autres PO dans l'entreprise d'Eric, mais ils n'ont pas ses compétences en conception, et ils s'associent judicieusement à des designers pour interroger en binôme les utilisateurs et envisager des solutions.

© Dunod – Toute reproduction non autorisée est un délit.

Eric a finalement présenté des prototypes aux utilisateurs. Comme je n'étais pas là, je ne sais pas ce qui s'est réellement passé. Mais j'ai vécu cette situation de très nombreuses fois, et je suis toujours surpris de ce que j'apprends de la part des gens qui vont vraiment utiliser ma solution. Tout ce que je peux vous dire, c'est que vous devez vous préparer à des surprises et à des mauvaises nouvelles. Et vous devez vous réjouir de ces mauvaises nouvelles, car vous auriez pu les recevoir quelques mois plus tard, après avoir développé le logiciel. Et à ce moment-là, c'est vraiment pénible. Au stade du prototype, apporter des modifications ne coûte pas trop cher et c'est à ce moment que vous devriez vous en occuper. C'est d'ailleurs ce qu'Eric a fait.

> Prototypez et testez votre solution avec les utilisateurs pour savoir si elle est valable et utilisable.

Après plusieurs itérations de sa solution et après l'avoir présentée plusieurs fois à ses clients, Eric était convaincu qu'il tenait une assez bonne idée de solution. Il pouvait maintenant créer un backlog et mettre au travail son équipe de développeurs pour transformer cette solution prototypée en un véritable logiciel fonctionnel. Mais il ne le fit pas car il avait encore un défi plus important à relever.

3.4 ATTENTION À L'EXPRESSION DES BESOINS DES UTILISATEURS

Eric a prototypé ce qu'il croit être une solution viable, mais il n'est pas vraiment sûr qu'elle soit minimale, car il a présenté aux clients beaucoup d'idées séduisantes. Et par définition, quand vous montrez aux gens des idées séduisantes, ils les adorent. Eric sait cependant que son travail consiste à réduire la quantité à développer tout en maintenant la satisfaction des clients. Combien d'idées séduisantes peut-il écarter tout en gardant une solution viable ?

Eric sait aussi quelque chose d'autre qui est un peu troublant : les gens qui ont dit qu'ils aimeraient certaines fonctionnalités et les utiliseraient ne font en fait que des suppositions.

Pensez à ce qui se passe quand vous achetez vous-même quelque chose. Vous avez sans doute étudié le produit, regardé un vendeur faire une démonstration, testé vous-même des fonctions séduisantes, et vous vous êtes vraiment imaginé en train d'utiliser le produit et l'apprécier. Mais quand vous l'avez acheté et que vous avez commencé à l'utiliser, vous avez trouvé que les fonctions séduisantes n'étaient pas si importantes que cela. Ce qui importait vraiment, c'étaient les caractéristiques auxquelles vous n'aviez pas pensé. Et, pire encore, il n'était peut-être pas vraiment nécessaire d'acheter ce produit. Je décris peut-être mon cas personnel, mais j'ai beaucoup de choses dans mon garage que je regrette d'avoir achetées.

Mais revenons à Eric. Il sait que ses clients et ses utilisateurs peuvent imaginer que le produit sera utile, mais la véritable preuve existera quand les gens choisiront de l'utiliser réellement tous les jours. C'est le véritable outcome qu'il recherche et le seul outcome dont son entreprise tirera parti. Et il va falloir plus qu'un prototype pour le découvrir.

© Dunod – Toute reproduction non autorisée est un délit.

3.5 CRÉEZ POUR APPRENDRE

C'est maintenant qu'Eric va pouvoir montrer toute son intelligence.

Le premier objectif d'Eric et de son équipe n'est pas de créer un produit viable minimal, mais en fait de concevoir quelque chose de minimaliste qui soit juste suffisant pour que les utilisateurs potentiels puissent s'en servir. Il s'agit d'un produit qui ne va pas impressionner les foules et que certains pourraient même détester. Ce n'est donc certainement pas un produit que vous souhaiteriez avoir à votre catalogue. En fait, les seules personnes à qui vous voulez montrer ce produit sont celles qui l'utiliseront éventuellement un jour et qui sont à la recherche d'un logiciel permettant de résoudre leur problème.

Il se trouve qu'Eric a un petit groupe de personnes qui correspondent à ce profil. Ce sont les clients et les utilisateurs avec lesquels il a travaillé plus tôt quand il validait leur problème. Ce sont ses partenaires de développement qui ont formulé des commentaires sur les premiers prototypes. Et il existe un sous-ensemble de ces personnes qu'Eric croit susceptibles de mieux l'aider dans son apprentissage. Ce sont ceux à qui il va présenter sa version de produit minimaliste, et certainement pas viable, et il espère en prime qu'ils vont l'adopter tout de suite.

Cette photo montre Eric pointant une tranche de son backlog. Quand elle a été prise, il avait déjà livré le logiciel à ses partenaires de développement. Après cela, il a recueilli leurs commentaires. Son équipe a également intégré quelques indicateurs simples qui permettent de mesurer si les gens utilisent vraiment le logiciel et ce qu'ils effectuent dans le programme.

Eric sait que les gens sont polis et qu'ils peuvent prétendre aimer un produit sans jamais l'utiliser. L'usage est le véritable outcome qu'il recherche et la politesse ne lui sera d'aucun secours. Eric sait aussi que certaines personnes sont exigeantes. Elles peuvent énumérer tous les problèmes du produit, ou se plaindre des bugs, mais les indicateurs peuvent nous dire qu'elles utilisent quand même tous les jours le produit. Et c'est une bonne chose, en dépit de toutes leurs plaintes. Le mécontentement est positif parce qu'il donne à Eric des idées pour ses prochaines améliorations.

Le backlog d'Eric est organisé sous la forme d'une story map avec une colonne vertébrale composée de Post-it® jaunes dans la partie supérieure. Ces Post-it® jaunes comportent des phrases verbales courtes qui racontent la grande story des usages du produit, mais à un niveau élevé. En dessous, on trouve tous les détails, les choses spécifiques qu'ils font et dont ils ont vraiment besoin pour utiliser le produit. Alors que les détails sur lesquels lui et son équipe travaillent changent d'une version à l'autre, la colonne vertébrale reste assez cohérente.

La tranche supérieure, au-dessus de la ligne délimitée par un ruban adhésif, est celle sur laquelle Eric et son équipe travaillent. Cette release prendra à Eric deux sprints. Il utilise un processus de développement Scrum dans lequel ses sprints sont de deux semaines, si bien que deux sprints équivalent à un mois. Au-dessous de la ligne, se trouvent les autres tranches du tableau. La tranche suivante contient ce qui est censé être la release suivante et ainsi de suite. À gauche de chaque tranche, tout comme c'était le cas avec l'équipe de Globo.com, il y a un Post-it® avec le nom de la release et quelques mots sur ce que l'on veut apprendre dans cette release. À l'exception de la release du haut qui contient une BD de *Dilbert*, mais c'est une blague que leur équipe est la seule à comprendre…

Si vous regardez attentivement, la partie supérieure de cette tranche a été nettoyée. J'ai dessiné quelques Post-it® à l'endroit où ils sont habituellement, mais ils n'y sont plus parce que ces choses qui sont au-dessus sont les premiers éléments que l'équipe crée. Comme les membres de l'équipe ont travaillé ensemble pour planifier leur travail, ils ont enlevé ces Post-it® et les ont placés sur un tableau des tâches à droite du backlog avec des story maps. Ce tableau montre les stories sur lesquelles ils travaillent à présent dans ce sprint, ainsi que des tâches de livraison, ces choses spécifiques que les développeurs et les testeurs doivent faire pour transformer les idées de la story en un logiciel qui fonctionne.

L'un des points les plus intéressants du backlog d'Éric, qui prouve qu'il est intelligent, est l'épaisseur de la tranche supérieure (elle est deux fois plus épaisse que les tranches inférieures). Quand Eric et son équipe terminent une tranche et livrent à leurs partenaires de développement ce qu'ils appellent une *beta client*, ils remontent les Post-it® de la tranche en dessous. Quand ils le font, ils ont beaucoup de discussion sur les détails de cette release suivante. Ils vont jouer au « *Que se passe-t-il si ?* » pour trouver des problèmes, puis renseigner les détails. Ils évoqueront certaines des idées de la release suivante, et cette discussion peut entraîner le fractionnement d'une grande idée en deux ou trois petites idées. Puis, ils auront besoin de la hauteur verticale de cette tranche pour établir des priorités, c'est-à-dire faire des choix sur ce qu'il faut créer en priorité.

Vous voyez à présent comme ils sont malins ?

3.6 ITÉREZ JUSQU'À LA VIABILITÉ

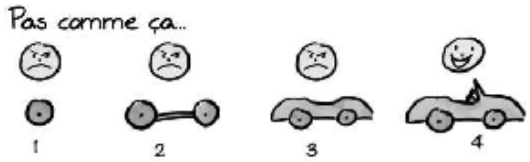
Eric a sans doute démarré ce processus avec une idée en tête de ce que serait le produit viable minimum, mais il a volontairement créé quelque chose de minimaliste pour commencer. Il en rajoute ensuite un peu plus chaque mois. Il reçoit un feedback de ses partenaires de développement, sous la forme de données subjectives issues de leurs conversations et de choses plus objectives qu'il récolte de la lecture des données.

Il va suivre cette stratégie, en améliorant lentement le produit, jusqu'à ce que ses partenaires de développement commencent à l'utiliser régulièrement. En fait, Eric a l'espoir qu'ils recommandent le produit à d'autres personnes et deviennent de véritables clients de référence. Une fois ce stade atteint, Eric saura qu'il a trouvé le produit minimum *et* viable, qui sera prêt à être commercialisé. Si Eric et son équipe avaient essayé de le vendre avant, ils auraient déçu de nombreux clients, qui auraient été des gens beaucoup moins sympathiques que ceux avec lesquels il a construit des relations personnelles tout au long de ce processus.

3.7 COMMENT SE TROMPER

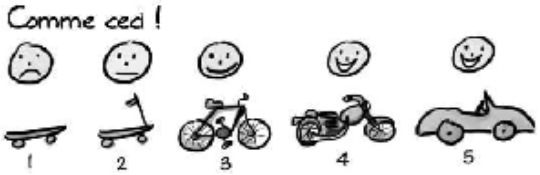
Eric aurait pu procéder autrement et prendre son dernier meilleur prototype, le décomposer en parties élémentaires et commencer à le développer partie par partie. Plusieurs mois plus tard, il aurait eu quelque chose à livrer et il aurait alors su si son hypothèse était juste. Croyez-moi, c'est malheureusement rarement le cas.

© Dunod – Toute reproduction non autorisée est un délit.



Il s'agit d'un schéma simple réalisé par mon ami Henrik Kniberg. Il illustre admirablement une stratégie de release décomposée où, à chaque release, j'obtiens quelque chose que je ne peux pas utiliser, et où il faut attendre la dernière release pour avoir quelque chose de fonctionnel.

Henrik suggère cette stratégie alternative :



Si je planifie mes releases de cette façon, à chaque nouvelle release je livre quelque chose que les gens peuvent utiliser réellement. Cela étant, dans cet exemple de transport simpliste, si mon objectif est de parcourir une longue distance et de transporter des bagages avec moi, et que l'on me donne une planche à roulettes, je vais me sentir un peu frustré. Si le but de cette conception était de me faire plaisir, l'objectif est raté. Mais le but réel est d'apprendre, et c'est ce qui a été réalisé ici car vous avez appris que je voulais me déplacer sur de longues distances.

Dans la progression d'Henrik, les choses commencent à prendre forme avec la release du vélo parce que je peux effectivement l'utiliser comme moyen de transport adéquat. Et avec la moto, je peux *vraiment* voir que cette solution fonctionne pour moi. Cela pourrait constituer le produit minimal et viable. Si j'aime vraiment la moto, peut-être que l'étape suivante serait une Harley-Davidson plus puissante et plus rapide à la place d'une voiture de sport. Mais c'est seulement après avoir essayé la moto, et en avoir tiré tous les deux un enseignement, que nous pourrons prendre la meilleure décision.

> Traitez chaque release comme une expérience et soyez conscient de ce que vous voulez apprendre.

Mais qu'en est-il des autres personnes qui ont besoin de parcourir une distance plus longue et qui ont des enfants ? Pour *ce* marché cible, aucune de ces deux solutions ne serait un choix pertinent.

Gardez toujours à l'esprit vos clients cibles, les utilisateurs et l'outcome que vous espérez car c'est vraiment difficile d'obtenir le même outcome pour tous les types d'utilisateurs.

© Dunod – Toute reproduction non autorisée est un délit.

3.8 APPRENTISSAGE VALIDÉ

Mon ami Eric a appliqué une **stratégie d'apprentissage validé**, qui est un des concepts importants de la méthode Lean Startup. Eric savait que les problèmes qu'il réglait, pour le compte de ses clients et de ses utilisateurs, et les solutions qu'il avait à l'esprit n'étaient que des hypothèses. Certaines d'entre elles étaient plutôt bonnes, mais elles n'en restaient pas moins que des hypothèses. Eric a entrepris d'analyser les hypothèses, puis de les valider, en passant des problèmes de ses utilisateurs aux solutions qu'il avait imaginées. À chaque étape, il a réalisé quelque chose avec l'objectif explicite d'en apprendre un peu plus.

Ce qu'a fait Eric est le cœur même de la boucle créer-mesurer-apprendre décrite par Eric Ries. Et selon la définition de Ries, chaque release livrée par Eric était un produit viable minimum. Mais vous pouvez voir que la release n'était pas viable aux yeux de ses utilisateurs cibles, tout du moins pas encore. C'est la raison pour laquelle j'aime me référer au MVP de E. Ries en parlant de MVPe (*minimum viable product experiment* ou expérimentation de MVP). Il s'agit de la plus petite chose que je puisse créer pour apprendre quelque chose. Et ce que j'apprends, c'est la compréhension de ce qui est vraiment viable aux yeux de ma clientèle cible.

Eric a utilisé beaucoup d'outils et de nombreuses techniques au cours de ce processus, mais raconter des histoires à l'aide de mots et d'images a toujours fait partie de sa façon de travailler. Et l'emploi d'une carte pour organiser ses stories l'a aidé à garder en tête ses clients et leur cheminement, au fur et à mesure qu'il améliorait de façon itérative son produit pour en faire un produit viable.

J'aime employer le terme *découverte de produit* pour décrire ce que nous faisons vraiment à ce stade. Notre objectif n'est pas d'obtenir quelque chose de construit, mais plutôt d'apprendre si nous construisons la bonne chose. Il se trouve que créer

quelque chose que l'on présente aux clients est l'une des meilleures façons d'apprendre si nous créons la bonne chose. Ma définition du **concept de découverte** est empruntée à Marty Cagan[1]. Ma définition de la *découverte* englobe les pratiques Lean Startup, les expériences utilisateurs Lean, la pratique du Design Thinking et beaucoup d'autres idées. Et ce que je fais pendant la découverte est en perpétuelle évolution, mais l'objectif reste le même : apprendre aussi vite que possible si je construis la bonne chose.

3.9 RÉDUISEZ AUTANT QUE POSSIBLE LES EXPÉRIENCES

Si nous reconnaissons que notre but est d'apprendre, alors nous pouvons réduire au minimum ce que nous créons et nous concentrer sur la seule création de ce qu'il nous faut apprendre. Si vous faites cela correctement, cela signifie que ce que vous créez au début ne doit pas être prêt à l'emploi. En fait, si c'est le cas, vous en avez sans doute trop fait.

Prenons un exemple : quand j'étais PO d'une entreprise qui développait des logiciels pour les grandes chaînes de détaillants, je savais que mes produits devaient tourner sur une grande base de données Oracle située sur le back end. Malheureusement la collaboration avec les administrateurs de la base de données ne s'est pas toujours très bien passée car ils ont voulu superviser toutes les modifications que j'ai faites, si bien que parfois de simples changements prenaient une semaine, voire plus. Cela nous ralentissait beaucoup trop. Les préoccupations des administrateurs de la base de données étaient légitimes car toutes les autres applications dépendaient de cette base de données et un plantage aurait été vraiment risqué pour tout le monde. Ils avaient un processus bien huilé pour évaluer les modifications à apporter à la base de données, mais cela prenait tout simplement beaucoup de temps.

Pour ma part, le plus risqué était de m'assurer que mon produit était correct si bien que nous avons créé les premières versions du logiciel à l'aide de bases de données simples. Bien entendu, elles n'auraient pas supporté le passage à l'échelle, et nous ne pouvions jamais livrer nos premières versions à un large public. Cependant, nos premières expériences de produit viable minimum nous ont permis de tester nos idées avec un petit sous-ensemble de clients tout en continuant à utiliser des données réelles. Après plusieurs itérations avec les clients, et après avoir trouvé une solution que nous pensions fonctionnelle, nous avons effectué les modifications de la base de données et migré notre application. Les administrateurs de la base de données nous aimaient bien aussi, parce qu'ils savaient que lorsque nous faisions des modifications, nous étions convaincus qu'elles étaient les bonnes.

1. Marty a décrit pour la première fois ce qu'il entend par « découverte de produit » dans un essai en 2007 (http://www.svpg.com/product-discovery). Il a par la suite complété sa description dans son livre *Inspired: How to Create Products Customers Love* (SVPG Press).

3.10 RÉCAPITULONS

Gary a utilisé une carte pour se sortir du piège du backlog plat et obtenir une vision d'ensemble de son produit, puis pour vraiment commencer à se concentrer sur ce qu'il devait faire.

Les équipes de Globo.com ont utilisé une carte pour coordonner un grand projet impliquant plusieurs équipes et décomposer un sous-ensemble de travail qu'ils considéraient comme une solution viable.

Eric a utilisé une carte pour décomposer des releases minimalistes en MVPe, ce qui lui a permis de manière itérative de trouver ce qui serait viable.

Il y a un dernier défi pour ce qui semble être la plaie du développement de logiciels : le respect des délais. Supposons que vous soyez persuadé de l'intérêt de développer un produit et que d'autres personnes dépendent de la date de livraison de ce produit. Il existe un secret, connu depuis des siècles par les artistes, pour achever un travail à temps. Dans le prochain chapitre, nous apprendrons à appliquer ce secret au logiciel.

4

Planifiez pour finir à temps

Eux, ce sont Aaron et Mike. Ils travaillent pour une société appelée Workiva qui commercialise une suite de produits sur une plateforme nommée Wdesk. Workiva résout des problèmes importants pour de grandes entreprises, et c'est l'une des plus grandes sociétés de SAS (*Software As a Service*) dont vous n'avez probablement jamais entendu parler.

Aaron et Mike ont l'air heureux, n'est-ce pas ? C'est typique chez les personnes qui ont travaillé ensemble à résoudre des problèmes difficiles. Ou serait-ce parce que le type sur la droite a une bière à la main ? Non, ce n'est pas cela, mais bien le sentiment d'avoir résolu un problème difficile qui les rend heureux. La bière est juste une récompense. Si *vous* n'obtenez pas de bière, ou de récompense équivalente, quand vous résolvez des problèmes difficiles dans votre travail, vous devriez en parler à quelqu'un.

© Dunod – Toute reproduction non autorisée est un délit.

Aaron et Mike viennent d'achever plusieurs cycles de découverte du produit, et ils sont convaincus qu'ils tiennent quelque chose qui mérite d'être créé et de partir en production.

Et c'est en commençant à formuler les idées de fonctionnalités qu'ils ont vraiment commencé à comprendre à qui ce produit était destiné et pourquoi ils allaient le développer. Puis ils ont cherché à valider leurs hypothèses en interrogeant directement leurs clients sur leurs méthodes de travail et sur leurs véritables problèmes. Après cela, ils ont créé des prototypes simples. Aaron et Mike ont pu développer des prototypes numériques simples avec Axure, puis les tester avec les clients à distance ; cela a d'abord permis de voir s'ils étaient conformes à la solution, puis de s'assurer qu'ils étaient utilisables. Pour la feature sur laquelle ils travaillaient, ils n'ont pas jugé utile de créer un prototype logiciel fonctionnel pour apprendre ce dont ils avaient besoin.

Après plusieurs itérations avec des prototypes simples, ils ont été finalement convaincus qu'ils avaient quelque chose qui méritait d'être créé. Cela peut paraître un très gros travail, mais ça n'a en fait duré que trois jours. Leur dernière étape consistait à créer un backlog et un projet pour fournir la feature. C'est leur projet que l'on voit sur la photo, et comme c'est un bon projet, ils sont heureux.

Il est important de noter que cette carte ne représente pas un produit entier, mais simplement une feature qu'ils ajoutent à un produit existant. C'est la raison pour laquelle la carte est plus petite que celle de Gary au chapitre 1 ou bien que celle des équipes de Globo.com. Je vous dis cela parce que certaines personnes croient à tort qu'elles ont besoin de cartographier la totalité de leur produit pour faire une petite modification, et elles utilisent ce prétexte pour ne pas faire de story mapping.

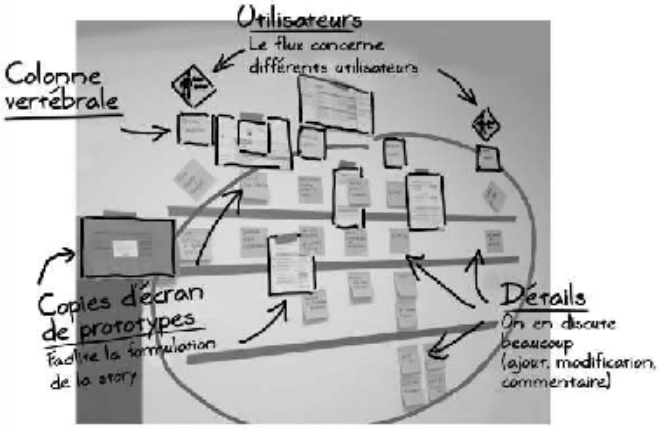
> Cartographiez uniquement ce dont vous avez besoin pour commencer vos conversations.

4.1 COMMUNIQUEZ AVEC L'ÉQUIPE

Pour développer cette nouvelle feature, Aaron et Mike vont devoir forger une compréhension commune avec leur équipe. Cette équipe doit être en mesure de signaler les problèmes et les possibilités d'amélioration, ainsi que d'estimer combien de temps tout cela va prendre. C'est la raison d'être de la création de cette carte qui décrit chaque story de la feature, étape par étape, du point de vue de l'utilisateur. Avez-vous remarqué les copies d'écran intégrées à la carte ? Quand ils expliquent la carte à leur équipe, ils montrent du doigt les copies d'écran et les détails mis en évidence si bien que ceux qui les écoutent peuvent mieux se représenter la solution. Les gens de chez Disney qui conçoivent des films à l'aide d'un storyboard n'ont rien à leur envier.

Lorsque les membres de l'équipe leur ont demandé pourquoi l'écran se comportait de telle manière, ils disposaient de stories pour décrire les variantes qu'ils avaient testées et le comportement des utilisateurs. Lorsque l'équipe pose des questions précises sur ce qui se passe exactement lorsque les données sont saisies ou des informations validées, ils peuvent apporter des réponses.

© Dunod – Toute reproduction non autorisée est un délit.



Quand ils ne savent pas, ils en discutent avec l'équipe et rédigent des commentaires sur les prototypes ou sur les Post-it® du modèle. Ils peuvent même ajouter quelques Post-it® pour des détails auxquels ils n'avaient pas pensé, mais que leur équipe leur a suggérés. Aaron m'a avoué que l'équipe avait repéré plusieurs dépendances techniques que lui et Mike n'auraient jamais trouvées.

4.2 LE SECRET D'UNE BONNE ESTIMATION

Toute personne qui travaille dans le secteur du développement de logiciels depuis un certain temps sait que l'un des plus grands défis consiste à estimer la durée d'un développement. Je vais vous confier un des secrets les mieux gardés d'une bonne estimation :

> Les meilleures estimations proviennent des développeurs qui comprennent vraiment ce qu'ils ont à estimer.

Il y a de nombreuses méthodes qui promettent de donner des estimations plus précises, et je ne traiterai d'aucune d'entre elles ici. Vous pouvez cependant être certain qu'aucune méthode ne marche si les développeurs du logiciel n'améliorent pas leur compréhension mutuelle et leur communication avec ceux qui ont imaginé le logiciel.

Le fait de partager une compréhension mutuelle pour bien estimer ne devrait pas être un secret bien gardé et vous devriez tout de suite aller le répéter largement autour de vous.

4.3 CRÉEZ PIÈCE PAR PIÈCE

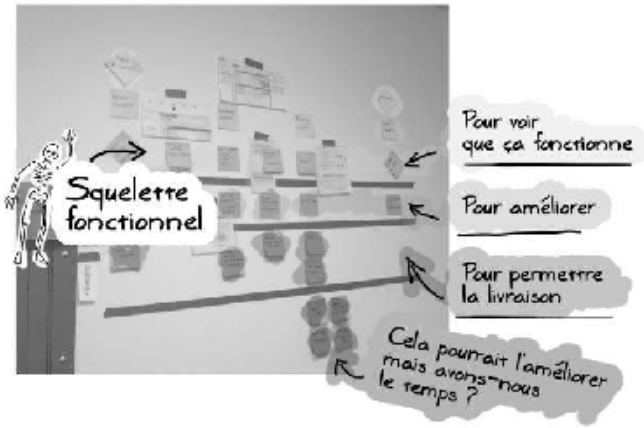
À ce stade, l'équipe de Workiva ne peut plus vraiment s'en sortir en développant moins que ce minimum. Ils ne peuvent faire ce que Globo.com a fait au chapitre 2 et décomposer les choses, parce qu'ils ont déjà validé qu'ils avaient besoin de tout cela. Quand ils faisaient du prototypage, ils avaient la possibilité d'abandonner beaucoup de fonctionnalités tout en vérifiant que leur solution était toujours utile à leurs clients. Mais, quand vous regardez leur carte, elle est découpée en trois tranches.

Vous pourriez demander en quoi cela constitue un problème ; mais livrer un tiers de ce que les clients demandent, c'est un peu comme fournir un tiers d'une voiture de sport : personne ne peut la conduire. Mike est le PO et il ne va pas changer d'avis après avoir identifié une bonne solution. Son rôle est à présent modifié et il est un peu comme un réalisateur dans un film : il doit être présent lorsque chaque scène est tournée. Et il doit décider quelles scènes doivent être tournées d'abord, et quelles scènes doivent être tournées ensuite. Il sait qu'à la fin toutes les parties du film doivent être réunies et ressembler à un tout cohérent.

Mike a donc travaillé avec son équipe pour créer un plan de développement. C'est ce qu'ils ont fait en découpant leur carte en trois tranches horizontales.

La première tranche englobe toute la feature. Une fois qu'ils ont créé toutes les pièces, ils peuvent la voir fonctionner d'un bout à l'autre. Elle ne couvre pas toutes les situations possibles, et si elle était livrée aux utilisateurs dans cet état, ceux-ci seraient mécontents. Mais Mike et son équipe sont en mesure d'avoir un morceau de logiciel qui fonctionne de bout en bout. Ils peuvent y injecter des données réelles pour voir comment il se comporte, et ils peuvent exécuter certains outils de tests automatisés pour voir comment il supporte la montée en charge. Ils peuvent en apprendre beaucoup sur les risques techniques qui pourraient leur causer des ennuis par la suite. Au fur et à mesure qu'ils avancent, ils gagnent en sérénité et pensent qu'ils pourront livrer le logiciel dans les délais. Ils sont tout au moins capables de repérer les problèmes imprévus qui pourraient les ralentir.

© Dunod – Toute reproduction non autorisée est un délit.



J'appelle cette première tranche un **squelette fonctionnel**, terme que j'ai emprunté à Alistair Cockburn. J'ai aussi entendu d'autres personnes appeler cela un « fil d'acier » (*steel thread*) ou une « balle traçante » (*tracer bullet*).

Ils vont s'appuyer sur la deuxième tranche pour compléter la fonctionnalité, afin de s'approcher d'une version livrable. En chemin, ils sont susceptibles d'apprendre des choses qu'ils ne pouvaient pas prévoir. Ils peuvent avoir négligé certaines caractéristiques que cette fonctionnalité devait intégrer, comme des subtilités qui n'ont pas été explorées dans le prototype. Ils peuvent aussi trouver que le système ne se comporte pas comme ils s'y attendaient, et qu'un travail supplémentaire devra être effectué pour obtenir la vitesse désirée. Il s'agit des « imprévus prévisibles », un concept étroitement lié aux « inconnues inconnues » de Donald Rumsfeld[1]. Ne faites pas semblant de croire qu'ils n'existent pas car vous savez que c'est faux.

Enfin, ils vont s'appuyer sur la troisième tranche pour affiner la fonctionnalité et la rendre aussi parfaite que possible. Ils en profiteront également pour rajouter certaines de ces choses imprévisibles.

4.4 NE LIVREZ PAS CHAQUE TRANCHE

Chacune de ces tranches ne constitue pas une version destinée aux clients et aux utilisateurs : c'est une étape importante pour les membres de l'équipe qui en profiteront pour s'arrêter et faire le point. Du point de vue des utilisateurs et des clients, le produit est incomplet et il vaut donc mieux éviter les ennuis.

1. Jeff Patton fait ici référence à un célèbre discours prononcé le 12 février 2002 par Donald Rumsfeld où il a tenté de prouver que le gouvernement irakien de l'époque fournissait des armes de destruction massive aux groupes terroristes.

L'équipe de Mike et d'Aaron a estimé que cette feature allait prendre environ deux mois de travail. Comme Eric, ils ont des sprints de deux semaines, et il leur faudrait donc quatre sprints. Je suppose qu'ils auraient pu faire quatre tranches, une pour chaque sprint, mais ils ont pensé différemment. Il faut considérer ces tranches comme trois compartiments différents avec des objectifs d'apprentissage différents, et décider dans quels sprints iront les tranches.

4.5 L'AUTRE SECRET D'UNE BONNE ESTIMATION

Il y a une autre chose qui semble être un secret, mais qui ne le devrait pas, c'est que les estimations sont... des approximations. Allez sur le Web et cherchez n'importe quelle liste d'oxymores. Je suis certain que vous trouverez l'expression suivante : **estimation précise**. Si l'on savait exactement combien de temps prendraient les choses, alors on ne parlerait pas d'*estimation*, n'est-ce pas ?

Mais si vous développez de petits bouts de logiciel, vous allez avoir une idée assez précise du temps qu'il faudra pour les écrire. On appelle cela une **mesure**, et c'est un peu plus précis.

Voici donc l'autre secret : plus vous mesurez souvent, meilleure sera votre prévision. Si vous vous rendez au travail tous les jours, je suppose que vous avez une assez bonne idée de la durée de votre trajet. Si je vous demande combien de temps il faut pour se rendre à une adresse voisine de votre lieu de travail, je parie que vous pourrez prédire le temps de trajet avec une précision d'une dizaine de minutes. C'est la manière dont on fait une estimation.

En décomposant de grandes choses en éléments plus petits, on multiplie les occasions de mesurer. Il y a bien sûr plus de subtilité dans tout cela, mais en règle générale, on obtient de meilleures prévisions si on a plus d'exemples de temps de développement de choses semblables.

En tant que PO, Mike est finalement responsable de la livraison de la feature dans les délais. Comme c'est un bon PO, il aide tout le monde dans sa petite équipe à s'approprier cet objectif. Il traite les premières estimations comme son budget de livraison.

4.6 GÉREZ VOTRE BUDGET

Mike et Aaron ont travaillé dès le début avec des développeurs en qui ils ont confiance pour obtenir une première estimation du temps de développement. Ils traitent cette estimation comme un budget qu'ils suivent scrupuleusement.

Ils peuvent mesurer le temps imparti à chaque tâche réalisée par l'équipe, et ils traitent ce qui a été créé comme une dépense dans leur budget. Ils peuvent estimer qu'ils ont dépensé la moitié de leur budget, mais qu'ils ont seulement développé un tiers de la feature. Ils ne s'attendaient certainement pas à ça, mais maintenant qu'ils en sont conscients, ils peuvent faire quelque chose pour y remédier. Ils pourraient ponctionner du budget d'autres features sur lesquelles ils travaillent. Ou bien ils pourraient modifier légèrement la feature sans pour autant que les utilisateurs en pâtissent considérablement. Sinon, ils pourraient prendre leurs responsabilités et voir comment modifier les attentes des utilisateurs à qui ils ont promis quelque chose.

Selon la difficulté de la situation, il est possible que tout le monde ait besoin de plus de bière.

Lors de l'analyse de la stratégie de développement, ils vont chercher à s'attaquer dès que possible aux choses susceptibles de faire exploser leur budget. C'est grâce à des conversations avec toute l'équipe qu'ils peuvent repérer ces risques.

Exposition des risques dans la story map

Chris Shinkle, SEP

Une grande entreprise de sécurité veut créer un système de contrôle d'accès sans fil, de milieu de gamme, pour des bâtiments de taille moyenne (par exemple, des écoles, des cabinets médicaux, etc.). La société a engagé la SEP pour développer le firmware des serrures ainsi que la passerelle ZigBee sans fil qui communiquera avec ces serrures.

Le projet était techniquement motivant, mais il comportait tous les ingrédients menant à un échec, notamment un budget étriqué, un échéancier serré, des changements dans l'encadrement à mi-parcours, une technologie non éprouvée et une inflation des demandes des clients.

Bien évidemment, les choses ont commencé rapidement à se dégrader. L'équipe du projet n'avait pas respecté plusieurs échéances importantes, le client était mécontent et le moral de l'équipe était au plus bas. Au cours d'une rétrospective, l'équipe a découvert que la plus grande cause du décalage de l'échéancier était le travail non planifié, principalement en raison des incertitudes et des risques identifiés. Il fallait que les choses changent.

Comme n'importe quelle équipe d'ingénieurs sensés, ils se sont attaqués à la racine du problème. Leur solution a été de modifier la story map.

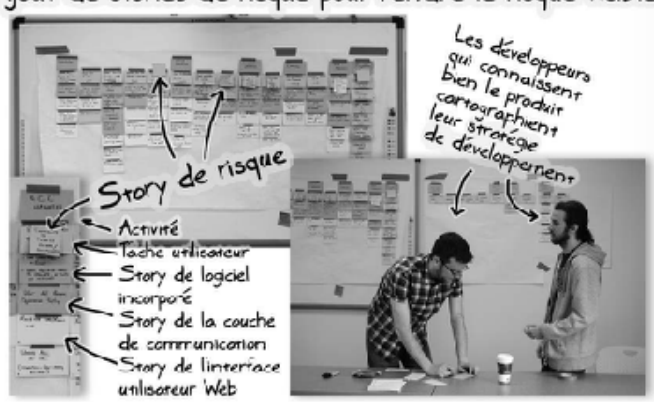
Au niveau supérieur, ils ont augmenté la fréquence et la fidélité de leur story map. En augmentant la fréquence du story mapping à chaque release provisoire, ils espéraient qu'ils augmenteraient la probabilité d'identifier davantage de risques. En augmentant la fidélité de la carte par une intégration des « stories de risque » (en plus des stories sur les activités normales, les tâches et les détails), ils espéraient qu'ils seraient en mesure de visualiser, de discuter et de mieux gérer les risques.

Les résultats ont été étonnants.

L'équipe savait que la largeur et la profondeur d'une story map classique donnent une idée de la taille du projet. Ils savaient également que le nombre de chemins traversant la carte est un bon indicateur de sa complexité. Mais, étant donné que l'incertitude et le risque n'étaient pas auparavant répertoriés sur la story map, la carte ne représentait pas la quantité réelle de travail (y compris l'apprentissage) à effectuer.

La nouvelle carte, avec des stories de risque, a donné une meilleure image de la taille et de la complexité de la tâche. Ces facteurs du projet étaient mieux représentés, car la carte était composée à la fois des stories originales déjà connues ainsi que des nouvelles « stories inconnues », c'est-à-dire les risques ou les connaissances que l'équipe avait besoin d'acquérir pour avancer en toute confiance avec les stories connues.

© Dunod – Toute reproduction non autorisée est un délit.



Comme on pouvait s'y attendre, la story map est devenue beaucoup plus utile pour la planification. Elle met à présent en évidence les risques et les incertitudes qui demanderont du temps à l'équipe. La capacité d'intégrer tout ce temps dans la planification a amélioré les prévisions et la fiabilité de l'équipe.

Parmi les bénéfices annexes, on peut lister une façon concrète de mesurer et de renseigner les parties prenantes sur l'apprentissage. Conjointement avec le traditionnel burndown chart de la feature, l'équipe a inclus un burndown chart du risque. Ce dernier a été particulièrement utile au client pour visualiser le risque quand le burndown chart de la feature n'avait pas fière allure.

L'équipe a donc appris que le fait d'augmenter la fréquence de création des story maps et que l'ajout de nouvelles stories de risque sont de puissants moyens pour que les cartes reflètent mieux la réalité.

4.7 QUE FERAIT LÉONARD DE VINCI ?

Je me pose souvent cette question. Bon, pour être honnête, ce n'est pas vrai, mais je devrais peut-être le faire.

Mike et Aaron ont en fait suivi une stratégie utilisée par les artistes pour finir dans les délais. C'est aussi celle-ci que j'ai utilisée pendant des années avec le développement de logiciel. Et quand j'ai rencontré mes amis de chez Globo.com, c'est encore celle qu'ils ont employée, car comme je l'ai déjà mentionné, ce n'est pas parce qu'ils sont en retard avec leurs nouveaux trucs interactifs pour les Jeux olympiques, que le Comité olympique va décaler les JO. Je suppose d'ailleurs que c'est cette stratégie que vous utilisez encore régulièrement, sans même y penser.

Laissez-moi tout d'abord expliquer ce que Léonard de Vinci *ne* fait *pas*, contrairement à ce que malheureusement les développeurs de logiciel tentent trop souvent de faire.

© Dunod – Toute reproduction non autorisée est un délit.

Supposons que vous soyez Léonard de Vinci, et que vous vouliez créer un tableau en travaillant de la même manière qu'une équipe de développement un peu simpliste[1]. Vous pourriez commencer par ce que vous croyez être une vision claire de la scène que vous avez en tête. Ensuite, vous décomposez la toile en différentes parties. Admettons que vous ayez cinq jours pour peindre ce tableau et que chaque jour vous peignez une partie ; à la fin du cinquième jour, vous avez terminé le tableau. Peut-il y avoir méthode plus simple ?

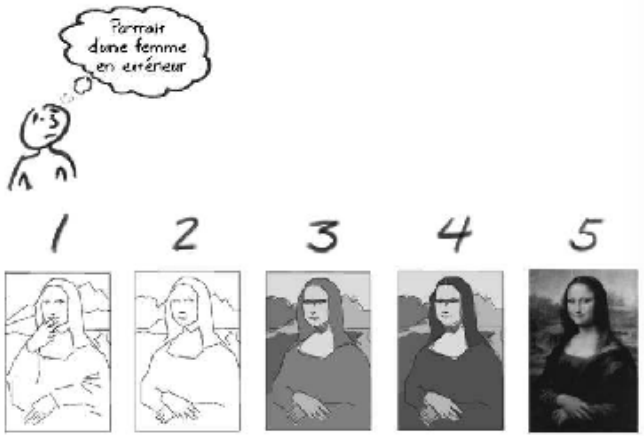
Mais cela ne fonctionne pas comme ça, du moins pas pour les artistes. Cette manière de créer des choses suppose que notre vision soit correcte et précise. Cela suppose également que le créateur sache définir avec précision les différentes parties sans les voir dans leur contexte. Si vous faites cela en développement logiciel, ça s'appelle une **stratégie incrémentale**. C'est la façon dont un maçon construit un mur et ça marche si chaque partie est dimensionnée aussi régulièrement et définie aussi précisément qu'une brique.

Lorsque j'étais enfant, j'avais l'habitude de dessiner et je tombais dans ce piège. Je dessinais souvent une sorte d'animal et je commençais par la tête. Je travaillais sur cette partie jusqu'à ce qu'elle soit parfaite, puis je continuais avec le reste du corps, les pieds, la queue et ainsi de suite. Au moment de terminer le dessin, je me rendais compte que les proportions de mon animal n'étaient pas bonnes : la tête était trop grande ou trop petite par rapport au reste du corps.

C'est bien plus tard que j'ai appris que j'aurais mieux fait de créer d'abord une esquisse de la totalité de la composition. À partir de là, il était facile d'obtenir les bonnes proportions et d'apporter des modifications à la pose de la créature.

Je n'étais pas présent quand il a peint la Joconde, mais je suppose que Léonard de Vinci a fait quelque chose dans ce goût-là.

1. J'ai eu l'idée de cette représentation à la lecture d'un article de John Armitage paru en 2004, « Are Agile Methods Good for Design ? ». John décrit de cette façon le concept de UX design et je suggère que nous filions la métaphore pour la manière dont nous concevons les logiciels.



Même Léonard de Vinci aurait probablement reconnu que sa vision n'était pas parfaite, et qu'il avait appris quelque chose au fur et à mesure qu'il peignait. Le premier jour, j'imagine qu'il aurait tout d'abord créé une esquisse de la composition, ou peut-être une sous-couche légère. Je peux l'imaginer à ce moment-là apporter des modifications à sa composition : « *Je pense que le sourire va être un élément important et je vais éloigner sa main de sa bouche ; et puis ces montagnes en arrière-plan, ça fait beaucoup trop chargé* ».

En milieu de semaine, Léonard ajoute beaucoup de couleurs et de formes à la toile, mais il peut encore faire des modifications. À la fin de la semaine, il sait qu'il est à court de temps, et tous ses efforts se concentrent donc sur l'amélioration de la peinture. On peut se demander si les sourcils qui manquent à *Mona Lisa* sont un choix délibéré ou si Léonard de Vinci a simplement manqué de temps pour ajouter une nouvelle feature...

> « *Un chef d'œuvre n'est jamais terminé, il est seulement abandonné.* »
> Léonard de Vinci

Cette citation, qui est attribuée à Léonard de Vinci, avance l'idée que nous pourrions toujours continuer à ajouter des fonctionnalités et à améliorer le produit, mais qu'à un moment donné, il faut livrer ce produit. L'œuvre de Léonard de Vinci et celles de beaucoup d'autres artistes constituent de bons exemples car les gens qui apprécient l'art n'ont aucune idée que les œuvres ont été abandonnées puisqu'elles leur semblent achevées.

4.8 ITÉRATIF ET INCRÉMENTAL

Un artiste ou un auteur fonctionne de cette façon. En fait, les gens qui publient le journal du matin ou les nouvelles du soir travaillent aussi de cette façon. Toute personne qui doit respecter un délai et apprendre au fur et à mesure se reconnaît dans cette stratégie.

> Utilisez une pensée itérative pour évaluer et apporter des modifications à ce que vous avez déjà fait.

Dans le développement de logiciels, **itérer** a deux significations. Du point de vue d'un processus, cela signifie qu'il faut répéter le même processus à plusieurs reprises. C'est pourquoi la timebox de développement utilisée dans les méthodes agiles est souvent appelée *itération*. Cependant, lorsque vous utilisez ce terme pour décrire ce que vous faites avec le logiciel que vous développez, cela signifie que vous voulez l'évaluer et le modifier. Et la modification d'un logiciel après son développement est trop souvent considérée comme un échec. C'est à ce moment-là que des termes comme *mauvaises spécifications* ou *dérives* sont utilisés pour blâmer les gens qui ont pris des décisions sur ce qu'il fallait développer. Mais nous savons pourtant tous que le changement est un résultat nécessaire de l'apprentissage.

> Utilisez une pensée incrémentale pour faire des ajouts.

Malheureusement, nous pouvons facilement nous faire piéger par le tourbillon des itérations. Nous devons donc garder un œil sur le calendrier et continuer à ajouter des fonctionnalités progressivement. L'artiste réalise des ajouts non seulement en ajoutant de nouvelles choses à un tableau, mais aussi en développant des choses qui étaient déjà présentes.

Vous pourriez faire la même chose avec un logiciel en créant d'abord une version simple de la fonctionnalité sans fioritures. Vous pourriez voir cela comme une sorte d'esquisse. Après avoir utilisé la version simple, vous pourriez l'améliorer en ajoutant davantage de fonctionnalités. Au fil du temps, cela constituerait la version finale que vous aviez initialement envisagée. Si les choses se passent vraiment bien, le résultat final sera différent par rapport à ce que vous aviez prévu initialement, mais il sera meilleur parce qu'il bénéficie de ce que vous avez appris.

4.9 STRATÉGIES D'OUVERTURE, DE MILIEU ET DE FIN DE PARTIE

Même si cela vous fait un peu mal à la tête, je vais ici mélanger quelques métaphores. Personnellement, je m'appuie sur une stratégie basée sur une métaphore du jeu d'échecs lors de la création de logiciels. Je suis un piètre joueur d'échecs qui sait

© Dunod – Toute reproduction non autorisée est un délit.

à peine comment déplacer les pièces, à telle enseigne que si j'utilise cette métaphore à mauvais escient, vous n'êtes pas autorisé à m'écrire ni à me corriger. Quelle que soit la taille de mon produit ou de la release de la feature, je préfère décomposer mon backlog de releases en trois groupes :

- **L'ouverture** – On met l'accent sur les caractéristiques essentielles ou sur les étapes de l'utilisateur qui se retrouvent sur l'ensemble du produit. On se concentre sur des choses qui sont techniquement difficiles ou risquées. On fait l'impasse sur les choses en option. On ignore les règles de gestion sophistiquées nécessaires avant de livrer l'application. On développe *juste* assez pour voir le produit fonctionner de bout en bout.
- **Le milieu de partie** – On complète les fonctionnalités. On ajoute des trucs qui prennent en charge les étapes facultatives des utilisateurs. On met en œuvre ces règles de gestion complexes. Si vous avez réalisé correctement votre ouverture, vous serez en mesure de commencer à tester les performances, l'évolutivité et la facilité d'utilisation du produit. Tout cela relève du contrôle de la qualité qui est difficile à mettre en œuvre, mais nous devons être conscients de ces problèmes et multiplier les tests.
- **La fin de partie** – On améliore la release en la rendant plus efficace à utiliser. Comme vous pouvez à présent l'employer avec des données réelles et monter en charge, c'est à ce moment que vous identifierez des possibilités d'amélioration qui étaient difficiles à voir avec un prototype. C'est ici que vous recueillerez les commentaires des utilisateurs que vous pourrez ensuite appliquer.

4.10 DÉCOMPOSEZ VOTRE STRATÉGIE DE DÉVELOPPEMENT À L'AIDE D'UNE CARTE

Si vous avez développé ce que vous croyez être une première version viable à présenter aux clients et aux utilisateurs, travaillez en équipe pour décomposer encore une fois cette première version publique avec des stories d'ouverture, de milieu de partie et de fin de partie. L'équipe de développement du produit est la meilleure candidate pour identifier les risques et les possibilités d'apprendre car elle s'est approprié le projet qu'elle a réalisé de manière collaborative.

C'est ce qu'Aaron et Mike ont fait avec l'aide de l'ensemble de leur équipe de développement. Regardez encore une fois comme ils ont l'air heureux.

4.11 IL S'AGIT DE RISQUES

Dans le chapitre 3, Eric a dû faire face au risque d'identifier le mauvais produit. Il a utilisé une stratégie de découpage des releases, ce qui lui a permis de présenter des produits complets à ses clients.

Dans ce chapitre, Aaron et Mike ont focalisé leur attention sur les risques techniques, ce qui pouvait faire dérailler leur planning de livraison ou bien provoquer l'inflation des coûts de développement. Ils n'ont pas montré ce qu'ils avaient produit à la fin de chaque cycle aux clients et aux utilisateurs parce qu'ils savaient que le résultat n'était pas suffisant. Mais avec leur équipe, ils ont effectué une analyse rigoureuse et ont utilisé ce qu'ils avaient appris pour piloter en toute sécurité le développement de cette feature.

C'est subtil, mais vous avez peut-être retenu au chapitre 2 qu'Eric prenait deux sprints de deux semaines pour développer son expérience suivante de produit viable minimum. Il devait décider quelles choses créer dans le premier sprint et quelles choses créer dans le deuxième sprint. Il a utilisé ce genre de réflexion pour prendre ces décisions : lui et son équipe ont placé les éléments risqués en premier, c'est-à-dire les parties qu'Eric ou son équipe voulaient voir fonctionner le plus tôt possible afin qu'ils puissent les corriger avant de les présenter aux clients.

4.12 ET MAINTENANT ?

Vous avez vu quatre très bons exemples de développement et d'utilisation de cartes à des fins différentes. Il y a beaucoup d'autres façons d'utiliser des cartes que nous explorerons dans les chapitres suivants. Mais avant d'aller trop loin, je veux vous montrer mon truc préféré pour enseigner comment faire du story mapping. Je vous promets que si vous l'essayez, vous allez créer des story maps comme un expert.

© Dunod – Toute reproduction non autorisée est un délit.

5

Vous savez déjà comment faire

Si vous pensez que la création d'une story map est compliquée, laissez-moi tout de suite vous rassurer. En fait, vous avez déjà tous les éléments pour comprendre les concepts de base utilisés pour la création d'une carte. Nous allons travailler en étudiant un exemple tiré de la vie quotidienne. Et, pour faire simple, nous prendrons comme exemple *votre* vie. En cours de route, je donnerai le nom de ces notions importantes que vous comprenez déjà.

Prenez un bloc de Post-it® ainsi qu'un stylo et suivez-moi. Ne vous inquiétez pas, prenez votre temps : je vous attendrai.

Vous êtes prêt ?

5.1 ÉCRIVEZ VOTRE STORY, UNE ÉTAPE À LA FOIS

Fermez les yeux et repensez au moment où vous vous êtes réveillé ce matin. Quelle est la première chose que vous vous rappelez avoir faite ? Maintenant, ouvrez les yeux et notez-la sur un Post-it®. Je vais écrire en même temps que vous. Sur mon premier Post-it®, il y a écrit : « *Appuyer sur le rappel de sonnerie du réveil* ». C'est en général malheureusement le cas, et certains matins difficiles, cela m'arrive même deux ou trois fois.

Maintenant, décollez le Post-it® et posez-le sur la table en face de vous. Ensuite, pensez à la chose suivante que vous avez faite. C'est terminé ? Maintenant, écrivez ce que c'est sur un Post-it® que vous placerez à côté du premier. Puis continuez selon le même principe. Mes deux Post-it® suivants indiquent « désactiver l'alarme » et « tituber jusqu'à la salle de bains ».

© Dunod – Toute reproduction non autorisée est un délit.

Continuez à lister sur des Post-it® toutes les actions réalisées avant votre départ au travail. Je termine généralement ma liste par « monter dans ma voiture » pour partir au travail. Vous devriez en avoir pour trois ou quatre minutes pour écrire tous vos Post-it®.

5.1.1 Les tâches sont ce que nous faisons

Examinez tous les Post-it® que vous avez écrits et vous remarquerez que pratiquement chacun d'eux commence par un verbe. Ces phrases verbales courtes comme « Prendre une douche » et « Se brosser les dents » sont des *tâches*, c'est-à-dire des choses que l'on fait afin d'atteindre un but. Quand on décrit les tâches que les utilisateurs de notre logiciel réalisent pour atteindre leurs objectifs, on les appelle des *tâches utilisateur*. C'est le concept le plus important pour concevoir de bonnes story maps, si l'on fait abstraction de la rédaction des stories. Vous constaterez que sur les story maps presque tous les Post-it® à propos des activités des utilisateurs de votre logiciel emploient ces phrases verbales courtes.

Maintenant, arrêtez-vous une minute et réfléchissez comme cela a été facile : je vous ai demandé d'écrire ce que vous faisiez et, naturellement, votre cerveau a formulé des tâches, ce qui signifie que le concept le plus important est quelque chose de parfaitement naturel.

Ne vous focalisez cependant pas trop sur le mot *tâche*. Si vous êtes un gestionnaire de projet, vous avez remarqué que les plannings de projet sont remplis de tâches. Si vous avez déjà utilisé des stories en développement agile, vous savez que la planification des travaux consiste à écrire toute une série de tâches de développement et de test. Si vous n'êtes ni gestionnaire de projet, ni développeur de logiciels, faites attention lorsque vous utilisez le mot *tâche* parce que d'autres personnes pourraient penser que vous voulez parler d'un type de tâches auquel *ils* sont familiers et ils vous diraient que vous employez mal ce terme.

> Les tâches utilisateur sont les modules de base d'une story map.

Comptez à présent le nombre de tâches que vous avez inscrites.

La plupart des gens écrivent entre 15 et 25 tâches, et si vous en avez écrit plus, c'est merveilleux. Si vous en avez écrit moins, c'est que vous avez une vie simple... Si seulement je pouvais être prêt le matin aussi rapidement ! Mais vous pouvez revenir sur votre liste pour voir si vous n'avez rien oublié.

5.1.2 Mes tâches sont différentes des vôtres

Je suis sûr que cela n'est pas un scoop, mais les gens sont différents les uns des autres. On remarque ces différences dans la façon dont ils choisissent de faire les choses.

Par exemple, certaines personnes ont la motivation et l'autodiscipline nécessaires pour faire de l'exercice presque tous les matins. Si vous avez écrit plusieurs tâches concernant votre exercice physique, vous êtes génial ! J'en suis pour l'instant resté au stade de l'intention.

Certaines personnes ont tout simplement plus de responsabilités à cause de la vie qu'ils mènent. Si vous avez des enfants, je suis persuadé que vous avez noté plusieurs tâches que les personnes sans enfants n'ont pas inscrites. Si vous avez un chien, vous pouvez avoir une tâche ou deux consacrées à votre animal.

Gardez cela à l'esprit lorsque vous pensez aux personnes qui vont utiliser votre logiciel. Ils peuvent avoir des objectifs différents quand ils s'en servent. Ils peuvent l'employer dans des contextes différents qui les obligent à prendre en considération d'autres personnes ou d'autres choses.

© Dunod – Toute reproduction non autorisée est un délit.

5.1.3 Je suis juste plus soucieux des détails

Dans cet exercice, certaines personnes écrivent plus de détails que d'autres. Certains écriront « Faire le petit-déjeuner » alors que d'autres détailleront en « Insérer le pain dans le grille-pain », « Verser un verre de jus de fruit » et « Préparer un smoothie ».

Les tâches sont comme des pierres : si vous prenez une grosse pierre et la frappez avec un marteau, elle se cassera et formera un tas de pierres plus petites. Ces pierres plus petites sont toujours des pierres et c'est la même chose avec des tâches. Je ne sais pas quand une pierre est assez grande pour être appelée un rocher, ou assez petite pour être appelée un caillou, mais il y a une bonne manière de distinguer une grande tâche d'une petite.

Mon ami Alistair Cockburn décrit le concept de *niveau d'objectif* dans son livre *Writing Effective Use Cases* (publié chez Addison-Wesley Professional). Rassurez-vous, nous n'allons pas à commencer à écrire des *use cases*. C'est simplement que le concept est vraiment utile quand on parle de comportement humain.

Alistair utilise la métaphore de l'altitude où le niveau de la mer représente le milieu, et tout le reste se situe au-dessus ou au-dessous du niveau de la mer. Une tâche du niveau de la mer est une tâche que l'on compte terminer avant d'arrêter volontairement de faire autre chose. Avez-vous écrit dans votre liste de tâches « Prendre une douche » ? C'est une tâche du niveau de la mer, parce que vous ne vous arrêtez pas au beau milieu de votre douche pour vous dire : « *Tiens, cette douche traîne en longueur. Je pense que je vais prendre une tasse de café et que je terminerai ma douche plus tard* ». Alistair appelle ces tâches *des tâches de niveau fonctionnel* et il les annote avec une petite vague. Personnellement, je me contente de les appeler tâches.

Les tâches telles que « Prendre une douche » se décomposent en *sous-tâches* plus petites comme « Ajuster la température de l'eau », « Se laver les cheveux », etc.

N'oubliez pas que les gens sont différents, et que vous verrez par conséquent des différences de comportement dans la façon dont ils abordent les tâches. Alistair annote ces tâches avec un petit poisson, parce qu'elles sont au-dessous de la mer.

Enfin, nous pourrions regrouper une série de tâches en une *tâche récapitulative*. Prendre une douche, se raser, se brosser les dents et toutes ces choses que l'on fait le matin après s'être levé pourraient être regroupées en une seule tâche que l'on pourrait appeler « Toilette ».

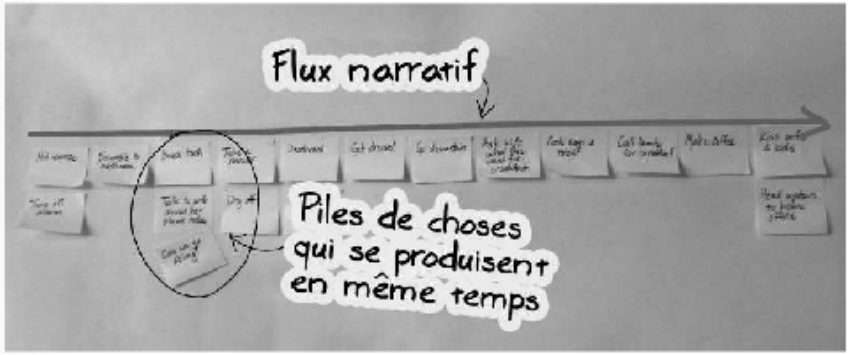
> Utilisez le concept de niveau d'objectif pour vous aider à regrouper les petites tâches ou à décomposer les grandes tâches.

5.2 ORGANISEZ VOTRE STORY

Si vous ne l'avez déjà fait, organisez vos tâches dans un flux de gauche à droite avec ce que vous faites en premier à gauche, et ce que vous faites plus tard à droite.

Essayez de raconter une histoire en pointant le premier Post-it® et en disant « D'abord, j'ai fait ça », puis en pointant le Post-it® suivant, dites « ensuite, j'ai fait cela ». Continuez de la sorte en vous déplaçant de gauche à droite et en racontant votre story. Vous pouvez constater que chaque Post-it® est une étape et que se cache entre chaque Post-it® une petite phrase de conjonction « et puis j'ai... ».

J'appelle cet ordre de gauche à droite *flux narratif*, ce qui est une façon élégante de dire « ordre de la narration ». Nous appellerons cet ensemble une *carte* dont le flux narratif constitue l'axe de gauche à droite.



Mon flux est devenu assez large et j'ai commencé à empiler les choses qui se passent plus ou moins en même temps. Au fur et à mesure que je trace le flux, je constate déjà que j'ai oublié quelques détails, et je me demande s'ils sont importants.

> Les cartes sont organisées de gauche à droite selon un **flux narratif**, c'est-à-dire l'ordre dans lequel on raconte l'histoire.

© Dunod – Toute reproduction non autorisée est un délit.

5.2.1 Renseignez les détails manquants

Le plus intéressant dans cette disposition de Post-it®, c'est que cela permet de voir la story en entier. La vision de la story organisée en un flux narratif permet de voir plus facilement les parties de la story qui manquent.

Examinez votre carte en cours de réalisation et recherchez les étapes que vous auriez pu oublier.

J'en ai ajouté quelques-unes. Il y a beaucoup de détails qui sont au-dessous du niveau de la mer que j'ai finalement décidé d'ignorer. Si j'avais tout pris en compte, il y aurait eu des centaines de Post-it®.

5.3 EXPLOREZ LES STORIES ALTERNATIVES

Jusqu'à présent, tout cela paraît évident et vous n'en avez pas eu pour votre argent. Mais attendez, ça va commencer à devenir intéressant.

Prenez une minute et pensez à ce que vous avez fait *hier* matin. S'il y a des choses différentes de celles que vous avez faites ce matin, écrivez-les et ajoutez-les à votre carte.

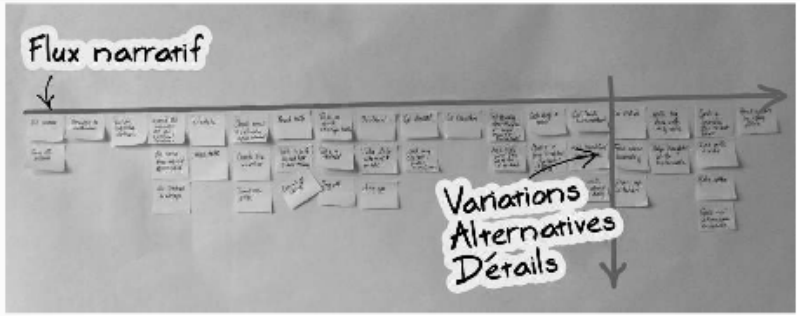
Pensez aux matins difficiles où les choses tournent mal. Que se passe-t-il s'il n'y a pas d'eau chaude ? Que faites-vous alors ? Que se passe-t-il si vous êtes à court de lait, de céréales ou de toute autre chose que vous mangez normalement au petit-déjeuner ? Que faire si votre fille panique parce qu'elle a oublié de faire ses devoirs qui sont à rendre aujourd'hui ? Écrivez les tâches que vous accompliriez dans ces cas-là et ajoutez-les à la carte.

Maintenant, pensez à votre matin idéal. Quelle serait votre matinée parfaite ? En ce qui me concerne, ce serait faire de l'exercice et savourer un copieux petit-déjeuner pendant que je récupère mon retard de lecture. Mais il faudrait alors que je me lève beaucoup plus tôt et que j'arrête de repousser l'heure du réveil.

Notez également que vous allez avoir envie de disposer certaines tâches dans la même colonne, pour économiser de l'espace, mais aussi parce qu'elles paraissent semblables à d'autres tâches qui sont normalement à accomplir. Par exemple, vous pourriez regrouper dans une même colonne les tâches pour faire un copieux petit-déjeuner avec les tâches pour réaliser le petit-déjeuner rapide que vous prenez habituellement le matin.

Mon ami David Hussman appelle cela jouer à « que se passe-t-il ? », une expression que nous avons déjà rencontrée aux chapitres 2 et 3. J'ai ajouté quelques petites choses à ma carte, notamment ce que j'aimerais faire, comme de l'exercice ou un peu de lecture pendant le petit-déjeuner. J'ai aussi ajouté quelques options courantes qui se produisent souvent dans la matinée.

> Les détails, les solutions de rechange, les variations et les exceptions composent le corps d'une carte.



5.3.1 Maintenez le flux

Vous remarquerez que lorsque vous commencez à ajouter ces nouvelles tâches, vous allez sans doute devoir réorganiser votre flux narratif. Je le sais parce que j'ai dû le faire. Par exemple, j'ai eu besoin d'intercaler ma séance d'exercice entre mon lever et ma douche. Et j'ai dû ajouter « Enfiler un survêtement », qui est une étape différente de « S'habiller » après la douche.

Si vous placez les choses calmement là où elles semblent naturelles, vous trouverez un flux narratif qui convient. Quand vous racontez votre histoire maintenant, vous constaterez que vous pouvez le faire de multiples façons différentes. Vous pouvez raconter la journée *classique*, la journée *extraordinaire* et la journée où des cas d'*urgence* se présentent, tout en pointant différents Post-it® pendant que vous déroulez votre récit de gauche à droite. Essayez d'utiliser d'autres conjonctions pour lier vos tâches ensemble. Vous pourriez dire « la plupart du temps, je fais ceci, mais parfois je fais cela » ou bien encore « je fais ceci, ou cela et puis cela ».

Quand j'étais gamin, il y avait une série populaire de livres pour enfants qui s'appelait *Un livre dont vous êtes le héros*. L'idée était qu'à la fin d'une section, le lecteur bénéficiait de plusieurs choix pour les actions du héros de l'histoire. Pour chaque choix, il y avait un numéro de page et quand vous aviez fait votre choix, vous pouviez aller directement au numéro de page indiqué et continuer la lecture de l'histoire à partir de là. À vrai dire, je n'ai jamais été un fan de ces livres car il m'a toujours semblé que l'on se retrouvait au même endroit, quel que soit le choix effectué. Il n'y avait jamais assez de choix pour constituer une très belle aventure. La carte fonctionne un peu comme ça, mais en mieux. Le nombre de chemins à travers une carte est presque illimité, ce qui n'est pas très éloigné de la réalité quand on pense au nombre de possibilités offertes aux personnes utilisant un logiciel pour atteindre leurs objectifs.

Si vous voulez faire des choses *vraiment* difficiles, faites cet exercice avec vos collègues de travail. Vous en apprendrez beaucoup sur eux et vous aurez du mal à trouver un flux narratif avec lequel tout le monde soit d'accord. Il y a toujours les gens qui prennent leur petit-déjeuner avant la douche, et pour d'autres, c'est le contraire. Et je n'évoquerai même pas le grand débat du brossage des dents avant ou après le petit-déjeuner, ou bien avant et après...

Détendez-vous.

© Dunod – Toute reproduction non autorisée est un délit.

S'il y a un débat, cela signifie probablement que ce n'est pas grave. Par exemple, prendre son petit-déjeuner avant ou après avoir pris sa douche est une question de goût. Adoptez le point de vue de la majorité des gens du groupe. Vous constaterez que les gens ne se disputent pas pour les choses *importantes*. Par exemple, placer la tâche « S'habiller » après la tâche « Prendre une douche » n'est pas seulement une question de préférence. Si vous inversez les tâches, vous irez au travail avec des vêtements mouillés...

5.4 EXTRAYEZ L'ESSENTIEL DE VOTRE CARTE POUR CONSTITUER UNE COLONNE VERTÉBRALE

Maintenant, votre carte devrait être assez large, et si vous avez exploré beaucoup d'options, peut-être même un peu épaisse. Il est probable qu'elle comporte une trentaine de tâches et elle devrait ressembler à la colonne vertébrale d'un animal étrange.

Si vous prenez du recul et balayez votre carte de gauche à droite, vous y trouverez toute une série de stories qui paraissent aller ensemble ; par exemple, toutes ces choses que vous faites dans la salle de bains pour vous laver, ou toutes les choses relatives à la préparation du petit-déjeuner. Pouvez-vous identifier ces groupes de tâches qui paraissent aller ensemble afin de vous aider à atteindre un but plus important ?

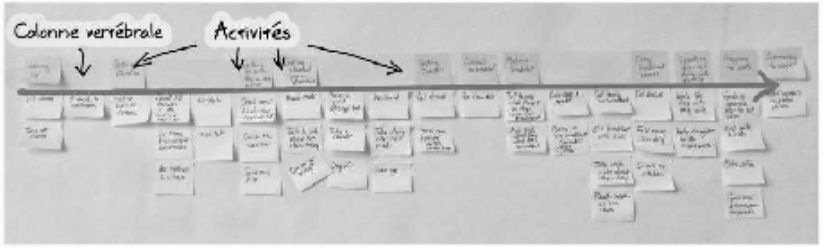
Au-dessus de chacun de ces groupes de Post-it® similaires, placez un Post-it® de couleur différente. Écrivez dessus une phrase verbale courte qui retire l'essentiel de toutes les tâches qui sont en dessous.

Si vous n'avez pas de Post-it® de couleur différente, faites pivoter un Post-it® de 45 degrés et, hop, vous avez une jolie forme de diamant que vous pouvez utiliser pour faire varier l'aspect.

Ces Post-it® avec une tâche d'objectif de niveau supérieur sont appelés *activités*. Les activités organisent toute une série de tâches effectuées par des personnes similaires en même temps afin d'atteindre un objectif particulier. Lorsque vous lisez les activités dans la partie supérieure de la carte, elles représentent aussi un flux narratif. La rangée de Post-it® constitue la colonne vertébrale de la carte. Si vous avez une carte composée de nombreux Post-it® que vous souhaitez partager avec d'autres personnes, un bon moyen de commencer consiste à raconter une histoire de haut niveau. Pour cela, il suffit de lire la colonne vertébrale de la carte, avec les conjonctions « et ensuite ils... » entre chaque activité.

Les activités agrègent des tâches ayant un objectif commun.

Voici ma carte en cours de construction avec les activités que j'ai rajoutées pour constituer une colonne vertébrale. Sa lecture est ainsi plus facile et il est aussi plus aisé d'identifier des choses, tout au moins pour moi. On a également une meilleure vision d'ensemble de ce qui se passe dans ma matinée.



Les activités et les tâches de haut niveau forment la colonne vertébrale d'une story map.

Les activités ne semblent pas avoir de langage commun comme les tâches. Par exemple, comment appelez-vous cette chose que vous faites avant de quitter la maison ? Cette activité où vous rassemblez vos affaires dans un sac, attrapez une liste de courses, consultez la météo et prenez un parapluie au cas où ? Je pourrais l'appeler « rassembler mon fourbi », mais vous pouvez l'appeler comme vous le souhaitez.

Lorsque vous créez votre carte pour les produits de vos clients, vous devez employer la terminologie *qu'ils* utilisent.

5.5 DÉCOMPOSEZ LES TÂCHES QUI VOUS AIDENT À ATTEINDRE UN OUTCOME SPÉCIFIQUE

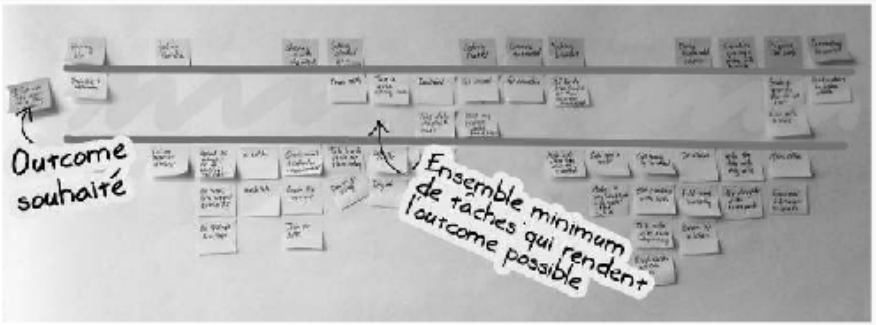
Maintenant, on aborde la partie vraiment intéressante où vous allez utiliser la carte pour vous aider à imaginer quelque chose qui ne s'est pas produit.

Si vous regardez la carte que vous avez créée, vous allez probablement voir « Appuyer sur le rappel de sonnerie du réveil » ou « Désactiver l'alarme » quelque part sur le bord gauche. Imaginez que ce matin vous sautiez ces étapes tout simplement parce que la nuit dernière, vous avez oublié de programmer votre réveil. Vous vous réveillez en sursaut, regardez le réveil et vous vous apercevez que vous êtes terriblement en retard. Ne paniquez pas, c'est juste un exercice.

Écrivez sur un Post-it® « Se préparer en quelques minutes » et placez-le à gauche de la carte vers le haut. Maintenant, imaginez une ligne découpant le milieu de la carte de gauche à droite, un peu comme une ceinture. Déplacez à présent toutes les tâches en dessous de cette ligne si elles ne sont pas nécessaires pour atteindre l'objectif de vous préparer en quelques minutes. Ne déplacez pas les activités vers le bas, même s'il n'y a plus aucune tâche en dessous. S'il y a une activité qui ne contient plus aucune tâche, cela indique que vous n'allez pas atteindre cet objectif ce matin.

Il va probablement ne rester que quelques tâches dans la tranche supérieure. Revenez maintenant au flux et remplissez les tâches qui manquent et qui sont nécessaires si vous êtes en retard.

© Dunod – Toute reproduction non autorisée est un délit.



Par exemple, en temps normal, vous prenez une douche, mais lorsque vous êtes en retard vous remplacez les tâches similaires par « S'asperger le visage d'eau ». Lorsque je réalise cette activité avec un groupe de développeurs, je vois souvent la tâche « Appliquer un peu plus de déodorant ». Je ne juge pas, je constate simplement.

Voici ma carte découpée en tranches pour trouver les tâches que je vais devoir effectuer pour être prêt en quelques minutes.

Vous pouvez pratiquer cet exercice en pensant à des objectifs différents à positionner sur le côté gauche de la carte, comme « Partir en vacances pour quinze jours ». Vous verrez que le flux narratif reste assez stable, mais que vous devez ajouter ou supprimer des tâches pour atteindre cet objectif différent.

Utilisez des tranches pour identifier toutes les tâches et tous les détails qui relèvent d'un outcome spécifique.

5.6 C'EST FINI ! VOUS AVEZ APPRIS TOUTES LES NOTIONS IMPORTANTES

C'était vraiment facile, n'est-ce pas ? En créant cette carte vous avez appris que :

- *Les tâches* sont de courtes phrases verbales qui décrivent ce que font les gens.
- Les tâches ont des *niveaux d'objectif* différents.
- Les tâches d'une carte sont disposées selon un *flux narratif* qui va de gauche à droite.
- La *profondeur* d'une carte contient des variations et des tâches alternatives.
- Les tâches sont classées par *activités* dans la partie supérieure de la carte.
- Les activités forment la *colonne vertébrale* de la carte.
- Vous pouvez découper la carte en *tranches* pour identifier les tâches dont vous aurez besoin pour atteindre un *outcome* particulier.

© Dunod – Toute reproduction non autorisée est un délit.

5.7 ESSAYEZ CECI À LA MAISON OU AU TRAVAIL

Je suis pratiquement sûr qu'un grand nombre d'entre vous se sont contentés de lire et n'ont pas réalisé de carte à la lecture de cet exercice. N'imaginez pas que je n'ai pas remarqué. Mais si vous êtes un de ces fainéants qui n'a pas eu le courage d'établir la carte de leur matinée, promettez-moi que vous allez essayer. C'est certainement ma façon préférée d'enseigner les concepts élémentaires du story mapping. Si c'est la première fois que vous créez une carte dans votre entreprise, réunissez un petit groupe de personnes et réalisez cet exercice. Vous apprendrez toutes les bases et vous serez ensuite capable de cartographier *n'importe quoi*.

Avez-vous besoin de prendre une douche avant de partir au travail ?

Rick Cusick, Reading Plus, Winooski, Vermont

Nous avons effectué cet exercice de cartographie de la matinée avec quatre développeurs, le PO, un testeur, un responsable UX et deux de nos formateurs. Le groupe a été divisé en deux équipes, et nous avons enregistré rapidement la matinée de chaque personne, puis synthétisé nos matins respectifs en une représentation unique de ce qui ressemblait à « un matin moyen ». Les participants ont apprécié le travail de construction de la carte, même s'ils n'avaient jamais fait cela auparavant ou s'ils n'avaient jamais envisagé cela en termes de création de leur propre expérience de produit.

Mes objectifs au cours de cet exercice étaient de promouvoir l'efficacité de la visualisation de notre travail, de montrer comment le fait d'assembler la carte créait une compréhension mutuelle et de tirer parti de l'affichage de l'expérience dans un format accessible. Cette étroite collaboration a produit des avantages inattendus, à savoir le travail en équipe sur un projet où le but a été révélé par le biais du travail lui-même, et les moments d'empathie entre les participants : « Je ne savais pas que tu déposais tes enfants à l'école tous les jours », « Tu fais du yoga le matin avant de partir au travail ? ». Il y a eu une certaine confusion autour des événements survenus en même temps, ou avec un lien de causalité : « *Si je lis le journal pendant que je bois du café, est-ce qu'il faut un ou deux Post-it® ? Le vendredi, c'est ma femme qui emmène les enfants à l'école, alors comment dois-je représenter cela ?* ». L'autre défi a été que la nature linéaire du temps dans une story map orientée de gauche à droite n'était pas en mesure d'intégrer toutes les possibilités. En tant qu'animateur, j'ai trouvé gratifiant de voir ce genre de réflexion pendant l'exercice, même si je n'avais pas toutes les réponses à ce moment-là.

Comme nous avons hiérarchisé les activités, des choix difficiles ont dû être faits, ce qui a engendré des choses comiques. « *Avons-nous besoin de prendre une douche avant de partir au travail ?* » est une plaisanterie qui a surgi au milieu de nos réflexions. « *Peu importe ce nous devons sacrifier : il faut se réveiller, s'habiller et se rendre au travail* » a observé l'un des participants, avant qu'un autre ne le coupe en disant « *sauf si tu travailles à la maison !* ».

Peu de temps après cet exercice, le story mapping est devenu notre méthode préférée pour communiquer une expérience, hiérarchiser les user stories et planifier les itérations et les releases. Cette pratique est rentrée définitivement dans les usages de la société et la culture de l'entreprise.

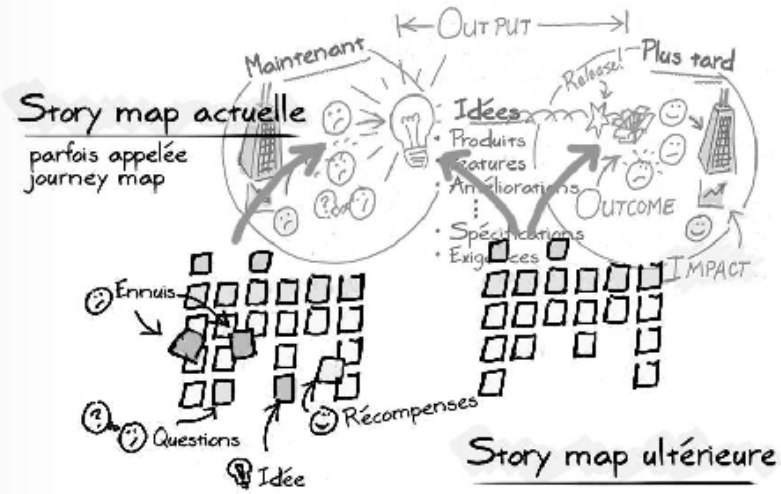
Après avoir fait ce même exercice avec plusieurs équipes au sein de notre organisation, j'en ai tiré une leçon : il faut au départ conditionner l'état d'esprit des participants. Démarrez la session en demandant à chaque personne d'écrire juste une chose qu'elle a faite après son réveil et avant de partir au travail. Puis demandez à chaque personne de répondre à la question : « *Pourquoi avez-vous choisi cette action ?* ». Je trouve que cela instaure un réflexe mental qui resurgira plus tard au cours des séances de planification : « *Quelle est la valeur de cette user story ? Pourquoi nos utilisateurs feraient-ils cela ?* ».

5.8 IL S'AGIT D'UNE CARTE ACTUELLE, PAS D'UNE CARTE DU FUTUR

Je soupçonne qu'un certain nombre d'entre vous ont compris cela, mais la carte que vous venez de créer est fondamentalement différente des cartes créées dans les quatre premiers chapitres. Les cartes réalisées par Gary, Globo.com, Eric, Mike et Aaron imaginaient toutes comment les utilisateurs allaient se servir de leurs produits dans le futur, c'est-à-dire *plus tard* après la livraison du produit. Ils ont écrit des tâches et des activités qu'ils ont imaginé que des gens réaliseraient avec un produit. En revanche, la carte que vous avez créée est une carte sur la façon dont vous faites les choses *maintenant*, en l'occurrence ce matin. Il s'avère que les concepts sont les mêmes dans les deux cas et vous n'avez donc pas perdu votre temps.

Une des choses intéressantes à propos des « story maps actuelles » est que vous pouvez les créer afin de mieux comprendre la manière dont les gens fonctionnent aujourd'hui. Vous avez juste fait cela pour apprendre comment se préparer ce matin. Vous pouvez encore apprendre bien d'autres choses si vous retournez à votre carte et y ajoutez des détails supplémentaires. Parmi les choses faciles à ajouter, on peut citer :

- **Les difficultés** :
  Les choses qui ne fonctionnent pas, ce que les gens détestent.
  Les récompenses.
  Les choses motivantes, les choses qui font que cela en vaut la peine
- **Les questions** – Pourquoi les gens font-ils cela ? Que se passe-t-il quand ils le font ?
- **Les idées** – Les choses que les gens pourraient faire ou que nous pourrions créer pour les soulager.



Beaucoup de gens dans la communauté de l'expérience utilisateur ont créé ce genre de choses depuis plusieurs années afin de mieux comprendre leurs utilisateurs. On appelle cela parfois des *journey maps*, mais l'idée de base est identique.

5.9 ESSAYEZ CECI POUR DE VRAI

Au début des années 2000, je dirigeais une équipe dans une petite société nommée Tomax. Nous développions des logiciels pour le commerce de détail, ces magasins dans lesquels nous avions l'habitude d'aller avant de passer tout notre temps sur Internet. Nous avions un nouveau client qui tenait une grande chaîne de magasins de peinture et de décoration d'intérieur. Nous connaissions bien le commerce de détail (des utilisateurs vendent des choses dans un point de vente et gèrent des stocks), mais il y avait certaines choses qui étaient propres aux magasins de peinture et de décoration que nous ignorions. Par exemple, nous ne savions pas comment vendre de la peinture avec une teinte personnalisée ou bien des stores sur mesure, et nous avons dû apprendre rapidement.

Pour faciliter notre apprentissage, nous avons demandé l'aide de ces trois dames. Elles ne sont pas informaticiennes, mais ce sont des décoratrices d'intérieur travaillant pour l'entreprise qui voulait notre logiciel. Ce sont elles qui nous ont appris les tenants et aboutissants de la vente de stores sur mesure. Afin que nous puissions apprendre rapidement, nous leur avons demandé de repenser à la dernière fois où elles avaient vendu des stores sur mesure.

© Dunod – Toute reproduction non autorisée est un délit.

Nous leur avons demandé d'écrire tout ce qu'elles avaient fait, dès l'instant où un client les avait contactées, jusqu'au moment où les stores avaient été installés et le client satisfait. Cela doit maintenant vous paraître familier, parce que nous leur avons demandé de faire la même chose que ce que vous venez de faire, c'est-à-dire décrire votre matinée à l'aide d'une carte. Elles ont pu dire ce *qu'elles* avaient fait pour vendre des stores sur mesure aussi facilement que vous quand vous avez décrit vos préparatifs du matin. Et lorsque nous avons organisé leurs tâches, nous avons tous appris qu'il n'y avait pas qu'une seule façon de faire les choses, que chacune d'entre elles faisait les choses différemment ou bien dans un ordre différent. Vous constaterez la même chose si vous essayez de créer la carte des préparatifs du matin avec un petit groupe de personnes différentes.

À partir de cette simple narration et grâce à l'activité de cartographie, nous avons tous forgé une *compréhension mutuelle* de la manière dont elles travaillaient *à l'époque*. C'est à partir de ce moment-là que nous avons pu commencer à traduire cette carte en tâches qu'elles auraient besoin d'accomplir dans le logiciel que nous allions développer *plus tard*.

5.10 AVEC LES LOGICIELS, C'EST PLUS DIFFICILE

Je ne vous mentirai pas : si vous êtes un informaticien, il est possible que cela vous prenne un certain temps pour cesser de parler de features et d'écrans, et commencer à écrire de courtes phrases verbales qui décrivent ce que les gens tentent réellement de faire. Mais continuez à vous entraîner et vous y arriverez.

© Dunod – Toute reproduction non autorisée est un délit.

Ce sera vraiment difficile si vous ne savez pas exactement qui est votre utilisateur, ce qu'il tente d'accomplir, ou comment il s'y prend. Malheureusement, la création d'une carte dans cette situation permettra simplement de signaler ce que vous ne connaissez pas. Si tel est votre cas, alors vous aurez besoin d'en apprendre davantage sur les gens et sur ce qu'ils font. Ce qui est encore mieux, c'est de travailler avec eux directement pour créer une carte.

Six étapes simples pour faire du story mapping

Je peux résumer les quatre derniers chapitres en seulement six étapes. Vous vous demandez peut-être pourquoi je n'ai pas fait cela plus tôt. Mais dans ce cas-là, je ne vous aurais pas parlé des stories et je vous aurais juste fourni les spécifications. Et dans ces conditions, ça ne fonctionne pas.

Même si je sais qu'il y a beaucoup de bonnes méthodes pour créer et utiliser une story map, je trouve que le processus suivant en six étapes marche bien pour moi :

1. **Cerner le problème** – À qui est destiné le logiciel et pourquoi nous le développons ?

2. **Cartographier la vision d'ensemble** – Il faut se concentrer sur la largeur, et pas sur la profondeur (1 kilomètre de large pour 1 centimètre de profondeur). Si vous n'avez pas une solution claire à l'esprit, ou même si vous pensez que c'est le cas, essayez de cartographier le monde tel qu'il est aujourd'hui, en y incluant les joies et les peines de vos utilisateurs.

3. **Explorez** – Parlez avec d'autres types d'utilisateurs, demandez-leur comment ils peuvent faire les choses autrement et étudiez la manière dont les choses peuvent mal tourner. Encore mieux, créez une esquisse et un prototype, faites des tests et affinez la solution, en modifiant et en améliorant la carte au fur et à mesure.

4. **Analysez une stratégie de release –** N'oubliez pas qu'il y a toujours trop de choses à créer. Concentrez-vous sur ce que vous essayez de réaliser pour votre entreprise et les utilisateurs de votre produit. Écartez ce qui n'est pas nécessaire pour présenter des solutions minimales qui satisferont les gens et aideront votre organisation à atteindre ses objectifs.

5. **Analysez une stratégie d'apprentissage** – Vous pensez avoir identifié une solution viable minimum, mais n'oubliez pas que ce n'est qu'une hypothèse tant que vous ne l'avez pas prouvée. Utilisez la carte et les discussions pour vous aider à identifier les risques les plus importants. Découpez la carte en expériences de produit viable minimum encore plus petites que vous pouvez présenter à un sous-ensemble de vos utilisateurs afin d'apprendre ce qui compte vraiment à leurs yeux.

6. **Analysez une stratégie de développement** – Si vous avez écarté tout ce que vous *n'avez pas besoin de* livrer au client, il ne vous restera que le strict nécessaire. Décomposez à présent votre solution viable minimum en parties que vous souhaitez

développer en priorité et en parties qui seront développées ultérieurement. En vous concentrant sur le développement des choses prioritaires, cela vous apprend à identifier et à anticiper les problèmes techniques et les risques de développement.

5.11 LA CARTE N'EST QUE LE COMMENCEMENT

La création d'une carte vous permet d'appréhender la vision d'ensemble et c'est l'un des plus grands avantages du story mapping. Vous avez déjà appris les deux choses principales qui font que les stories marchent :

- Employez une narration faite de mots et d'images pour créer une compréhension mutuelle.
- Ne vous contentez pas de parler de ce qu'il faut développer, mais parlez de vos utilisateurs et de leurs motivations afin de minimiser l'output et de maximiser l'outcome.

Gardez ces choses à l'esprit, et tout se mettra en place au fur et à mesure que vous allez de l'avant.

Il est temps d'évoquer quelques tactiques d'utilisation des stories parce qu'il est facile de se tromper, et vous avez encore des choses à apprendre pour employer les stories efficacement.

User story mapping chez SAP — Un problème de passage à l'échelle

Andrea Schmieden

Lorsque Jeff nous a présenté pour la première fois son concept de user story mapping, les gens de SAP ont immédiatement été séduits. Cela semblait être une méthode simple, mais néanmoins puissante pour transformer une vision de produit en un backlog et pour comprendre la nature du produit à développer, son public et ses motivations. Nous avons donc décidé de tester cette méthode.

Pourtant, comme nous n'allions pas tarder à le découvrir, ce qui peut être une chose simple pour un entrepreneur isolé ou une équipe Scrum individuelle est un monstre complètement différent pour des équipes de développement produit composées de *plusieurs* équipes Scrum. Chez SAP, entreprise composée de près de 20 000 développeurs, les grandes équipes de développement produit avec des dépendances sur d'autres équipes sont généralement la norme, et non pas l'exception. Il nous fallait donc trouver un moyen fiable pour le passage à l'échelle du concept de user story mapping dans une organisation de grande taille.

Le défi

Le défi était double pour nous :

– Comment arriver à cartographier des produits complexes sans se perdre dans les Post-it® ?

– Comment populariser la méthode au sein de l'organisation du développement et permettre aux gens de l'utiliser ?

© Dunod – Toute reproduction non autorisée est un délit.

1. User story mapping pour les grands produits

Pour trouver des réponses à la première question, nous avons décidé qu'il était préférable d'animer quelques ateliers pilotes avec des projets réels. Nous avons commencé avec une petite équipe de formateurs enthousiastes et une dizaine de projets pilotes, le plus important étant composé de 14 équipes Scrum. Dans cette phase pilote, nous avons fait varier plusieurs aspects de la méthode, comme les formats de l'atelier, les contenus, les phases du projet, les formats des cartes, etc. Après plusieurs boucles de rétroaction et des itérations, nous sommes arrivés à un ensemble de bonnes pratiques qui, pour l'instant, semble plutôt bien fonctionner dans notre contexte de développement à grande échelle.

Les bonnes pratiques clés

Quand une équipe utilise pour la première fois le user story mapping, nous recommandons l'implication d'un coach expérimenté. Le coach organise une réunion avec le demandeur et explique les objectifs de l'atelier, qui il faut inviter, le planning, les données pertinentes et ainsi de suite. En général, nous faisons un atelier d'une journée animé avec toute l'équipe et des séances de suivi plus petites selon les besoins.

Le jour de l'atelier, nous commençons généralement par un exercice de vision de produit comme le célèbre argumentaire éclair (*elevator pitch*) ou bien le format Cover Story[1], où l'équipe décrit ce qu'elle aimerait lire sur son produit dans un article d'une revue spécialisée publié dans un an. Cela montre si l'équipe a une compréhension commune de la direction générale, ou si elle a peut-être besoin d'investir dans des recherches supplémentaires (par exemple, plus d'entrevues, d'essais de prototypage, etc.).

L'étape suivante consiste à étudier les utilisateurs typiques du produit. Si l'objectif de l'atelier est de spécifier un backlog détaillé, les rôles utilisateur ou les personas doivent émerger de la phase de recherche utilisateur. Si le projet en est à son commencement, l'équipe rédige ses hypothèses qui peuvent ensuite être testées dans les phases de recherche utilisateur. Cela s'est avéré être un bon moyen de préparer la recherche utilisateur. C'est aussi un aspect où les pratiques de Design Thinking et le user story mapping fonctionnent très bien ensemble.

Nous utilisons une approche à trois niveaux pour définir les user stories : (1) on commence avec des étapes d'usage de haut niveau qui sont décomposées en (2) activités plus fines par rôle utilisateur qui, à leur tour, sont divisées en (3) user stories concrètes sous la forme *« en tant que <rôle>, je veux <la fonctionnalité> si bien que cette <valeur> »*. Ces user stories aboutissent à un premier backlog de produit. Cette approche à trois niveaux est particulièrement utile pour les projets de grande envergure. Sur chaque niveau, l'équipe peut décider où il paraît sensé d'analyser les détails et où les dépendances avec les autres équipes doivent être prises en considération. Cette approche aide à se concentrer sur les tâches de développement clés tout en conservant à l'esprit une vision d'ensemble.

1. Cover Story est l'une des nombreuses bonnes pratiques que l'on trouve dans le livre *Gamestorming* (http://gamestorming.com/?s=cover+story) publié par Dave Gray chez O'Reilly.

Pour faciliter la compréhension de la carte, nous utilisons un codage par couleur pour les activités et les user stories associées à un rôle ou un persona individuel, comme vous pouvez le constater dans le graphique suivant.

Souvent, tandis que l'équipe est en train de créer la carte, des problèmes surgissent, comme des « taches blanches » qui nécessitent des recherches supplémentaires de la part de l'équipe, ou des questions restées en suspens, des dépendances ou des lacunes. Pour mettre en évidence ces problèmes, nous utilisons des Post-it® de différentes couleurs ou de différentes tailles. Dans un premier temps, il peut sembler maladroit de placer tous ces problèmes non résolus sur la carte. Toutefois, selon notre expérience, c'est l'un des aspects les plus utiles du processus de mapping : on obtient une impression honnête et tangible des choses qui doivent encore être clarifiées. Une fois que les questions sont sur la table, il est beaucoup plus facile de s'y attaquer.

Lorsque l'équipe a atteint un niveau raisonnable de détail, on hiérarchise les user stories dans le backlog. En fonction de la taille et de la phase du projet, cela se fait parfois même sur le niveau des activités plutôt que sur le niveau des user stories. Généralement, nous utilisons des techniques simples de vote, comme le *dot voting*. Parfois, nous utilisons un modèle simplifié de Kano pour le vote, ce qui signifie que les équipes étiquettent les user stories avec les mentions « Indispensable », « Générateur de plaisir » ou « Générateur de satisfaction ». Ces résultats de votes simples sont à nouveau une bonne base pour la validation future avec les parties prenantes, les utilisateurs et les clients.

Comme le dit l'un de nos product owners, *« En tant que PO, c'est souvent un défi de faire tenir un grand nombre de spécifications dans un calendrier très serré. Nous avons invité nos clients à un atelier d'une journée sur le user story mapping, et il s'est avéré que c'était un moyen très efficace d'aboutir à une compréhension mutuelle de leurs priorités. »*.

En général, on ne traite pas dans l'atelier des détails supplémentaires, des estimations détaillées des efforts, et des choses du même ordre qui sont plutôt discutées en petits groupes par la suite.

2. Passage à l'échelle du user story mapping

Pour le passage à l'échelle de cette approche, la première équipe de coaches a fourni du matériel, notamment un modèle Excel pour les cartes, des modèles pour les personas, un programme de formation pour un atelier standard, des articles de wiki et des mémos de description de la méthode. En outre, un outil interne de user story mapping est en cours de développement.

Cependant, la production de documents de formation est une chose, mais la conduite d'un atelier en est une autre. Encore une fois, nous recommandons fortement l'implication d'un coach expérimenté dans la mise en place du processus. Pour être en mesure de fournir suffisamment de coaches, l'équipe d'encadrement initial a formé plus de coaches. Ces « coaches juniors » ont participé à un atelier avec un coach senior, ont suivi des séances individuelles puis ont organisé leurs propres ateliers. Nous avons aussi animé des ateliers et sessions de « formation de formateurs » dans les principales villes du monde où SAP a des équipes de développement. Pour nous assurer que nous apprenons les uns des autres, et de nos expériences diverses, nous avons mis en place un réseau mondial où les coaches peuvent partager leurs questions et leurs bonnes pratiques qui sont enregistrées dans des pages wiki et des communautés de pratique. Pour finir, nous avons énormément appris des nombreux échanges enrichissants que nous avons eus avec Jeff.

Nos efforts pour le passage à l'échelle du user story mapping ont été couronnés de succès. Nous avons animé plus de 200 ateliers dans différentes unités et différents endroits, et maintenant la plupart des équipes sont en mesure d'utiliser le user story mapping avec succès et de manière autonome.

Pour faciliter la compréhension de la carte, nous utilisons un codage par couleur pour les activités et les user stories associées à un rôle ou un persona individuel, comme vous pouvez le constater dans le graphique suivant.

Souvent, tandis que l'équipe est en train de créer la carte, des problèmes surgissent, comme des « taches blanches » qui nécessitent des recherches supplémentaires de la part de l'équipe, ou des questions restées en suspens, des dépendances ou des lacunes. Pour mettre en évidence ces problèmes, nous utilisons des Post-it® de différentes couleurs ou de différentes tailles. Dans un premier temps, il peut sembler maladroit de placer tous ces problèmes non résolus sur la carte. Toutefois, selon notre expérience, c'est l'un des aspects les plus utiles du processus de mapping : on obtient une impression honnête et tangible des choses qui doivent encore être clarifiées. Une fois que les questions sont sur la table, il est beaucoup plus facile de s'y attaquer.

Lorsque l'équipe a atteint un niveau raisonnable de détail, on hiérarchise les user stories dans le backlog. En fonction de la taille et de la phase du projet, cela se fait parfois même sur le niveau des activités plutôt que sur le niveau des user stories. Généralement, nous utilisons des techniques simples de vote, comme le *dot voting*. Parfois, nous utilisons un modèle simplifié de Kano pour le vote, ce qui signifie que les équipes étiquettent les user stories avec les mentions « Indispensable », « Générateur de plaisir » ou « Générateur de satisfaction ». Ces résultats de votes simples sont à nouveau une bonne base pour la validation future avec les parties prenantes, les utilisateurs et les clients.

Comme le dit l'un de nos product owners, *« En tant que PO, c'est souvent un défi de faire tenir un grand nombre de spécifications dans un calendrier très serré. Nous avons invité nos clients à un atelier d'une journée sur le user story mapping, et il s'est avéré que c'était un moyen très efficace d'aboutir à une compréhension mutuelle de leurs priorités. »*.

En général, on ne traite pas dans l'atelier des détails supplémentaires, des estimations détaillées des efforts, et des choses du même ordre qui sont plutôt discutées en petits groupes par la suite.

2. Passage à l'échelle du user story mapping

Pour le passage à l'échelle de cette approche, la première équipe de coaches a fourni du matériel, notamment un modèle Excel pour les cartes, des modèles pour les personas, un programme de formation pour un atelier standard, des articles de wiki et des mémos de description de la méthode. En outre, un outil interne de user story mapping est en cours de développement.

Cependant, la production de documents de formation est une chose, mais la conduite d'un atelier en est une autre. Encore une fois, nous recommandons fortement l'implication d'un coach expérimenté dans la mise en place du processus. Pour être en mesure de fournir suffisamment de coaches, l'équipe d'encadrement initial a formé plus de coaches. Ces « coaches juniors » ont participé à un atelier avec un coach senior, ont suivi des séances individuelles puis ont organisé leurs propres ateliers. Nous avons aussi animé des ateliers et sessions de « formation de formateurs » dans les principales villes du monde où SAP a des équipes de développement. Pour nous assurer que nous apprenons les uns des autres, et de nos expériences diverses, nous avons mis en place un réseau mondial où les coaches peuvent partager leurs questions et leurs bonnes pratiques qui sont enregistrées dans des pages wiki et des communautés de pratique. Pour finir, nous avons énormément appris des nombreux échanges enrichissants que nous avons eus avec Jeff.

Nos efforts pour le passage à l'échelle du user story mapping ont été couronnés de succès. Nous avons animé plus de 200 ateliers dans différentes unités et différents endroits, et maintenant la plupart des équipes sont en mesure d'utiliser le user story mapping avec succès et de manière autonome.

6

La véritable histoire des stories

Le story mapping est une idée extrêmement simple. À l'aide d'une carte, vous pouvez travailler en groupe pour décrire une story de produit et bénéficier d'une vision d'ensemble au fur et à mesure du processus. Ensuite, vous composez cette vision d'ensemble pour prendre de bonnes décisions de planification. Il n'y a rien d'autre derrière ce concept de story en développement agile.

6.1 L'IDÉE ÉTONNAMMENT SIMPLE DE KENT

Le concept de story a été inventé par quelqu'un de très intelligent qui se nomme Kent Beck. Kent travaillait avec d'autres personnes au développement de logiciels à la fin des années 1990, et il a remarqué que l'un des plus grands problèmes du développement provenait du processus traditionnel d'utilisation de documents pour décrire précisément ce qui était souhaité, c'est-à-dire les **spécifications**. Vous connaissez maintenant le problème puisque différentes personnes peuvent lire le même document, mais imaginer des choses différentes. Ils peuvent même « signer » un document en estimant qu'ils sont d'accord.

C'est plus tard, quand nous entrons au cœur du développement de logiciels, voire plus tard encore, quand l'application a été livrée, que nous nous rendons compte que nous ne pensions pas aux mêmes choses. Beaucoup de gens appellent ce manque de compréhension mutuelle de « mauvaises spécifications ».

J'ai le plaisir de travailler avec de nombreuses équipes et on commence souvent notre collaboration en évoquant leurs plus grands défis. De très loin, ce sont les « mauvaises spécifications » qui sont le plus citées. Et puis tout le monde dénigre ce

© Dunod – Toute reproduction non autorisée est un délit.

Je suis heureux que nous soyons tous d'accord

document. L'auteur du document se sent mal à l'aise car il aurait dû rédiger plus de pages, ou moins de pages, ou bien encore utiliser une technique sophistiquée d'ingénierie des exigences. Ceux qui ont signé ce document commencent par culpabiliser, pour ensuite s'indigner : « *Vous ne vous attendiez quand même pas à ce que je lise tous les détails ! Après tout, nous avons parlé de cela pendant des jours et j'espérais que vous aviez compris ce que j'avais dit. De toutes les façons, je trouve votre foutu document de spécifications illisible* ». Et les développeurs du logiciel se sentent pris de court. Ils s'en sont sortis tant bien que mal avec ces documents cryptiques, et ce qu'ils ont développé ne convient toujours pas. Tout le monde finit par détester ce document, mais pourtant nous continuons à essayer de l'améliorer.

Deux personnes peuvent lire le même document, mais aboutir à une compréhension différente.

Mais les malentendus nés de la lecture du document ne sont que la moitié du problème. Nous gaspillons beaucoup de temps et d'argent à développer ce que décrit le document, pour finir par découvrir plus tard que ce qui résout réellement le problème prévu est quelque chose de bien différent. En effet, ces documents décrivent souvent précisément ce qui ne convient pas. Ils décrivent en général *ce dont* nous avons besoin, mais pas *les raisons* pour lesquelles nous en avons besoin. Si la personne chargée de développer un logiciel pouvait simplement parler avec quelqu'un qui a compris les besoins des utilisateurs du logiciel et leurs motivations, on trouverait alors souvent des moyens plus rentables de contenter les utilisateurs. Si l'on ne se parle pas, on ne peut tout simplement pas se comprendre.

Les meilleures solutions proviennent de la collaboration entre ceux qui ont des problèmes à résoudre et ceux qui peuvent les résoudre.

© Dunod – Toute reproduction non autorisée est un délit.

L'idée simple de Kent était d'*arrêter* tout cela, d'arrêter de travailler aussi dur pour rédiger le document parfait et se réunir pour *raconter des histoires*. Les stories tirent leur nom non pas de la façon dont elles sont censées être écrites, mais de la façon dont elles sont censées être utilisées. Permettez-moi de reformuler cela avec plus d'emphase. Arrêtez tout ce que vous faites et dites ceci à haute voix :

On dit une story parce qu'on raconte l'histoire, pas parce qu'on l'écrit.

L'idée de Kent était simple : si nous nous réunissons pour parler du problème que nous allons résoudre avec le logiciel, sans oublier les utilisateurs et leurs motivations, nous pourrons ensemble parvenir à une solution et forger une compréhension mutuelle.

6.2 SIMPLE NE VEUT PAS DIRE FACILE

Il y a quelque temps, j'ai commencé à remarquer qu'avec ce concept de story les choses dérapaient, c'est-à-dire que beaucoup de ceux qui écrivent des livres, qui enseignent ou qui utilisent les stories se focalisaient sur l'activité de rédaction des stories.

Avec tout cet accent mis sur l'écriture des stories, j'ai repris contact avec Kent pour m'assurer que je n'avais pas raté quelque chose. Au cours d'un échange par courriel, Kent m'a expliqué d'où l'idée de story lui était venue :

> Je réfléchissais à la manière dont les utilisateurs racontent parfois des histoires sur les nouvelles fonctionnalités intéressantes du logiciel qu'ils utilisent. Par exemple, si je saisis le code postal, le nom de la ville est rempli automatiquement sans que j'aie à cliquer sur un bouton.
>
> Je pense que c'est l'exemple qui a déclenché mon idée. Si vous pouvez raconter des histoires sur les fonctionnalités du logiciel et susciter de l'intérêt chez la personne qui vous écoute, pourquoi ne pas raconter des histoires avant le développement du logiciel ?
>
> Courriel personnel de Kent Beck, août 2010.

L'idée est donc bien de *raconter*, et c'est un succès quand vous produisez de l'énergie, de l'intérêt et de la compréhension dans l'esprit de votre auditeur. C'est énorme, mais cela semble beaucoup plus amusant que de lire un cahier des charges classique[1].

Mais les gens qui commencent à utiliser des stories pour le développement de logiciels, et qui ont encore un schéma traditionnel dans leurs têtes, tendent à se concentrer sur la partie rédactionnelle. J'ai vu des équipes remplacer leur processus traditionnel de rédaction des spécifications par l'écriture de stories, puis se sentir frustrées par leur tentative d'écrire des stories pour communiquer précisément ce qui devait être développé. Si c'est à présent votre cas, arrêtez tout de suite !

1. Il a été prouvé qu'on mémorise beaucoup plus facilement un fait présenté dans une histoire qu'un fait présenté tout seul (22 fois plus selon le psychologue Jerome Bruner).

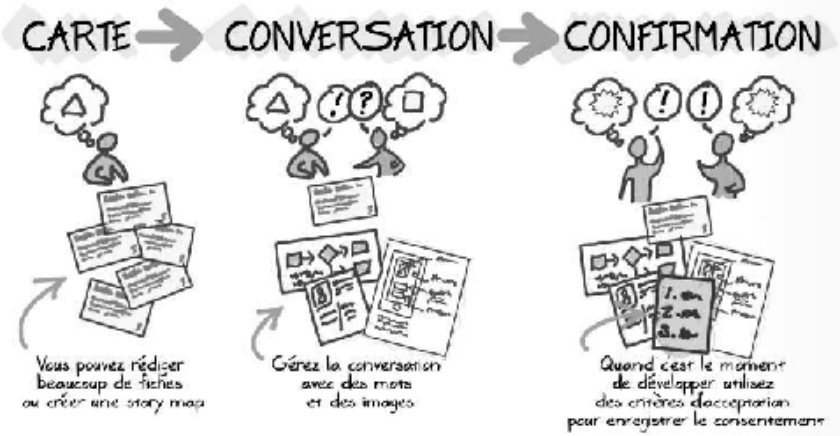
Si vous ne vous réunissez pas pour avoir des conversations fructueuses sur vos stories, alors vous n'utilisez pas vraiment des stories.

6.3 RON JEFFRIES ET LES 3 C

Dans son livre *Extreme Programming Installed*, Ron Jeffries (Addison-Wesley Longman Publishing) décrit très bien le processus de story :

- **Carte** – Écrivez ce que vous voudriez voir dans le logiciel sur une série de fiches[1].
- **Conversation** – Réunissez-vous et ayez une conversation enrichissante sur le logiciel à développer.
- **Confirmation** – Mettez-vous d'accord sur la manière dont vous allez valider que le logiciel est terminé.

Si cela paraît simple, c'est parce que ça l'est. Mais n'oubliez pas que simple ne signifie pas facile.



6.3.1 Carte

Imaginez que vous êtes chargé de travailler avec une équipe qui doit développer un logiciel. Si pour chaque chose que les utilisateurs veulent réaliser avec le produit, vous écrivez une fiche, vous vous retrouverez alors avec un tas de fiches. L'idée originale de Kent était de rédiger sur des fiches parce qu'il est facile d'organiser un tas de fiches sur une table. Il est aussi plus facile de les hiérarchiser ou de les organiser dans une structure qui permet d'avoir une vision d'ensemble, sous une forme de story map, bien entendu !

Le tas de fiches qui décrit l'ensemble du produit, ou tous les changements que nous souhaitons apporter à un produit, s'appelle un **backlog de produit**, terme qui provient de la méthode agile Scrum.

1. Ndt : il s'agit en fait du mot anglais *card*, que nous traduisons habituellement par fiche pour éviter la confusion avec le terme anglais *map* que nous traduisons par carte. Pour garder les 3 C, nous avons choisi, dans ce contexte, de traduire *card* par carte, mais il s'agit bien d'une fiche.

6.3.2 Conversation

La conversation peut commencer par la description de ce que vous pensez, et la personne qui vous écoute peut se représenter une idée dans sa tête, en se basant sur ce qu'elle a entendu. Comme il est difficile d'expliquer les choses parfaitement, et qu'il est facile d'imaginer des choses différentes en se basant sur nos expériences passées, votre auditeur a probablement imaginé quelque chose de différent. Mais c'est là qu'intervient la magie.

Parce qu'il s'agit d'une conversation, l'auditeur peut poser des questions et revenir en arrière pour améliorer sa compréhension et aider tout le monde à parvenir à un certain degré de compréhension mutuelle.

Dans un processus logiciel traditionnel, l'objectif de la personne chargée de rédiger les spécifications est de les écrire correctement ; et pour la personne qui va développer le logiciel, c'est de les comprendre correctement. Mais parce qu'il s'agit ici d'un processus piloté par les stories, ces objectifs auparavant différents évoluent pour se fondre dans un objectif commun. Votre but est de travailler ensemble pour comprendre le problème à résoudre par le logiciel, et le résoudre du mieux possible. Au bout du compte, vous aurez besoin de vous entendre sur ce que vous devez développer en espérant tous les deux que cela facilitera la vie des utilisateurs du produit.

Je le répète parce qu'il s'agit d'un point important :

Les conversations de stories sont une façon de travailler ensemble pour parvenir à une bonne compréhension du problème à résoudre.

6.3.3 Confirmation

C'est bien beau de parler, mais nous devons au bout du compte développer un logiciel. Dans ces conditions, lorsque nous sentons que nous convergeons vers une solution correcte, il faut nous concentrer sur les réponses à ces questions :

a) Si nous développons ce sur quoi nous nous sommes mis d'accord, qu'allons-nous vérifier pour être sûrs que nous avons terminé ?
   La réponse à cette question est habituellement une courte liste de points à vérifier. Cette liste est souvent appelée *critères d'acceptation*, ou *tests de story*.

b) Quand plus tard il faudra faire une démonstration du logiciel, comment allons-nous procéder ?

© Dunod – Toute reproduction non autorisée est un délit.

La réponse à cette question révèle souvent quelques lacunes. Par exemple, si le logiciel fonctionne, il vous faudra mettre la main sur un jeu de données pour effectuer la démonstration. Le fait de discuter de la démonstration peut rallonger votre liste de critères d'acceptation.

6.4 DES MOTS ET DES IMAGES

Le chemin pour arriver à cet accord ne se résume pas à une seule fiche et beaucoup de mains qui s'agitent. La conversation se déroule mieux si elle est enrichie d'autres choses comme de simples personas, des diagrammes de flux de travail, des esquisses de l'interface utilisateur et tout autre modèle de logiciel traditionnel qui vous aidera à expliquer les choses. De cette façon, vous n'avez pas seulement à agiter les mains, mais vous pouvez aussi les utiliser pour pointer. Tout ce qui enrichit la conversation est utilisé pour baliser, écrire, corriger et modifier. Utilisez un tableau blanc ou un tableau papier, et n'oubliez pas de prendre quelques « photos de vacances » avant de quitter la pièce. Ces photos de ce que vous avez créé vous aideront à vous souvenir de tous les détails de la conversation qui sont difficiles à transcrire.

Les conversations d'une bonne story contiennent beaucoup de mots et d'images.

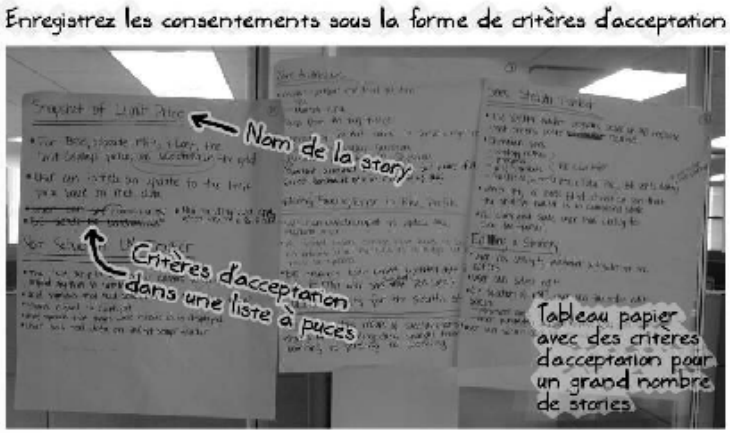
Vos critères d'acceptation doivent rester bien visibles au cours de la conversation. Cette équipe utilise un tableau papier pour les enregistrer au fur et à mesure.

6.5 C'EST TOUT

C'est tout ce qu'il y a à savoir de l'idée étonnamment simple de Kent. Et si vous la mettez en pratique, je vous promets que ça va tout changer.

À moins que ce ne soit pas le cas.

© Dunod – Toute reproduction non autorisée est un délit.



Pour les gens habitués à travailler de manière traditionnelle, cette conversation peut très mal se passer. Ils retombent dans leurs vieux schémas où chacun essaye de comprendre l'autre sans jamais vraiment y arriver. Si cela dure trop longtemps, ils ont souvent l'impression d'une animosité qui n'est guère propice à faire évoluer les choses.

Pour mener à bien ces conversations, il faut garder à l'esprit certaines choses qui feront l'objet du chapitre suivant.

7

Racontez de meilleures stories

L'idée des stories est simple, peut-être trop simple. Pour beaucoup de gens travaillant dans le développement logiciel, les conversations des stories sont peu familières et les mettent parfois mal à l'aise si bien qu'ils se remettent souvent à parler de spécifications, comme ils l'ont toujours fait.

Quand Kent Beck a décrit pour la première fois le concept de stories, il ne les a pas appelées *user stories*, mais simplement *stories*, parce qu'il les voyait comme de simples histoires. Mais très peu de temps après la publication des premiers livres sur *Extreme Programming*, le préfixe plus descriptif *user* a été accolé à *story* de telle sorte qu'on se souvienne de tenir des conversations avec le point de vue des gens extérieurs au logiciel. La modification du nom n'a cependant pas été suffisante.

© Dunod – Toute reproduction non autorisée est un délit.

7.1 LE MODÈLE DE CHEZ CONNEXTRA

Dans les années 1990, mon amie Rachel Davies a travaillé pour une société appelée Connextra, qui a été l'une des premières entreprises à adopter *Extreme Programming*, la méthode agile à l'origine des stories. Quand les employés de Connextra ont commencé à utiliser des stories, ils ont constaté qu'ils rencontraient couramment certains problèmes. La plupart des gens qui écrivaient les stories chez Connextra venaient du secteur des ventes et du marketing. Ils avaient tendance à rédiger la feature dont ils avaient besoin, mais quand c'était au tour des développeurs d'avoir des conversations, ils avaient du mal à trouver la partie prenante à l'initiative du projet pour entamer une bonne conversation, celle qui pose les questions du qui et du pourquoi. Le simple fait d'avoir le nom de la feature n'aidait pas l'équipe à trouver les

bonnes personnes à qui parler, ou à démarrer les bonnes discussions. Et, à cette époque, il n'y avait pas beaucoup de conseils sur ce qui devait figurer sur une fiche. Le modèle a évolué au fil du temps chez Connextra. Ce n'était pas spécifiquement l'invention de Rachel, mais c'était plutôt un désir collectif de créer des choses qui comptaient.

Rachel tient une fiche de story.

Après avoir utilisé le modèle pendant un certain temps, les employés de Connextra ont voulu faire la promotion de ce nouveau truc intéressant et ils ont imprimé un tas de fiches exemples pour les montrer à XPDay 2001, une petite conférence qui se tenait à Londres. C'est cette fiche que Rachel tient à la main.

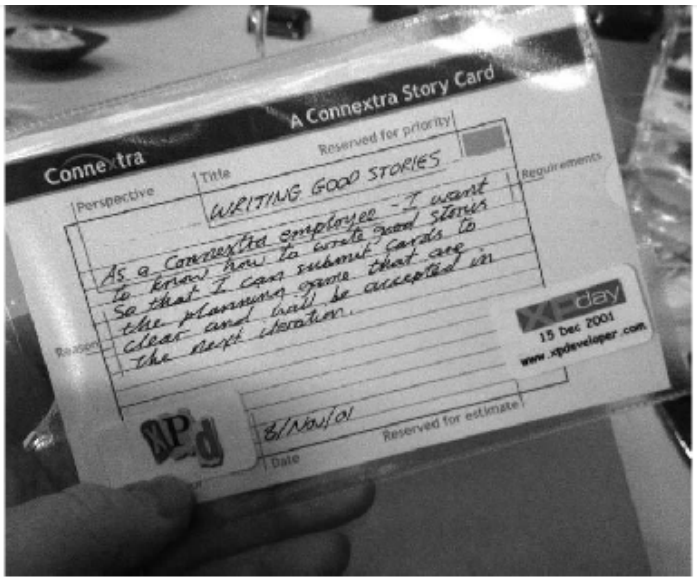
Le modèle se présente comme ceci :

- En tant que [type d'utilisateur]
- Je veux [faire quelque chose]
- Afin que je puisse [en tirer des bénéfices]

Les gens de chez Connextra utilisaient ce modèle simple pour écrire la description de leurs stories. Vous remarquerez que leurs stories ont toujours des titres courts et descriptifs. Ils ont trouvé que la rédaction de cette information supplémentaire pour commencer à raconter la story les contraignait à faire une pause et à réfléchir aux questions de base : qui, quoi et pourquoi. Et, quand ils ne savaient pas répondre, ils pouvaient se demander s'ils devaient finalement écrire la story.

Quand ils s'asseyaient pour entamer la conversation de la story, ils prenaient cette fiche et lisaient cette description qui constituait le début de la conversation.

Utilisez ce modèle simple de story pour commencer les conversations.



Si vous piochez une fiche individuelle dans une story map, le modèle est un bon moyen pour démarrer la conversation. Rappelez-vous le chapitre 1 où l'une des fiches de la carte de Gary stipulait : « Charger une image ». Cette fiche était l'un des éléments de la carte « Personnaliser mon dépliant publicitaire » (j'écris sur les fiches des phrases verbales courtes qui représentent les tâches que les utilisateurs exécutent avec mon logiciel). Mais si je prends une fiche au hasard, j'ai besoin de resituer le contexte de la fiche par rapport au grand voyage de l'utilisateur que décrit la carte et peux m'exprimer dans ces termes :

> « En tant que manager du groupe, je veux charger une image afin de personnaliser mon dépliant publicitaire. »

C'est un truc assez cool. Je peux trouver les fiches des différents utilisateurs au-dessus de la carte, et l'objectif principal des utilisateurs est souvent situé dans la carte au-dessus de l'objectif que vous visualisez à ce moment-là.

Mais n'oubliez pas qu'il s'agit juste du début de la conversation qui peut se poursuivre de la manière suivante :

> « – Pourquoi le manager du groupe voudrait personnaliser le dépliant ?
>
> – Eh bien, parce qu'il ne dispose pas de la photo de son groupe automatiquement, et il veut qu'elle soit sur le dépliant. Et il veut à tout prix être original et ne souhaite pas que son dépliant soit commun.
>
> – C'est logique. Où les gens comme les managers de groupes conservent-ils ces photos ?
>
> – Eh bien, en fait un peu partout. Elles peuvent être stockées sur leurs disques durs, dans des comptes Flickr, ou dans d'autres endroits sur le Web.

© Dunod – Toute reproduction non autorisée est un délit.

– C'est différent de ce que je pensais au départ car j'avais supposé qu'elles seraient simplement sur leurs disques durs.

– Non, beaucoup de gens à qui nous avons parlé les stockent un peu partout et c'est un problème. »

En fait, une grande partie de la conversation que j'ai eue avec Gary s'est passée comme ça. Au fur et à mesure que nous parlions, on écrivait des choses sur un tableau blanc ou directement sur une fiche.

Vous pouvez constater que le modèle très court n'est pas suffisant pour contenir les spécifications. Mais quand on commence une conversation à l'aide de ce modèle, on finit par avoir une conversation beaucoup plus riche que si nous avions simplement parlé de la possibilité de charger un fichier.

Redécouvrir la petite conversation

Matt Cropper, ThoughtWorks

Je travaillais comme analyste métier sur un projet ThoughtWorks pour une agence du gouvernement britannique. Nous étions chargés de la livraison, mais aussi de transférer à l'équipe cliente certaines expériences pratiques autour des méthodes agiles. Cela étant, nous étions une grosse équipe d'environ 25 membres, informaticiens et commerciaux.

Pour commencer, le PO et moi-même souhaitions écrire des stories, puis au début de chaque itération de deux semaines, toute l'équipe se réunissait pour la planification. C'était nécessairement une grosse réunion, et l'atmosphère était très tendue : « *Pourquoi faites-vous cela de cette manière ? Ces stories sont beaucoup trop grandes/petites. Cela n'est pas logique ! J'ai une forte préférence pour cette mise en œuvre particulière...* » Ces discussions étaient régulières et frustrantes. Pour être tout à fait honnête, je quittais ces réunions avec un sentiment de découragement et d'échec personnel.

Il fallait faire quelque chose pour résoudre ce problème, et nous avons alors décidé qu'au lieu d'avoir une réunion plénière pour discuter de tout, nous aurions des conversations avec un sujet plus restreint. L'affinage (*grooming*) du backlog, par exemple, a eu lieu au cours de la première semaine de l'itération avec un petit groupe (PO, chef de projet, analyste métier, architecte technique) qui a évalué les différentes stories, si bien que quand nous avons fait par la suite une séance de revue de backlog avec l'équipe au complet, il y a eu beaucoup moins d'interruptions. La conversation a tourné autour de l'optimisation et de l'amélioration de nos stories, et nous avons laissé de côté des choses comme la priorisation, les story points, etc. Notre changement de méthode a fonctionné.

Nous avons également fait en sorte de créer des stories de façon beaucoup plus constructive. J'avais en tête les stories sur lesquelles je travaillais cette semaine-là, et qui allaient se retrouver en haut sur le mur de fiches dans la colonne « En cours d'analyse ». Chaque jour, dans la réunion d'équipe, j'annonçais que je travaillais sur une story particulière et je pouvais réquisitionner deux développeurs pour travailler avec eux. On s'asseyait et on discutait de nos objectifs, généralement en nous penchant sur les aspects techniques, puis on notait tout sur du papier. Nous avons laissé de côté Trello, que nous utilisions pour notre mur de fiches numériques à l'époque et nous avons privilégié les conversations individuelles, parfois debout face à un tableau blanc. Le fait de travailler en groupe, et de se plonger dans les détails, est en fait assez gratifiant, et comme cela ne prend qu'une vingtaine de minutes à chaque fois, ça ne constitue pas trop une surcharge de travail. Les gens étaient réellement heureux de travailler en petit groupe et de contribuer.

Cela a eu pour heureuse conséquence que nos interminables sessions d'affinage de backlog se passèrent beaucoup mieux ; nous avons également constaté que la taille des stories avait tendance à s'uniformiser. La planification des itérations est donc devenue moins problématique. La qualité des conversations s'est améliorée, et il a en été de même du travail que nous avons produit.

7.2 LE PATTERN ZOMBIE ET LE CHASSE-NEIGE

Le terme *pattern zombie* est tiré du livre *Adrenaline Junkies and Template Zombies: Understanding Patterns of Project Behavior*[1] de Tom DeMarco (Dorset House). Même si le nom dit bien ce qu'il veut dire, je vais vous donner la définition des auteurs :

> Pattern zombie : l'équipe du projet permet que ses travaux soient pilotés par des patterns plutôt que par le processus de réflexion nécessaire pour livrer les produits.

Aussi simple que soit le pattern zombie, il est parfois maltraité. Je vois des gens s'acharner à faire rentrer des idées dans le pattern quand elles n'y ont pas leur place. Les stories concernant les services de back-end ou des problèmes de sécurité peuvent être complexes, et je vois des gens écrire et penser à des choses en ne prenant en compte que leur propre point de vue, et non pas celui des gens qui sont censés en bénéficier au bout du compte : « *en tant que PO, je veux que vous développiez un uploader de fichier de telle sorte que les exigences du client soient remplies* ». On ne doit pas penser les choses de cette manière.

Pire encore, le pattern est devenu si omniprésent et il est tellement enseigné dans les écoles, qu'il y a ceux qui croient qu'il ne peut y avoir de story si elle n'est pas écrite sous cette forme. Beaucoup de gens ont même cessé d'utiliser des titres dans les stories et n'écrivent qu'une longue phrase sur chaque fiche. Imaginez-vous en train d'essayer de lire une liste de stories écrites de cette façon. Imaginez-vous en train de raconter à quelqu'un une story à l'aide d'une story map où chaque Post-it® est écrit de cette façon. Votre cerveau ne va pas apprécier !

Tout cela me rend triste, parce que la valeur réelle des stories n'est pas ce qui est écrit sur la fiche, mais ce que nous apprenons lorsque nous la racontons.

Il n'est pas nécessaire qu'une story soit écrite à l'aide d'un modèle comme celui de chez Connextra pour qu'elle soit considérée comme une story.

1. On pourrait traduire en français le titre par « Accros à l'adrénaline et patterns zombies : comprendre les patterns de comportement de projet.

© Dunod – Toute reproduction non autorisée est un délit.

La personne sur cette photo[1] est en train d'apprendre à skier. Si vous n'avez jamais appris à skier et que quelqu'un de charitable tente de vous en enseigner les rudiments, vous allez reproduire ce que fait cette personne. La technique employée ici s'appelle le chasse-neige : les pointes de vos skis forment un V et vous vous penchez en avant. C'est la meilleure façon de contrôler votre vitesse et de rester debout quand vous avez deux planches glissantes accrochées à vos pieds. C'est cette méthode que je vous recommande pour apprendre à skier. Elle n'est cependant pas conseillée, même s'il s'agit de la meilleure méthode d'*apprentissage*. Il n'y a d'ailleurs pas d'épreuve olympique de chasse-neige et vous n'allez impressionner personne sur les pistes avec votre posture de chasse-neige. Il n'y a pas non plus à se sentir gêné car si les gens vous voient skier de cette manière, ils sauront que vous êtes en train d'apprendre.

Pour moi, le pattern de story fonctionne un peu comme l'apprentissage du chasse-neige : il permet de rédiger les descriptions de vos premières stories. Dites-le à voix haute pour démarrer vos conversations de stories, mais ne vous inquiétez pas trop si vous trouvez que cela ne fonctionne pas toujours. À l'instar de la technique du chasse-neige pour les skieurs, ce n'est pas le meilleur choix pour les terrains difficiles.

Voici mon pattern préféré : si j'écris des stories sur des Post-it® ou des fiches, et qu'elles ne font pas partie d'une story map plus grande, je commence par leur donner un titre court et simple, puis en dessous, j'écris :

- Qui :
- Quoi :
- Pourquoi :

1. Cette photo, que j'ai trouvée sur Flickr et qui est distribuée en licence Creative Commons (CC BY) a été prise par Ruth Hartnup.

Et je laisse quelques lignes d'espace entre chaque mot, parce que je compte lister nommément les différentes personnes concernées par la story (qui), dire quelques mots sur l'objet de la story (quoi) et noter mes commentaires sur ses différentes raisons (pourquoi). Je laisse de la place sur la fiche pour ajouter des informations supplémentaires quand on commencera à parler de la story. C'est en fait ma hantise quand les gens écrivent le titre au milieu de la fiche car cela ne me laisse pas assez de place pour prendre des notes quand on commence à parler. Mais c'est mon côté pinailleur.

7.3 LISTE DE CONTRÔLE DE CE DONT IL FAUT VRAIMENT PARLER

Il faut vraiment parler des « qui »

S'il vous plaît, ne vous contentez pas de parler de « l'utilisateur ». Soyez précis et évoquez l'utilisateur dont vous voulez vraiment parler. Pour Gary, par exemple, il peut s'agir du manager de groupe ou bien du fan de musique.

Il faut parler des différents types d'utilisateurs. Pour de nombreux logiciels, en particulier les logiciels grand public, il y a des types d'utilisateurs très différents qui utilisent la même fonctionnalité. Parlez de la fonctionnalité du point de vue des différents utilisateurs.

Parlez des clients. Pour les produits grand public, le client (ou le prescripteur) peut être la même personne que l'utilisateur. Mais pour les produits professionnels, nous aurons besoin de parler des gens qui prennent des décisions d'achat, de leur organisation dans son ensemble, et de la manière dont ils en tirent parti.

Parlez des autres parties prenantes. Parlez des gens qui financent l'achat du logiciel. Parlez des autres personnes qui pourraient collaborer avec les utilisateurs.

Il y a rarement un seul utilisateur qui compte.

Il faut vraiment parler des « quoi »

J'aime que mes stories démarrent par des tâches utilisateur, ces choses que les gens veulent faire avec mon logiciel. Mais qu'en est-il des services qui, par exemple, autorisent le débit de votre carte de crédit pour un achat, ou bien vous authentifient sur un site d'assurance ? Vos utilisateurs n'ont pas fait un choix délibéré pour l'autorisation du débit de leur carte de crédit ou la vérification de leur identité. Je suis d'accord pour discuter des services et des différents systèmes qui les appellent ; on est d'accord pour parler des composants d'interface utilisateur spécifiques et du comportement de l'écran, mais il ne faut pas perdre de vue qui cela concerne et pour quelles raisons.

Il faut vraiment parler des « pourquoi »

Parlez des motivations de l'utilisateur spécifique et creusez profondément car les raisons peuvent être multiples. Il faut aussi parfois discuter pendant très longtemps pour vraiment percer les raisons sous-jacentes.

© Dunod – Toute reproduction non autorisée est un délit.

Parlez des raisons qui font que les autres utilisateurs, l'entreprise de l'utilisateur et les parties prenantes sont concernés. Il y a beaucoup de choses importantes qui se cachent dans les « pourquoi ».

Parlez de ce qui se passe en dehors du logiciel

Parlez de l'endroit où se trouvent les personnes qui utilisent votre produit quand elles s'en servent. Parlez du moment où elles utilisent le produit et à quelle fréquence. Parlez des autres personnes présentes quand elles utilisent le produit. Toutes ces choses donnent des indices sur ce que pourrait être une bonne solution.

Parlez de ce qui se passe mal

Que se passe-t-il lorsque les choses tournent mal ? Que se passe-t-il lorsque le système est hors-service ? Les utilisateurs ont-ils une solution de repli ? Comment leurs besoins sont-ils satisfaits aujourd'hui ?

Parlez des questions et des hypothèses

Si vous parlez de toutes ces choses, vous allez probablement tomber sur quelque chose que vous ne connaissez pas. Identifiez vos questions et discutez de leur importance pour obtenir des réponses avant le développement du logiciel. Décidez quelles personnes vont faire les démarches pour obtenir des réponses à ces questions et invitez-les à une nouvelle conversation. Vous constaterez qu'il faut de nombreuses conversations pour approfondir certaines stories.

Prenez le temps de remettre en question vos hypothèses. Comprenez-vous vraiment vos utilisateurs ? Est-ce vraiment ce qu'ils veulent ? Ont-ils vraiment ces problèmes ? Utiliseront-ils vraiment cette solution ?

Remettez en question vos hypothèses techniques. Quels sont les systèmes sous-jacents sur lesquels vous vous appuyez ? Fonctionnent-ils vraiment comme prévu ? Est-ce qu'il y a des risques techniques dont il faut tenir compte ?

Toutes ces questions et ces hypothèses peuvent nécessiter un travail réfléchi pour la résolution des problèmes ou bien l'apprentissage. Élaborez un plan pour faire exactement cela.

Parlez des solutions améliorées

Lors d'une conversation de story, le grand défi est d'arriver à abandonner des hypothèses initiales sur la solution, pour revenir au problème à résoudre et ensuite parvenir ensemble à une solution plus efficace et plus économique à développer.

Parlez du « comment »

Lors des conversations de story, j'entends souvent quelqu'un s'inquiéter et dire que l'on devrait parler du « quoi » et non pas du « comment », ce qui signifie que l'on devrait évoquer ce que les utilisateurs ont besoin de faire, et non pas la manière dont le code doit être écrit. Et je ressens la même inquiétude lorsque nous parlons du « quoi » sans

parler du « pourquoi ». Mais la vérité est que nous essayons d'optimiser le « quoi », le « comment » et le « pourquoi » dans une bonne conversation de story. Et il y a un problème quand l'une ou l'autre partie suppose qu'une solution particulière ou la façon dont elle est mise en œuvre est une « spécification ». Sans parler explicitement du « comment » (et si vous êtes développeur, je sais que vous y pensez), il est difficile de réfléchir au coût de la solution, qui, si elle est trop coûteuse, n'est peut-être pas le bon choix.

Respectez l'expertise des autres dans la conversation. Ne dites pas à un technicien hautement qualifié comment il doit faire son travail. Ne dites pas à quelqu'un qui connaît très bien les utilisateurs et leur travail qu'il ne comprend pas. Posez des questions et tentez vraiment d'apprendre les uns des autres.

Parlez de la durée

En fin de compte, nous devons prendre certaines décisions pour aller de l'avant et déterminer s'il faut développer quelque chose ou non. Et il est difficile de prendre ce genre de décision sans connaître le prix de revient.

Pour les logiciels, cela revient à estimer combien de temps va prendre l'écriture du code. Dans les premières conversations, cela peut être exprimé en des termes comme « vraiment beaucoup de temps » ou « quelques jours ». Il est préférable de comparer à quelque chose qui a déjà été développé, comme « environ le même temps que cette fonctionnalité de commentaire que nous avons développée le mois dernier ». Au fur et à mesure que nous nous rapprochons du développement de quelque chose, et que nous avons eu plusieurs conversations et pris plusieurs décisions, nous sommes en mesure d'être un peu plus précis. Mais nous savons toujours que nous parlons ici d'estimation et non pas d'engagement.

© Dunod – Toute reproduction non autorisée est un délit.

7.4 CRÉEZ DES PHOTOS DE VACANCES

Comme on parle beaucoup et que l'on ne veut rien oublier, il faut s'assurer d'enregistrer les choses spécifiques qui vous aident à vous souvenir des décisions que vous avez prises, ou des questions et des hypothèses que vous devrez examiner. N'oubliez pas d'extérioriser votre pensée pour que les autres participants à la discussion puissent voir ce qui est enregistré.

Si vous l'écrivez, vous pouvez vous y référer plus tard. Si c'est affiché sur le mur, vous pourrez uniquement pointer le contenu. Et si vous parlez en équipe, vous constaterez que vous n'aurez pas à tout répéter aussi souvent, parce que les gens s'en souviendront, surtout si vous représentez vos conversations par des dessins et des documents simples, nos fameuses photos de vacances...

La photo illustre un petit groupe en train d'avoir une discussion de story. Quand ils parlent, ils visualisent leurs idées et prennent des notes sur ce qu'ils décident.

Mon approche préférée consiste à faire exactement comme eux. J'enregistre sur un tableau papier ou un tableau blanc au fur et à mesure que l'on parle. J'aime indiquer qui participait à la conversation directement sur le tableau, puis je photographie le tableau quand j'ai terminé. Je partage ensuite la photo à l'aide d'un wiki ou d'un autre outil. Je sais que je pourrai extraire des détails ou les rédiger de façon plus formelle quand j'en aurai besoin ultérieurement. Si je ne me souviens pas exactement de ce qui a été dit dans la conversation, une des personnes présentes pourra m'aider et c'est une bonne chose que d'avoir écrit leurs noms.

7.5 IL Y A BEAUCOUP DE SUJETS DE PRÉOCCUPATION

Il est difficile de réfléchir à tous les sujets de conversation dans les stories. À ce stade, vous pourriez souhaiter revenir au bon vieux temps où la seule chose à faire était de se soucier de bien comprendre les « spécifications ». Une époque où votre métier n'était pas de vraiment résoudre les problèmes et où vous deviez juste développer ce qu'on vous avait demandé, car c'était le problème de quelqu'un d'autre de s'assurer du bon produit à développer. Mais je crois que, tout comme vous, la plupart des gens aiment vraiment résoudre les problèmes si bien que c'est maintenant à vous de jouer.

Il vous est sans doute apparu que parmi toutes ces choses dont on parle, il fallait garder la trace d'un grand nombre d'informations. Mais comme tout ceci ne va pas tenir sur un Post-it® ou une fiche, nous allons dans le prochain chapitre parler de ce qu'il faut vraiment écrire sur une fiche.

8

Tout ne figure pas sur la fiche

La grande idée était que les titres courts des stories indiqués sur les fiches nous aideraient à planifier et à fluidifier les nombreuses conversations entre les développeurs et les gens qui comprennent les problèmes résolus par les logiciels. Mais, malheureusement, il faut plus de deux personnes pour finaliser un logiciel.

Dans une équipe typique, vous trouverez des chefs de projet, des chefs de produits, des analystes métiers, des testeurs, des concepteurs d'interface (*UX designers*), des rédacteurs techniques et quelques autres rôles que j'ai probablement oubliés. Ils ont tous sous les yeux la même fiche, mais leurs conversations sont différentes car ils ont tous des préoccupations différentes.

8.1 DIFFÉRENTES PERSONNES, DIFFÉRENTES CONVERSATIONS

Si je suis chef de produit ou PO, et que je suis responsable du succès d'un produit, alors je dois en savoir un peu plus sur mon marché cible. J'ai besoin de formuler quelques hypothèses sur le nombre de personnes qui vont acheter ou utiliser ce produit, ou sur la manière dont il va affecter la rentabilité de mon entreprise. Je veux parler de ces choses-là.

Si je suis analyste métier, je peux me plonger dans de nombreux détails, et j'ai donc besoin de comprendre le fonctionnement de l'interface utilisateur et les règles métiers sous-jacentes.

© Dunod – Toute reproduction non autorisée est un délit.

Si je suis testeur, j'ai besoin de penser aux endroits où le logiciel est susceptible de se planter. J'ai besoin d'avoir des conversations pour m'aider à mettre sur pied un bon plan de test.

Si je suis concepteur d'interface utilisateur, je ne veux pas des conseils sur l'aspect de l'interface utilisateur, pas plus qu'un développeur ne souhaite bénéficier de conseils sur la façon dont le code doit être écrit. Je veux savoir qui utilise le logiciel, pourquoi et comment, de manière à pouvoir concevoir une interface utilisateur utile et conviviale.

Enfin, si je suis chef de projet chargé de la coordination de ce groupe de personnes, je vais devoir faire attention à tous ces détails lorsque chacun d'entre eux parle pour prendre des décisions. J'aurai besoin de prêter attention aux dépendances, aux plannings et à l'état du développement.

Cela fait beaucoup de conversations, et certaines d'entre elles devront avoir lieu avant les autres. En outre nombreuses sont celles qui se auront lieu plus d'une fois. Mais heureusement, si vous avez des véritables conversations et arrivez à une compréhension mutuelle, vous éviterez beaucoup de malentendus et des réorientations.

> Pour chaque story, il y a beaucoup de types différents de conversations avec des gens différents.

8.2 IL NOUS FAUT UNE FICHE PLUS GRANDE

Quand vous lisez ce titre, j'espère que vous pensez à ce vieux film, *Les dents de la mer*, où, après avoir vu pour la première fois l'énorme requin de près, le chef de la police, Brody, dit à Quint : « *Il va vous falloir un plus gros bateau* ».

L'idée de départ était que je puisse prendre une fiche et écrire le titre dessus, puis, lors des conversations, que je puisse la retourner et écrire au verso les détails sur lesquels nous nous sommes mis d'accord. Que je puisse aussi dessiner l'interface utilisateur et écrire beaucoup d'autres informations sur la fiche. Sur certains projets, je peux vraiment travailler de cette façon-là. C'est en général le cas avec de petites

© Dunod – Toute reproduction non autorisée est un délit.

équipes travaillant en étroite collaboration et qui ont beaucoup de connaissances en commun. Il s'agit d'équipes qui n'ont pas beaucoup à écrire pour se souvenir des choses.

Je ne pense pas que Kent et les autres personnes qui ont mis au point le concept des stories imaginaient réellement que toutes ces conversations entre toutes ces personnes différentes puissent tenir sur une seule fiche, ce qui dans les faits n'est effectivement généralement pas le cas.

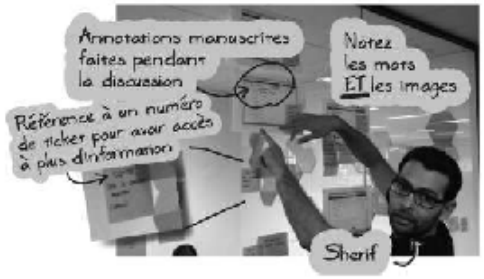
La métaphore qui fonctionne bien pour moi est celle d'une fiche d'un catalogue de bibliothèque, pour les personnes qui sont assez âgées pour se rappeler le temps où les bibliothèques avaient des catalogues rangés dans des bacs à fiches. Les stories écrites sur les fiches fonctionnent un peu comme ça.

Si je prends une fiche dans un catalogue de bibliothèque, elle contient juste assez d'informations pour que j'identifie le livre. Elle indiquera probablement le titre, le nom de l'auteur, une description, le nombre de pages du livre, la catégorie du livre, comme « fiction », et un code (vous vous souvenez de la classification décimale de M. Dewey ?) qui désigne un emplacement où je pourrai trouver un exemplaire du livre dans la bibliothèque. La fiche est juste une marque qui est facile à trouver et à organiser, et personne ne confond les fiches avec les livres. Le catalogue de fiches est pratique car il prend beaucoup moins d'espace que des milliers de livres. Et je peux organiser les fiches de différentes manières, par exemple, par auteur ou par thème.

Vos stories fonctionnent de la même manière, c'est-à-dire que vous pouvez les écrire sur des fiches, les conserver sous forme de liste dans une feuille de calcul, les saisir dans votre outil préféré de suivi ou bien dans l'outil de suivi que votre entreprise vous oblige à utiliser (vous savez, l'outil dont tout le monde se plaint). Dans une bibliothèque, vous savez qu'il y a quelque part un livre, et si vous avez identifié la bonne fiche dans le bac du catalogue, il est facile de le trouver. C'est pareil avec une story, et vous savez qu'il y a quelque part une quantité croissante d'informations qui se développe et évolue avec chaque conversation. Et quelle que soit la manière dont votre entreprise choisisse d'assurer le suivi des informations, nous espérons qu'elles sont faciles à retrouver.

Si vous voulez vraiment la jouer à l'ancienne, conservez les détails de toutes ces discussions en les enregistrant sur de grandes feuilles de papier que vous afficherez sur le mur si bien que vous pourrez continuer à en parler quand vous le voudrez. Mais n'oubliez pas qu'il faudra les conserver lorsque vous aurez terminé le travail ou quand vous manquerez de place sur le mur. Et il faut aussi les photographier et stocker ces photos pour la postérité.

Comment les créateurs d'outils ont de bonnes discussions de stories



Voici mon ami Sherif, qui est chef de produit dans une société appelée Atlassian. Atlassian produit *Confluence*, un wiki populaire utilisé dans une grande variété d'organisations pour garder la trace des connaissances qui s'accumulent. Cette société développe aussi JIRA, l'un des outils les plus populaires pour gérer les travaux en développement agile. Bien entendu, Atlassian utilise ses propres outils numériques pour conserver et partager l'information, mais il faut aussi apprendre comment avoir de bonnes conversations en tête à tête.

Quand je me promène dans les bureaux d'Atlassian à Sydney, je vois les murs couverts de Post-it®, de dessins et de copies d'écran. Si vous regardez attentivement, vous verrez que les Post-it® font référence à des numéros de tickets gérés dans les outils qu'utilise l'équipe. Il y a un mouvement incessant de va-et-vient entre les outils logiciels et l'espace physique. Lorsque Sherif me montre ce qu'ils stockent dans *Confluence*, je suis étonné par la combinaison de photographies, de courtes vidéos et de discussions.

8.3 RADIATEURS ET GLACIERES

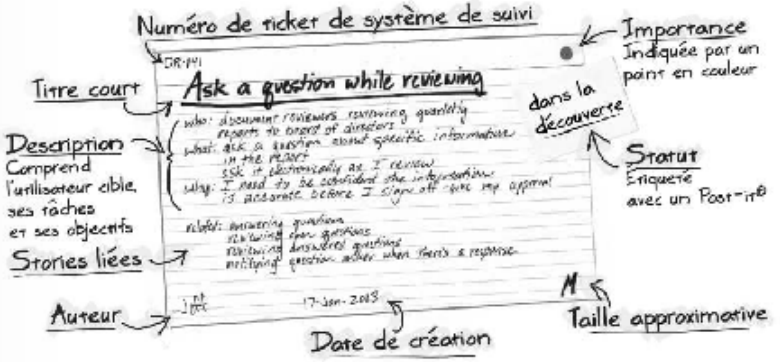
Dans son livre *Agile Software Development: The Cooperative Game* (Addison-Wesley Professional), Alistair Cockburn a créé le terme *radiateur d'informations* pour décrire à quel point les informations visibles sur le mur irradient des choses utiles dans la pièce. Les gens qui passent par là y jettent un coup d'œil et s'y intéressent. Lorsque l'information est vivante et utile, beaucoup de conversations se finissent devant le mur, où les gens peuvent pointer du doigt certains détails et compléter les informations qui s'y accumulent.

Quand je déambule dans des environnements où les murs sont nus, ou recouverts d'œuvres d'art, ou pire encore ornés d'affiches incitant à la motivation, ça me rend triste. On pourrait tellement mieux utiliser ces surfaces et notamment se servir des murs pour de riches activités collaboratives. Si certains appellent ce qui est sur le mur un radiateur d'informations, d'autres nomment *glacière d'informations* les outils qu'on utilise pour préserver l'information.

Ce qui est vraiment remarquable chez Atlassian, c'est qu'ils gardent l'information vivante et utile, que ce soit sur les murs ou bien dans les outils logiciels dont ils se servent.

Que trouve-t-on vraiment sur une fiche de story ?

Imaginez une fiche d'un catalogue de bibliothèque : elle contient des informations utiles pour faciliter la recherche et pour vous indiquer qu'il s'agit bien du livre que vous souhaitez. Une bonne fiche de story est un peu comme ça.



Parmi les choses courantes que l'on s'attend à trouver sur une fiche, on peut citer :

Titre abrégé – Il doit être facile à glisser dans une conversation lorsque vous abordez ce sujet. Un bon titre est la partie la plus précieuse de votre story. N'ayez pas peur de le reformuler s'il crée une certaine confusion.

© Dunod – Toute reproduction non autorisée est un délit.

Description – Une ou deux phrases qui décrivent ce que nous imaginons. C'est une bonne idée de décrire les « qui », les « quoi » et les « pourquoi » : qui utilise quoi ou qui en a besoin, que vont-ils en faire, et quels sont les avantages attendus ?

Lorsque vous commencez à discuter des stories, vous allez ajouter des informations qui résument vos discussions. Parmi ces informations, on peut lister :

Numéro de la story – Lorsque vous en avez toute une série ou que vous les enregistrez dans un système de suivi, cela vous aidera à les retrouver, un peu comme le système de classement décimal de Dewey dans une bibliothèque. Mais, quoi que vous fassiez, *s'il vous plaît*, ne commencez pas à vous référer à vos stories par leur numéro. Si vous le faites, c'est le signe que vous n'avez pas choisi un très bon titre. Même les bibliothécaires ne font pas référence à leurs livres par leur cote Dewey.

Estimation, taille ou budget – Lorsque vous commencerez à discuter de la story, vous allez avoir envie de prédire le temps nécessaire au développement du logiciel. Il y a beaucoup de termes pour désigner cela : *estimation*, *taille* ou *budget*. Employez le terme que votre société utilise.

Valeur – Vous pouvez avoir de longues discussions sur la valeur relative d'une chose par rapport à une autre. Vous pouvez adopter une échelle numérique pour noter vos fiches ou bien les annoter avec des mentions comme *importante*, *moyenne* ou *basse*.

Métrique – Si vous vous intéressez vraiment aux résultats, créez une métrique spécifique dont vous assurerez le suivi après avoir livré le logiciel afin de déterminer si c'est un succès.

Dépendances – Est-ce que d'autres stories sont en lien avec celle-ci ?

Statut – Est-ce prévu pour une release particulière ? Est-ce démarré, en cours de réalisation, ou bien terminé ?

Dates – À l'instar de la fiche d'un livre qui comporte la date de sa publication, vous pouvez enregistrer la date à laquelle cette story a été ajoutée, a démarré ou est terminée.

Vous pouvez griffonner tout autre commentaire que vous souhaitez sur la fiche. Vous pouvez aussi retourner la fiche et écrire des notes ou des critères d'acceptation sous la forme de liste à puces.

La seule chose vraiment nécessaire sur votre fiche est un bon titre. Toutes les autres informations peuvent être pratiques, mais c'est à vous et à votre équipe de décider ce que vous souhaitez utiliser.

On ne peut pas stocker trop d'informations sur la fiche et c'est une bonne chose. Rappelez-vous qu'il ne s'agit que d'un repère que vous utilisez pour la planification. Le fait de disposer de fiches physiques vous permet d'employer des mots très pratiques comme *ceci* et *cela* dans les conversations quand vous pointez du doigt des fiches disposées sur un mur ou sur une table. Vous ne pouvez pas faire cela avec un document épais. Des fiches, vous pouvez les étaler sur un bureau, les classer par ordre d'importance, les afficher sur un mur, et les agiter pendant que vous parlez afin d'attirer l'attention. Si vous le faites avec un document plus épais, vous pourriez blesser quelqu'un. Et, bien sûr, il est souhaitable de disposer de jeux de fiches sur les story maps pour raconter des stories encore plus grandes.

8.4 CET OUTIL N'EST PAS CONÇU POUR CELA

Un bûcheron rencontre un homme dans la forêt. L'homme travaille dur et tente d'abattre un arbre en le frappant avec un marteau. Le bûcheron arrête l'homme et lui dit : « *Vous n'utilisez pas le bon outil ! Essayez ceci…* » et il lui tend une scie. L'homme le remercie et le bûcheron continue son chemin, heureux d'avoir pu rendre service. L'homme commence alors à frapper l'arbre avec la scie de la même façon qu'il utilisait le marteau.

Cette blague nous rappelle que nous pouvons utiliser le *mauvais outil* pour un travail, mais que nous pouvons également *mal* utiliser le bon outil.

Quand je dis aux gens comment des entreprises comme Atlassian utilisent des outils, ils sont souvent surpris car ils ont essayé de se servir d'outils pour remplacer les tableaux blancs et les Post-it®. Et, comme prévu, ils ont eu des soucis. C'est peut-être qu'ils ont utilisé le mauvais outil pour la mauvaise tâche, ou bien mal utilisé l'outil. Pour comprendre ce qui a pu mal se passer, il est préférable d'examiner d'abord le travail et non pas l'outil.

8.4.1 Améliorez la compréhension mutuelle

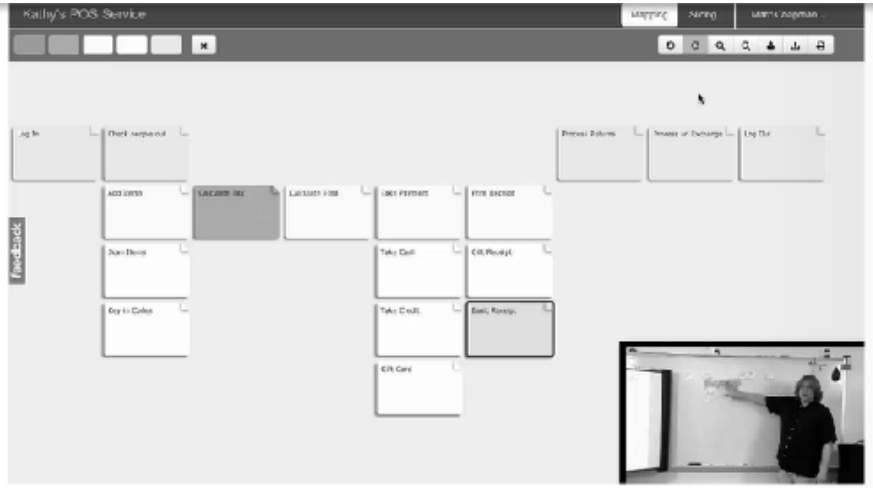
Lorsque l'on travaille ensemble pour raconter des stories et prendre des décisions sur les solutions à développer, le premier objectif est de mieux se comprendre. C'est à ce moment où l'extériorisation et l'organisation de vos pensées sont critiques. Et rien ne vaut le travail en tête à tête devant un tableau blanc, armé de Post-it®. Mais, si vous avez à accomplir cette tâche avec d'autres personnes qui travaillent à distance, c'est difficile. Les outils de vidéoconférence qui permettent de voir les visages des participants à la réunion ne sont pas d'une grande utilité, puisque ce ne sont pas leurs visages que l'on a besoin de voir, mais les idées que l'on place sur le mur ou sur la table.

> Utilisez une caméra de document ou une webcam lors d'une vidéoconférence pour permettre aux personnes à distance de voir ce qui est affiché sur le mur.

J'ai travaillé avec des équipes qui utilisaient des caméras pendant la visioconférence, et la réunion se concentrait sur les modèles en cours d'élaboration sur le mur, et non sur les visages des membres de l'équipe.

Si vous utilisez un outil de visualisation, c'est idéal si les gens des deux côtés de la réunion peuvent ajouter et déplacer les choses, tout comme ils feraient s'ils travaillaient ensemble avec un tableau blanc. Il s'agit d'une copie d'écran d'un outil appelé Cardboard.

© Dunod – Toute reproduction non autorisée est un délit.



La personne qui utilise Cardboard crée une carte en même temps que David Hussman (un des créateurs de Cardboard) cartographie sur un mur. Les autres qui partagent la même carte dans d'autres emplacements la voient s'élaborer en temps réel. Ils peuvent ajouter, supprimer et modifier les fiches, et tout le monde peut voir ce que tout le monde fait. Vous pouvez virtuellement « prendre du recul » et regarder la totalité du mur en un seul coup d'œil, ce qui est très pratique car les écrans d'ordinateur sont juste un petit portail sur ce que vous pourriez voir si vous travailliez sur un mur.

> Lors de la collaboration à distance, utilisez des outils qui permettent à chacun de voir, d'ajouter et d'organiser le modèle en même temps.

Heureusement, il y a de plus en plus d'outils qui autorisent le travail collaboratif pour forger la compréhension mutuelle, ce qui est une bonne chose.

8.4.2 Créez des souvenirs

Quand on a travaillé dur, et en groupe, sur le même document, il faut garder des copies des modèles ou des exemples qui ont été créés pour qu'elles servent de photos de vacances, ce qui aidera à se souvenir de tous les détails. Des outils comme *Confluence* de la société Atlassian offrent un riche wiki pour stocker non seulement des mots, mais aussi des photos et des vidéos. Prendre des photos et des vidéos après un travail en commun est l'un des moyens les plus rapides de documenter la collaboration.

Les gens de chez Atlassian font exactement cela : ils prennent une photo après avoir travaillé sur un mur et la déposent sur leur wiki pour en faire une sauvegarde.

> Utilisez des outils pour publier des photos, des vidéos et du texte, ce qui vous aidera à vous souvenir de vos conversations.

Personnellement, je préfère conserver l'information sur le mur, mais si je crains qu'une personne chargée du nettoyage détruise le travail, je le photographie au cas où. Si je partage des informations avec des gens qui ne peuvent pas être présents dans la pièce, même virtuellement, je tourne une courte vidéo du modèle sur le mur et je la poste dans un emplacement où d'autres personnes peuvent la voir.

8.4.3 Assurez le suivi

Une des choses dans laquelle les outils excellent, c'est le suivi de la progression de tout le travail qui a été planifié. Les outils sont performants pour assurer le suivi des données chiffrées qui sont pénibles à comptabiliser : quand avons-nous commencé, quand avons-nous terminé, combien reste-t-il à faire ? Les meilleurs outils vont générer des synthèses utiles au fur et à mesure que l'on assure le suivi systématique de ce qui est réalisé.

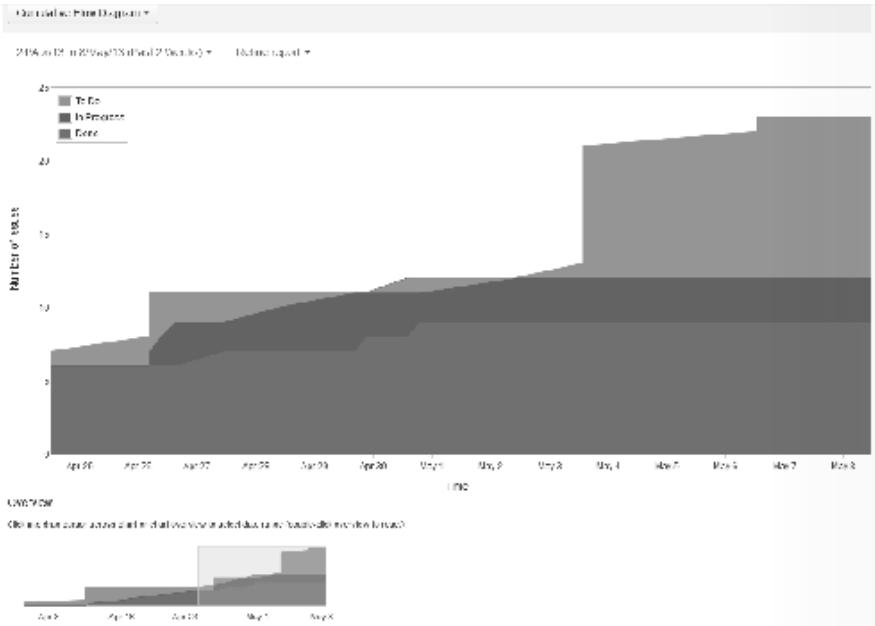
La copie d'écran qui suit illustre un diagramme de flux cumulé généré par JIRA, le produit de chez Atlassian. C'est un tableau qui montre les travaux que nous effectuons et leur état au fil du temps. C'est le genre de graphique que je n'aimerais pas avoir à produire à la main.

Pour les équipes uniques et les petits projets, le mur fera très bien l'affaire, mais, si vous avez des équipes plus grandes qui travaillent sur des sites différents et des projets qui durent longtemps, utilisez un outil pour garder la trace de tous les détails.

> Utilisez des outils pour classer, suivre et analyser la progression.

Toute l'astuce consiste à utiliser le bon outil pour la bonne tâche. N'essayez pas d'utiliser un outil de suivi vraiment génial pour forger la compréhension mutuelle. Et ne vous acharnez pas à faire des analyses complexes sur un tableau blanc.

© Dunod – Toute reproduction non autorisée est un délit.



En développement agile, l'idéal est de conserver les choses simples et rapides, et de continuer à travailler avec des fiches et des tableaux blancs autant que possible. Et je vous promets que si vous pouvez éviter le gigantisme, la lourdeur et les outils inutiles, vous serez plus heureux. Mais n'oubliez pas que ces outils ne sont qu'un moyen pour arriver à une fin. Dans le chapitre suivant, nous parlerons de ce qui se passe après la fiche.

9

La fiche n'est que le commencement

Les trois C[1] ne sont que le commencement.

Je sais que j'ai dit plus haut que je ne me sentais pas obligé de citer le *Manifeste agile*, mais je vais quand même le faire, tout au moins une petite partie. Une des déclarations du manifeste stipule que « *le logiciel opérationnel prime sur une documentation exhaustive* ». Je pourrais reformuler ceci en disant que « *le logiciel opérationnel prime sur une conversation exhaustive* », et le sens serait identique. Toutes ces conversations, et la documentation qui aide à nous en souvenir, ne sont qu'un moyen d'arriver à une fin. Au bout du compte, nous aurons besoin de développer quelque chose.

Si nous terminons le cycle, le modèle ressemble à ceci :

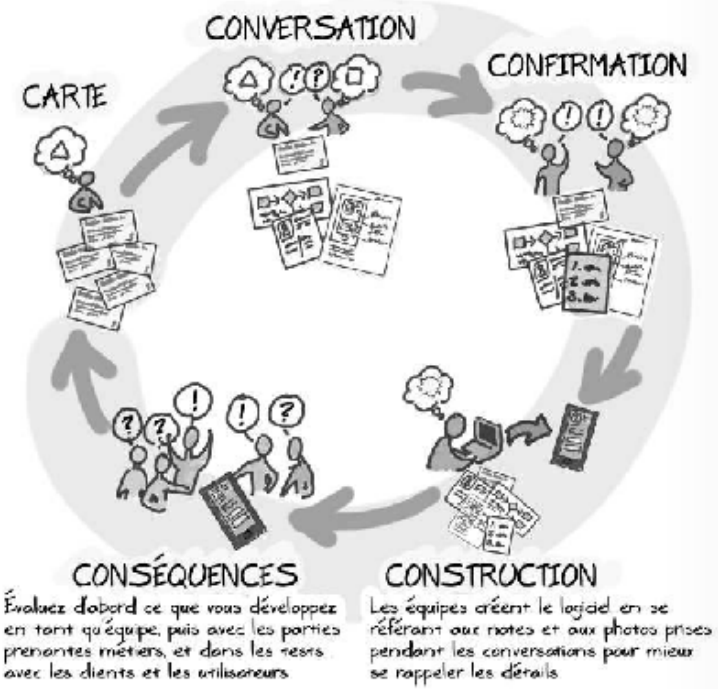
Il y a cependant quelques problèmes qui parviennent toujours à s'inviter dans ce modèle, une fois que nous avons forgé une compréhension mutuelle et que nous sommes tombés d'accord sur ce que nous allons développer.

9.1 CONSTRUISEZ AVEC UNE IMAGE CLAIRE EN TÊTE

Après avoir eu des conversations et rédigé les détails qui aideront à nous en souvenir, après avoir donné notre accord sur les choses à vérifier pour confirmer que le produit est bien fini, nous sommes enfin prêts à faire quelque chose :

1. Ndt : les trois C sont la carte, la conversation et la confirmation, mais nous avons habituellement choisi de traduire le mot anglais *card* par fiche pour éviter la confusion avec le terme anglais *map* que nous traduisons par carte. Pour garder les 3 C, nous avons choisi, dans ce contexte, de traduire *card* par carte, mais il s'agit bien d'une fiche.

© Dunod – Toute reproduction non autorisée est un délit.



- Les développeurs peuvent commencer la création du logiciel.
- Les testeurs peuvent créer des plans de test et commencer à tester.
- Les concepteurs d'interface utilisateur peuvent créer le détail des interfaces et des ressources numériques, si cela n'est déjà fait, afin d'arriver à une compréhension mutuelle.
- Les rédacteurs techniques peuvent écrire ou mettre à jour les fichiers d'aide ou tout autre document.

Le plus important ici est que tout le monde possède *la même image en tête, celle qui a été forgée lors des conversations*.

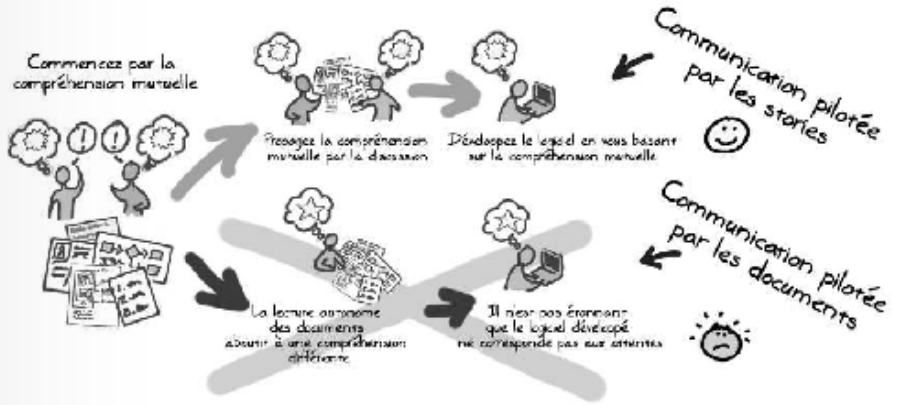
À présent, faites particulièrement attention à ce qui suit.

> Le fait de confier tous les détails de la story à développer à quelqu'un d'autre ne fonctionne pas. Évitez de le faire !

Si vous avez travaillé avec un groupe de personnes pour comprendre ce qui doit être développé, et si vous avez documenté toutes les choses importantes que quelqu'un doit savoir pour en assurer le développement, vous pouvez être tenté de transmettre ces données à quelqu'un d'autre. Après tout, quand vous regardez ces informations, elles vous paraissent limpides. Mais ne vous méprenez pas : elles sont claires pour vous parce que votre cerveau est suffisamment intelligent pour combler tous les détails qui n'ont pas été rédigés. Votre cerveau est tellement bon dans cette activité qu'il vous est difficile de détecter ce qui pourrait manquer. N'oubliez pas que ces détails sont *vos* photos de vacances, pas les *leurs*.

9.2 CONSTRUISEZ UNE TRADITION ORALE DU STORYTELLING

Partager des stories est assez simple. Quelqu'un qui comprend la story et les informations recueillies qui aident à la raconter, n'a besoin que de très peu de temps pour raconter la story à la personne suivante qui doit la connaître. Cela devrait normalement aller beaucoup plus vite que lors des premières conversations où vous avez travaillé en groupe pour prendre des décisions difficiles, car heureusement vous n'aurez pas besoin de refaire tout ça. Vous allez utiliser ce qui a été écrit pour faciliter la narration de la story, parler, montrer du doigt les images, puis laisser votre interlocuteur poser des questions et apporter des modifications aux images qui aident à ne pas oublier. Aidez-le à transformer les informations associées à la story en photos de vacances.



Il y a un parfait anti-pattern que je rencontre souvent. Certains pensent que dans la mesure où n'importe quel membre d'une équipe peut travailler sur la story, tous les membres de l'équipe devraient être impliqués dans toutes les conversations. Vous travaillez peut-être dans ce type d'entreprise où vous entendez beaucoup de gens se plaindre qu'il y ait beaucoup trop de réunions. Par ailleurs, le terme « réunion » est souvent un euphémisme que l'on emploie pour désigner une collaboration improductive.

Les discussions et les prises de décision efficaces se déroulent mieux avec des petits groupes composés de deux à cinq personnes. Vous savez que si vous avez un groupe d'amis à dîner, il est plus facile de tenir une conversation quand ils ne sont pas trop nombreux. Au-delà de cinq personnes, la conversation devient plus difficile.

© Dunod – Toute reproduction non autorisée est un délit.

Faites travailler ensemble de petits groupes pour qu'ils prennent des décisions et poursuivez ensuite les conversations pour partager les résultats avec tout le monde.

9.3 INSPECTEZ LES RÉSULTATS DE VOTRE TRAVAIL

Si vous faites partie d'une équipe, vous allez tous travailler ensemble armés de la compréhension mutuelle de ce que vous développez. Au cours de votre travail, vous allez continuer à avoir des conversations parce que l'on ne pense jamais à tout du premier coup.

Quand le logiciel sera terminé, vous vous réunirez à nouveau pour en parler. C'est le moment idéal pour se féliciter d'un travail bien fait et constater de réels progrès. Dans le développement traditionnel de logiciels, les possibilités de voir les résultats de son travail acharné sont beaucoup moins fréquentes et elles sont rarement partagées au sein d'une équipe. Dans un processus agile typique comme Scrum, vous allez avoir un moment de partage toutes les deux semaines, à la fin d'un sprint, lors de la revue. Dans les équipes qui fonctionnent le mieux, les membres se réunissent fréquemment pour inspecter leur travail quand il est terminé. Mais vous aurez besoin d'aller au-delà des démonstrations et des discussions. Après les félicitations, prenez le temps d'une réflexion courte mais sérieuse sur la qualité du travail réalisé.

Quand on parle de qualité, j'entame les discussions selon trois axes :

- **Qualité de l'expérience utilisateur** – Examinez le produit du point de vue de l'utilisateur cible. Est-il facile à utiliser ? Est-il *convivial* ? Est-il compatible avec votre marque et d'autres fonctionnalités ?
- **Qualité fonctionnelle** – Le logiciel fait-il ce qui était convenu sans bugs ? Des testeurs et d'autres membres de l'équipe ont heureusement passé du temps à le tester et vous avez déjà corrigé des erreurs, mais les bons testeurs disent souvent qu'il y a probablement encore beaucoup de bugs dans le produit qui sont susceptibles d'apparaître plus tard. Dans le meilleur des cas, on vous dira que le logiciel est parfaitement robuste.
- **Qualité du code** – Le logiciel est-il de bonne qualité, conforme aux normes ? Il est possible que le logiciel soit utilisé pendant un certain temps, et il serait donc bon de savoir s'il sera facile d'y ajouter des fonctionnalités et de le maintenir, ou bien si nous avons accumulé une dette technique que nous aurons besoin de rembourser plus tard.

À ce stade, j'ai de mauvaises nouvelles à vous annoncer. Vous êtes susceptible de trouver des choses que vous allez devoir changer.

Pour clarifier les choses, il convient de séparer les deux problèmes. Premièrement : est-ce que l'on a développé ce qui était convenu ? Et deuxièmement : si c'est bien ce qui était convenu, devrions-nous y apporter quelques modifications maintenant que nous en prenons connaissance ?

Tout le monde a travaillé dur afin de comprendre d'abord ce qu'il fallait développer pour résoudre les problèmes des utilisateurs. Vous avez ensuite fait de votre mieux pour identifier les éléments à vérifier lors de la recette. Vous avez vérifié toutes ces choses et vous avez félicité votre équipe pour avoir accompli tout ce travail. Vous avez finalement obtenu exactement ce que vous souhaitiez obtenir.

Cela me rappelle une vieille chanson des Rolling Stones dont les paroles sont : « *Tu ne peux pas toujours avoir ce que tu veux. Mais si tu essayes, tu risques de trouver que tu as tout ce qu'il te faut.* » L'ironie avec le logiciel, c'est que c'est exactement le contraire.

Vous travaillerez ensemble pour vous mettre d'accord sur ce que vous voulez. Et si vous travaillez avec une équipe compétente, vous verrez que vous pourrez souvent obtenir ce que vous voulez. Mais c'est seulement après avoir vu le résultat final que vous pourrez mieux évaluer si c'est ce dont vous aviez besoin. C'est pénible, mais vous n'avez rien à vous reprocher car c'est comme cela que ça fonctionne.

Vous avez cependant un moyen de réparer les choses : commencez par écrire sur une fiche ce qu'il faut changer dans le logiciel pour l'améliorer. Ceci est bien entendu pénible si vous pensiez que tout marcherait du premier coup. Peut-être que Mick Jagger avait finalement raison : ce dont vous aviez besoin, c'était d'apprendre que réussir la première fois est une stratégie risquée, en particulier en informatique.

9.4 CE N'EST PAS POUR VOUS

J'en suis désolé, mais j'ai des nouvelles encore plus mauvaises.

En réalité, la personne qui, à l'origine, a écrit la fiche et a commencé ce cycle complet n'est probablement pas la personne qui utilisera le logiciel quotidiennement. La personne qui a rédigé la fiche et collaboré avec toute l'équipe croit peut-être qu'ils ont réussi à développer la solution parfaite pour leurs utilisateurs cible.

Mais ne vous y trompez pas.

Si nous faisons partie de cette équipe et que nous sommes sensés, nous allons présenter le logiciel aux utilisateurs et le tester avec eux. Il ne s'agit pas d'une simple démonstration et d'un beau discours. Nous allons faire un test en les regardant utiliser le logiciel pour atteindre un objectif réel qu'ils vont normalement accomplir à l'aide du logiciel.

Est-ce que vous vous êtes déjà assis à côté de quelqu'un quand il utilise le logiciel que vous avez contribué à développer ? Pensez à la première fois où vous l'avez fait. Comment ça s'est passé ? Je n'étais pas présent dans la pièce avec vous à ce moment-là, mais je suis prêt à parier que cela ne s'est pas passé comme prévu.

Si vous vous êtes déjà assis à côté d'un utilisateur quand il utilise votre produit, vous savez ce que je veux dire. Si vous ne l'avez jamais fait, c'est le moment de le faire !

Vous allez avoir besoin de faire régulièrement des tests avec les personnes qui vont effectivement acheter, adopter et utiliser votre produit. En général, j'attends d'avoir développé assez de logiciel afin qu'elles l'utilisent pour pouvoir accomplir quelque chose qu'elles ne pouvaient pas faire avant. Quelle que soit la fréquence que vous

© Dunod – Toute reproduction non autorisée est un délit.

adoptez, ne laissez pas passer plus de deux semaines sans voir un véritable utilisateur interagir avec le logiciel.

Tous les membres de l'équipe n'ont pas besoin d'être présents avec les utilisateurs car cela les inquiéterait, mais votre présence va créer de l'empathie que vous n'obtiendrez pas d'une autre manière. C'est un puissant facteur de motivation de voir des gens qui s'escriment à utiliser votre produit, surtout si vous pensiez que cela n'était pas nécessaire. Si vous faites cette expérience, partagez ce que vous avez vu avec les autres membres de l'équipe en leur racontant des stories.

Après avoir effectué des tests avec les utilisateurs, vous allez identifier les problèmes à régler et trouver des moyens évidents pour améliorer le logiciel. Et pour chacune de ces choses, vous devez écrire vos idées sur une fiche de story pour améliorer le logiciel.

9.5 DÉVELOPPEZ POUR APPRENDRE

Si vous avez travaillé avec la conviction que l'emploi de stories améliorerait la qualité du logiciel, vous avez à moitié raison. En fait, toutes les conversations entre des gens intelligents visent à comprendre le problème et la manière de le résoudre, mais le chemin est encore long pour élaborer un bien meilleur produit. Il faut cependant reconnaître que développer des logiciels n'est pas la même chose que de travailler sur une ligne d'assemblage. On ne développe pas juste un widget de plus comme celui que l'on a créé quelques minutes auparavant. Chaque nouvelle story qui sert de support au développement d'un logiciel est quelque chose d'original.

Mon ami Alistair Cockburn, dont j'ai déjà parlé et qui est l'une des sommités de la communauté du développement agile, m'a un jour confié que *« pour chaque histoire rédigée, il faut en écrire trois dans le backlog de stories »*.

Je lui ai demandé pourquoi, et il m'a répondu : « *Contente-toi de le faire.* »

J'ai alors demandé : « *Que dois-je écrire sur les deux autres ?* »

« *Peu importe ce que tu écris.* »

« *Que veux-tu dire ?* » ai-je demandé, « *Il faut bien que j'écrive quelque chose !* »

Alistair m'a répondu : « *Eh bien, si tu dois écrire quelque chose, alors écris ce que tu veux sur la première fiche, puis écris sur la deuxième fiche* « Réparer la première fiche ». *Ensuite, écris sur la troisième fiche* « Réparer la deuxième ». *Si tu n'effectues pas ce cycle trois fois pour chaque story, tu n'apprendras rien.* »

Dans un processus traditionnel, l'apprentissage renvoie à la *dérive des objectifs* ou bien à de *mauvaises spécifications*, mais dans un processus agile, l'apprentissage est le but. Vous aurez besoin de planifier l'apprentissage pour tout ce que vous développez et vous allez devoir prévoir de vous tromper assez souvent.

La stratégie qu'Eric a utilisée au chapitre 3 l'a aidé à développer des solutions plus petites et à continuer leurs itérations jusqu'à ce qu'elles soient viables. Eric s'est appuyé sur l'apprentissage à chaque version.

La stratégie *Mona Lisa* utilisée par Mike et Aaron au chapitre 4 leur a permis de décomposer chaque story en parties plus petites afin qu'ils puissent apprendre plus vite et gérer judicieusement leur livraison pour finir dans les délais.

Voici deux stratégies remarquables d'apprentissage et vous devriez les essayer ou bien inventer les vôtres. Mais s'il vous plaît, ne présupposez pas que vous avez toujours raison car je vous promets que vous serez déçu.

9.6 IL NE S'AGIT PAS TOUJOURS DE LOGICIEL

En 2011, Kent Beck, le créateur du concept de story, a inauguré une des premières conférences Lean Startup avec sa révision du *Manifeste agile*. Si je l'avais fait moi-même, cela aurait pu être considéré comme un blasphème, mais c'est l'un de ses créateurs, il doit donc savoir ce qu'il fait. Il a révisé le manifeste en ces termes :

« L'apprentissage validé prime sur le logiciel opérationnel (ou une documentation exhaustive). »

Si vous vous souvenez du chapitre 3, l'apprentissage validé (*validated learning*) est le concept fondamental hérité du processus Lean Startup. Dans ce contexte, le mot-clé est l'apprentissage qui se transforme en *apprentissage validé* quand on discute de ce que l'on veut apprendre dans le cadre d'une création et puis quand on revient en arrière en prenant en considération les *conséquences*, qui reflètent ce que l'on a appris ou pas appris. On se rend alors compte que l'on ne doit pas toujours développer du logiciel pour apprendre, mais il faut généralement fabriquer ou faire quelque chose.

J'aime utiliser des stories pour conduire le travail que nous faisons pour développer des prototypes simples ou planifier les travaux que nous effectuons pour interviewer ou observer des utilisateurs. J'aime également parler des « qui », des « quoi » et des « pourquoi » pour ces choses. J'aime que l'on se mette d'accord sur ce que l'on va faire avant de le réaliser. Et je repense aux conséquences de la réalisation pour étudier ce que nous avons appris.

> Essayez d'utiliser des stories pour piloter la réalisation de quoi que ce soit, qu'il s'agisse ou non de logiciel.

9.7 PLANIFIEZ POUR APPRENDRE ET APPRENEZ À PLANIFIER

Les story maps sont utiles pour décomposer nos idées de grands produits ou de features en parties plus petites. Dans les chapitres 3 et 4, nous avons vu comment découper ces parties plus petites en morceaux susceptibles d'être développés où chaque morceau porte principalement sur l'apprentissage de quelque chose. Mais vous devez savoir qu'il y a une autre façon de décomposer les choses et ce sera l'objet du prochain chapitre.

© Dunod – Toute reproduction non autorisée est un délit.

10

Mitonnez vos stories comme des gâteaux

Il y a deux semaines, c'était l'anniversaire de ma fille et nous voulions un gâteau. Nous avons fait appel au talent culinaire de Sydnie qui fait des gâteaux délicieux.

Pour commander le gâteau, j'ai appelé Sydnie qui m'a demandé pour qui était le gâteau et pour quelle occasion c'était. Ma fille Grace allait avoir 12 ans et nous avons parlé de ses goûts. Nous avons aussi parlé de formes de moules à gâteaux et de types de gâteaux susceptibles d'être prêts à temps. Nous sommes finalement tombés d'accord sur un gâteau en forme d'oiseau.

Raconter une story fonctionne de la même manière. Sydnie a posé beaucoup de questions sur les « qui », les « quoi » et les « pourquoi ». Elle a demandé à connaître le contexte (où et quand servirions-nous le gâteau, combien de personnes seraient là). Pendant les conversations, nous examinons plusieurs options différentes et nous parlons assez longtemps pour forger la compréhension mutuelle. Et comme nous avons déjà commandé beaucoup de gâteaux à Sydnie, nous avions une compréhension mutuelle sur l'aspect et le goût de ses gâteaux. Si cela n'avait pas été le cas, nous aurions aimé voir quelques photos ou bien goûter un de ses gâteaux, et le téléphone n'aurait pas été d'une grande utilité pour cela.

10.1 CRÉEZ UNE RECETTE

Au cours de notre discussion, Sydnie pense à la manière dont elle va réaliser le gâteau car elle doit déterminer si elle peut le faire dans les temps. Quand viendra le temps de la confection du gâteau, elle aura une liste de choses qu'elle devra faire, comme mesurer la farine, le sucre, le beurre, les œufs et le lait. Elle devra mélanger, cuire,

© Dunod – Toute reproduction non autorisée est un délit.

décorer et probablement effectuer quelques autres étapes secrètes dont je n'ai aucune idée. Je pense qu'elle a des recettes différentes pour ses différents types de gâteaux et une liste des tâches à réaliser pour chaque gâteau avant de le glisser dans une boîte, prêt à être emporté. Si Sydnie a rédigé la liste de toutes les choses qu'elle doit faire, elle a un programme de travail plein de tâches spécifiques à la confection des gâteaux.

La même chose se produit lorsque quelqu'un communique une story à une équipe de développement. Ensemble, ils prennent des décisions sur ce qu'il faut spécifiquement développer, et l'équipe de développement crée son programme de travail, qui est composé de nombreuses tâches de développement. L'équipe de développement comprend des testeurs, des concepteurs d'interface utilisateur, des rédacteurs techniques, ou toute personne dont les compétences sont nécessaires pour créer le logiciel, si bien que les tâches ne concernent pas toutes le codage. Et, tout comme Sydnie n'avait pas créé son projet alors qu'elle me parlait au téléphone, l'équipe de développement ne créera probablement pas son projet pendant la conversation de la story. Mais ils vont écouter, prendre des notes, dessiner et recueillir beaucoup de détails dont ils auront besoin pour créer leur projet. C'est du moins la façon dont nous espérons qu'ils procèdent.

Quand je parle avec Sydnie, je ne raconte pas des stories sur les tasses de sucre et de farine. Je ne raconte pas de story de pâtisserie sauf si mon but est de construire un four. Lorsque vous racontez des stories sur un logiciel et répertoriez les noms des stories au fur et à mesure, vous racontez la story en imaginant le logiciel que vous aurez à la fin. Et vous ne faites pas qu'*imaginer* le logiciel, vous pensez à l'utilisateur et vous évoquez ses motivations. Sydnie ne s'est pas contentée des détails sur le gâteau, mais elle m'a demandé pour qui il était, ce que ma fille aimait, combien de personnes seraient à la fête, et beaucoup d'autres informations pour nous aider à définir ensemble le gâteau idéal. Sydnie n'a pas juste posé des questions sur les spécifications du gâteau, mais nous avons travaillé ensemble pour décider de la meilleure façon de confectionner un gâteau qui plairait à tout le monde. C'est le véritable esprit qui se cache derrière une conversation de story.

10.2 DÉCOUPAGE D'UN GROS GÂTEAU

Souvent, quand nous commençons à raconter la story à des gens qui sont réellement capables de concrétiser notre vision, nous nous rendons rapidement compte que la feature que décrit notre story est *vraiment* gros, ce qui pose un problème. Pourtant, la fiche sur laquelle il est décrit a la même taille que les autres. Et le but que nos utilisateurs essaient d'atteindre avec le logiciel n'est peut-être pas plus important pour eux que d'autres objectifs, mais c'est quand nous en parlons que nous nous rendons compte qu'il va falloir beaucoup de temps pour écrire le logiciel qui répondra à cet objectif.

La même chose aurait pu m'arriver quand je parlais avec Sydnie de mon gâteau. J'aurais pu imaginer un gâteau sophistiqué qui nécessite des moules à gâteaux que Sydnie ne possède pas, ou bien demander des techniques de décoration qu'elle ne maîtrise pas. Le résultat aurait été un gâteau que je ne pouvais pas m'offrir et un gâteau que Sydnie n'aurait pas pu me livrer comme prévu pour l'anniversaire de ma fille.

Dans le chapitre 7, j'ai montré que lorsque la solution à laquelle nous pensons est trop coûteuse, nous devons prendre du recul et étudier vraiment les problèmes que nous tentons de résoudre, et l'effet que nous essayons d'atteindre. Nous devons aussi envisager d'autres solutions de rechange. C'est une façon d'obtenir un gâteau plus petit, ou peut-être une tarte...

© Dunod – Toute reproduction non autorisée est un délit.

Si la story décrit une solution trop coûteuse, envisagez une autre solution qui vous permette d'atteindre l'objectif.

Mais, si ce que l'on veut vraiment est gros et que l'on peut se l'offrir, alors il n'y a aucune raison de tout décomposer en parties plus petites, n'est-ce pas ? Eh bien, en fait, si ! Particulièrement avec un logiciel car en développant des choses en parties plus petites, on peut voir et mesurer les progrès accomplis plus tôt. Cela aide les gens qui financent le projet à se sentir un peu moins nerveux. Et, comme dans la stratégie de *Mona Lisa* qui est décrite au chapitre 4, cela aide les gens qui fabriquent le produit à évaluer les différentes parties, ce qui leur garantit qu'ils sont sur la bonne voie.

Si la story possède une solution abordable, mais est de grande taille, divisez-la en parties plus petites qui vous permettront d'évaluer et de voir les progrès plus rapidement.

Il y a un truc pour décomposer les grandes stories, et la métaphore du gâteau m'aide à le garder à l'esprit. Si vous aimez les gâteaux, il se peut que vous commenciez à avoir faim, surtout si le gâteau que vous êtes en train d'imaginer est particulièrement appétissant. Croyez bien que j'en suis sincèrement désolé.

Admettons que notre story décrive le besoin d'un gâteau beaucoup plus grand, comme un gâteau de mariage géant pouvant nourrir des centaines de personnes. Si tel est le cas, on ne parle plus de quelques tasses de farine et du sucre, mais de *sacs* de farine et de sucre. La plupart des gens décomposent un logiciel de la même manière. Au lieu d'un peu d'interface utilisateur, un peu de logique métier et une petite base de données, il y a une grosse quantité de chacun des ingrédients. Mais n'oubliez pas que le logiciel n'est pas un gâteau. Cela ne prend pas beaucoup plus de temps de mesurer deux kilos de farine que de mesurer deux tasses de farine, alors que développer l'interface utilisateur pour vingt écrans prend beaucoup plus de temps que pour seulement deux écrans. Dans ces conditions, si les équipes utilisent la structure de répartition simple qui semble logique, ils sont tentés de décomposer le logiciel en semaines de développement de front end, en semaines de développement de logique métier, et ainsi de suite. Quand on utilise cette stratégie, cela prend beaucoup de temps avant de pouvoir « goûter un gâteau », si je puis dire. Alors ne faites pas comme ça.

Ne décomposez pas les grandes choses en grands projets. Décomposez les grandes choses en petites choses à l'aide de petits projets.

À présent, la métaphore va vraiment s'effondrer, mais je réclame encore une minute de votre attention. Pour gérer un gros gâteau logiciel, il faut le décomposer en nombreux petits gâteaux. Chacun d'entre eux constitue un livrable, et chaque gâteau a une recette similaire, avec un peu de sucre, un peu de farine, un œuf ou deux et ainsi de suite.

© Dunod – Toute reproduction non autorisée est un délit.

Revenons maintenant à des choses sérieuses. Le logiciel *n'est pas* un gâteau. Il peut devenir énorme, horriblement cher et horriblement risqué. Au moment où j'écris ces lignes, je viens encore d'entendre aux infos une histoire de défaillance d'un site web gouvernemental. Il est facile de critiquer après coup, mais c'est aussi facile de constater que personne n'a goûté ce gâteau avant qu'il ne soit servi lors du mariage métaphorique, du moins pas quelqu'un qui ait du goût. Et ce gâteau à moitié cuit a gâché la fête.

Si vous avez déjà travaillé pendant un certain temps dans une société de développement logiciel plus traditionnelle, vous avez probablement appris, comme moi, à décomposer les gros logiciels en gros projets. Cela peut sembler contraire à la logique de décomposer en parties plus petites, ce qui ne ressemblera pas tout à fait au produit fini que vous essayez de livrer. Vous savez que lorsque vous combinerez ces briques de logiciel, vous aurez à faire un peu de réécriture et des ajustements pour réassembler les différents morceaux. Mais n'oubliez pas que vous pensez de cette façon-là pour un grand nombre de bonnes raisons, notamment pour éviter les risques liés à des tests trop tardifs du logiciel. Vous décomposez les grandes choses en petites parties évaluables pour pouvoir apprendre plus tôt.

Si je devais décomposer un gâteau dans le but de le déguster plus tôt ou de voir sa décoration plus tôt, l'idéal serait de faire cuire des petits gâteaux qui m'aideraient à apprendre plus rapidement. J'en confectionnerais un certain nombre avec des saveurs différentes de manière à pouvoir tous les goûter, et choisir celui que j'ai préféré, ce qui me garantirait d'avoir fait le bon choix. Si j'étais préoccupé par les couleurs et la

décoration, je regarderais différents petits gâteaux décorés dans des styles différents et je choisirais le plus joli.

Si l'on compare avec le logiciel, les petits gâteaux sont les parties du logiciel opérationnel qui permettent aux utilisateurs de vérifier qu'ils peuvent accomplir efficacement leur tâche. Il peut aussi s'agir de parties de logiciel qui permettent de révéler un risque technique, mais chaque partie nous aide à apprendre quelque chose.

Mais un tas de petits gâteaux ce n'est pas la même chose qu'un gros gâteau, n'est-ce pas ?[1]

Le logiciel n'est pas un gâteau et chaque morceau de logiciel que nous développons se combine en un produit fonctionnel plus grand, et ce n'est pas comme cela que l'on confectionne un gâteau.

Mon ami Luke Hohman a une maxime idiote qui dit que l'on peut livrer « *la moitié d'un gâteau cuit, mais pas un gâteau à moitié cuit* ». La moitié d'un gâteau cuit peut ne pas suffire à nourrir les invités d'un mariage, mais tout le monde pourra le goûter et aura envie de l'autre moitié.

1. Gâteau de mariage de Mary (photo publiée avec l'aimable autorisation de Mary Treseler).

11

Cassez les blocs

L'idée originale des stories était assez simple : noter quelque chose sur une fiche, en discuter puis se mettre d'accord sur ce qu'il faut développer. Ensuite, terminer le cycle en développant et en apprenant à partir de ce qui a été développé. Si vous travaillez dans le développement de logiciels, même depuis peu de temps, vous savez que la réalité n'est pas aussi simple. Les stories parcourent un long voyage, peuplé de nombreuses conversations impliquant beaucoup de gens, pour transformer une idée en produit, en feature ou en amélioration, puis pour sortir ce produit sur le marché. La bonne nouvelle est que vous pouvez utiliser des stories tout au long du parcours car je vous promets qu'en vous appuyant sur cette pratique, vous allez vous simplifier la tâche sur l'ensemble du processus.

11.1 LA TAILLE A TOUJOURS DE L'IMPORTANCE

J'ai terminé le dernier chapitre en parlant des gâteaux de Sydnie et en avançant l'idée de décomposer les grands gâteaux en gâteaux plus petits. Mais le logiciel est beaucoup moins concret et sa taille ne peut pas être mesurée en centimètres ou en grammes comme pour un gâteau.

L'idée originale était qu'un utilisateur ou une personne qui avait besoin de quelque chose pouvait noter ce qu'il lui fallait sur une fiche et ensuite en discuter lors d'une conversation. Mais cette personne ne savait pas comment exprimer son besoin sous la forme de quelque chose qui prendrait peu de temps à développer.

> Du point de vue de l'utilisateur, une story de taille appropriée est celle qui répond à un besoin.

© Dunod – Toute reproduction non autorisée est un délit.

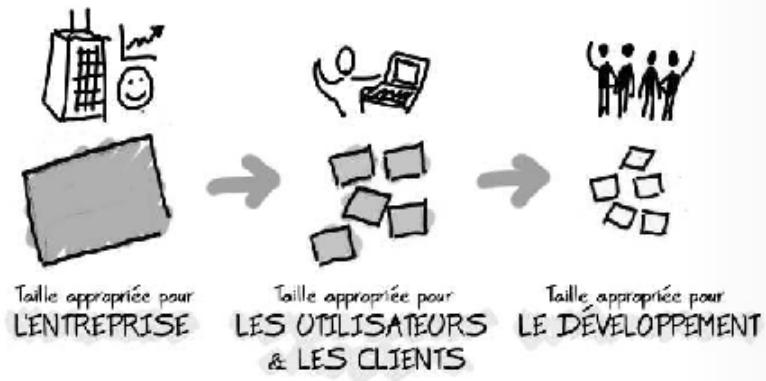
Quand c'est le moment d'écrire les logiciels, il est de loin préférable d'écrire, de tester et d'intégrer des petites portions de logiciel. Si je peux tester des petites parties plus tôt, je peux mesurer à quelle vitesse nous développons et la qualité obtenue. Si je peux diviser quelque chose de grand en de nombreuses parties plus petites, il est un peu plus facile pour mon équipe de développer des parties en même temps. Une bonne règle de base consiste à décomposer pour obtenir une story qui prend deux ou trois jours à développer et à tester.

Du point de vue de l'équipe de développement, une story de taille appropriée est celle qui ne prend que quelques jours à développer et à tester.

Mais, d'un point de vue industriel, il peut paraître plus logique de livrer aux clients et aux utilisateurs des lots de plusieurs features. Si vous livrez un tout nouveau produit, ce premier lot peut être assez volumineux. Il s'agit de ce que j'ai appelé précédemment une *solution viable minimum* qui vise à atteindre un avantage spécifique pour un groupe cible d'utilisateurs. Idéalement, les entreprises devraient s'efforcer d'en livrer plus fréquemment pour qu'elles correspondent mieux aux besoins des utilisateurs ou à l'impact de l'entreprise. Mais si la clientèle de votre produit est nombreuse et diversifiée et si vous n'avez pas un modèle commercial ou une infrastructure qui prend en charge un processus de livraison plus régulier, alors vos versions seront sans doute plus volumineuses.

Du point de vue de l'entreprise une story de taille appropriée est celle qui l'aide à obtenir un avantage en termes de business.

Je pourrais dire il n'y a pas de « bonne taille » pour les stories, mais ce n'est pas vrai. La bonne taille est la taille qui convient à la conversation que vous avez.

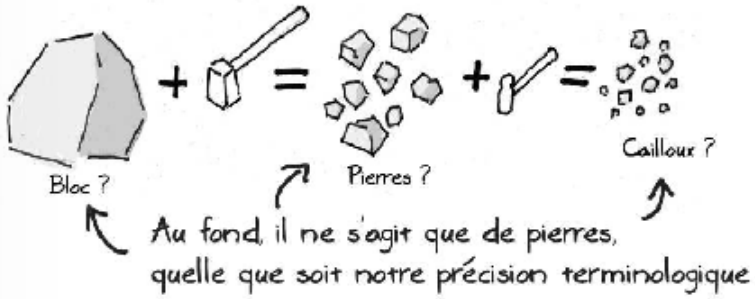


Ces grandes stories contiennent beaucoup de petites stories, qui à leur tour contiennent de nombreuses stories plus petites. Selon à qui vous parlez, vous pouvez avoir à remonter la conversation à un niveau supérieur.

© Dunod – Toute reproduction non autorisée est un délit.

11.2 LES STORIES SONT COMME DES PIERRES

Représentez-vous les stories comme des pierres. Si je devais prendre un très gros bloc de pierre pour le déposer sur le sol et le frapper avec une masse où il se casserait en une trentaine de morceaux, nous appellerions cela des pierres. Si vous prenez un de ces morceaux plus petits et le frappez à nouveau avec une masse, il va se briser en petits morceaux. Si nous voulons à présent faire preuve de créativité et donner des noms à ces diverses tailles, nous pourrions les désigner sous l'appellation de *bloc* ou de *cailloux*, mais je ne sais jamais quand quelque chose cesse d'être un rocher et commence à être simplement une bonne vieille pierre.

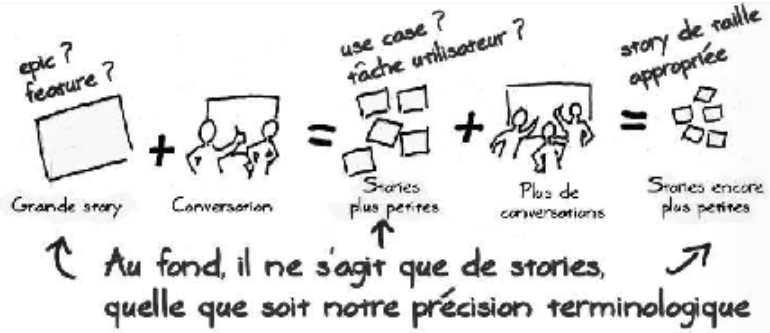


Tout comme les pierres, les grandes stories se décomposent en petites stories qui elles-mêmes peuvent se décomposer en stories encore plus petites. Et à chaque taille, aussi minime soit-elle, elles constituent toujours une story. Mais quel est le meilleur outil pour décomposer des stories ? C'est bien évidemment la conversation. Parfois, juste un peu de réflexion pourra convenir, mais si vous utilisez la conversation et la collaboration avec quelqu'un d'autre, alors vous augmentez la compréhension mutuelle.

Les conversations sont l'un des meilleurs outils pour décomposer les grosses stories.

À ce stade, les concepteurs de logiciels, dont je fais partie, sont mal à l'aise avec cette imprécision. Dans la plupart des organisations que j'ai côtoyées, on essaye de classer les stories selon leur taille. Et on retombe sur la querelle terminologique « des rochers et des cailloux ». La précision à propos de la taille a de l'importance, surtout quand vous êtes concerné, et ceci pourrait expliquer pourquoi les créateurs de logiciels sont préoccupés par la classification des stories.

Si vous définissez une terminologie pour votre organisation, n'essayez pas d'être trop précis. Le caractère mouvant de ce qui se trouve dans une story et de la taille qu'elle devrait faire est délibéré. Cela nous procure la flexibilité dont nous avons besoin pour utiliser cette simple idée tout au long du cycle de développement.



11.3 LES EPICS SONT DE GROS ROCHERS, PARFOIS UTILISÉS POUR FRAPPER LES GENS

Epic est un terme courant (je ne suis pas sûr de savoir qui en est à l'origine) utilisé pour décrire de grandes user stories, un peu comme le terme *bloc* est employé pour désigner un gros rocher. Maintenant, je vais être honnête avec vous et confesser qu'il m'a fallu des années pour me sentir à l'aise avec la dénomination *stories* pour désigner ce que nous développons, mais je comprends à présent pourquoi on les appelle ainsi. Le terme *epic* me pose encore des problèmes. Je me souviens de mon prof qui nous parlait de littérature épique et d'epics pour décrire des histoires de héros combattant le mal, comme Beowulf, Achille ou Frodon, en général avec une espèce d'arme magique ou l'assistance des dieux, mais je m'égare...

> Une epic est une story qu'on estime trop grande et qui devra être décomposée.

C'est bien d'avoir un terme pour désigner une grande story, mais méfiez-vous. Le terme *epic* est parfois utilisé comme une arme. J'ai souvent entendu un membre de l'équipe de développement dire à un commercial, un chef de produit, un utilisateur ou à quelqu'un qui demande quelque chose, que son histoire est une *epic*, pas une story. Cela est généralement dit d'un ton qui indique que le rédacteur de la story a fait quelque chose de mal, provoquant ainsi une réaction hostile de l'équipe de développement. Alors, s'il vous plaît, si vous faites partie de l'équipe, n'employez pas le terme *epic* comme un bâton pour réprimander quelqu'un. C'est un mauvais départ (et, éventuellement, une fin prématurée) de ce que devrait être une conversation productive.

Rappelez-vous que les epics sont de grandes stories qui peuvent avoir une taille appropriée du point de vue de l'entreprise, du client ou de l'utilisateur, mais pas pour l'équipe développement. Vous devez travailler ensemble pour les décomposer. Mais gardez l'epic à votre disposition, car vous aurez besoin d'en parler aux gens et de toutes les stories détaillées qui en découlent.

Si vous utilisez un outil numérique qui prend en charge le développement agile, il se sert probablement du concept d'epic pour désigner une grande story parente qui peut se décomposer en de nombreuses petites stories enfants.

11.4 LES THÈMES ORGANISENT DES GROUPES DE STORIES

Utilisez le terme *thème* pour décrire un ensemble de stories qu'il est utile de regrouper. Quand vous commencez à décomposer les grandes stories, vous finissez par avoir beaucoup de petites stories, et un thème est pour moi comme un sac dont je me sers pour collecter un tas de stories qui sont en rapport les unes avec les autres. Je peux utiliser un thème pour collecter une série de stories qui sont nécessaires pour une prochaine release, qui font partie de la même feature, qui relèvent d'un type particulier d'utilisateur, ou qui sont liées d'une quelque autre façon. Mais ma métaphore ne marche pas parfaitement car la même story peut se retrouver dans deux thèmes différents, alors que la même pierre ne peut être dans deux sacs différents.

Si vous utilisez un des outils disponibles qui facilitent l'organisation des groupes de stories agiles, il peut prendre en charge le concept de regroupement des stories dans un thème. Vous pouvez alors simplement faire référence au thème pour ce qu'il est réellement : votre prochaine release, la feature ou les stories relevant d'un type particulier d'utilisateur.

11.5 OUBLIEZ CES TERMES ET CONCENTREZ-VOUS SUR LE STORYTELLING

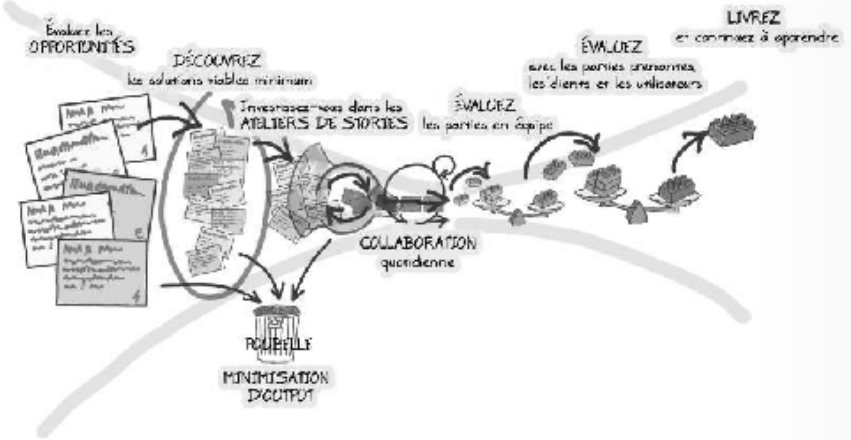
Les termes *epic* et *thème* ont trouvé leur place dans les outils de gestion du cycle de vie agile, dans certaines approches agiles spécifiques et dans le langage courant utilisé

© Dunod – Toute reproduction non autorisée est un délit.

pour discuter des stories. C'est pour cette raison que vous aurez besoin de les connaître et de les comprendre.

Mettons cependant ces termes de côté et oubliez même que je les ai mentionnés, tout au moins pendant un petit moment. Nous allons prendre du recul et examiner l'ensemble du cycle de vie de décomposition. Dans ce cycle, nous allons commencer par les grandes idées que vous pouvez vous représenter comme de gros rochers pour aboutir à de petits morceaux de logiciel fonctionnel. Puis nous réassemblerons ces petits morceaux de logiciel en features, produits et releases désirés par les clients et les utilisateurs.

De loin, voici à qui ressemble ce cycle :



Rentrons à présent dans les détails.

11.6 ON COMMENCE PAR LES OPPORTUNITÉS

Le voyage d'une story commence sous la forme d'une idée. Il peut s'agir d'une idée d'une nouvelle feature ou d'un tout nouveau produit. Cela peut être aussi une idée d'amélioration d'une feature existante ou un problème que l'on doit résoudre. J'utiliserai le terme *opportunité*, parce qu'il s'agit d'une opportunité de faire quelque chose dont nous allons tirer parti. Je vous suggère de créer une liste de ces opportunités que j'appelle *backlog d'opportunités*.

Notre première véritable conversation de story est une discussion de haut niveau sur les « qui », les « quoi » et les « pourquoi » dont le principal objectif est de prendre une décision go/no go. La décision go ne signifie pas que nous allons faire un développement, mais que nous allons prolonger et approfondir les discussions pour vraiment comprendre la story. Mais je ne veux pas aller de l'avant et dépenser beaucoup de temps à faire cela si nous pouvons détecter que c'est une mauvaise idée dès le départ. La décision no go est une manière polie de dire « poubelle » et on peut donc traduire la décision go/no go en décision « aller de l'avant » ou « poubelle ». N'oubliez pas qu'il y a toujours trop de choses à développer et que le fait de tuer une opportunité médiocre avant qu'elle ne fasse perdre trop de temps à tout le monde devrait être célébré.

> Utilisez les discussions sur les opportunités pour vous mettre d'accord sur l'intérêt du problème à résoudre, ce qui permettra de prendre une décision aller de l'avant/poubelle.

11.7 DÉCOUVERTE D'UNE SOLUTION VIABLE MINIMUM

Maintenant que vous avez choisi d'aller de l'avant, il est temps d'approfondir les choses. Utilisez la découverte pour trouver une solution qui mérite d'être développée, mais n'oubliez pas de minimiser vraiment cette solution en cherchant à la rendre aussi petite et précieuse que possible.

© Dunod – Toute reproduction non autorisée est un délit.

Au cours du processus de découverte, vous allez répondre aux questions suivantes :

- À votre avis, quels clients et utilisateurs vont utiliser votre solution ?
- Comment répondent-ils à leurs besoins aujourd'hui sans votre solution ?
- Comment votre solution impactera leur vie quotidienne ?
- À quoi votre solution va-t-elle ressembler et comment va-t-elle réagir ?
- Combien faudrait-il de temps pour développer votre solution ?

Il y a un grand nombre de pratiques qui peuvent se révéler utiles au cours du processus de découverte, notamment le story mapping. Cette technique peut vous aider à comprendre comment les gens fonctionnent à l'heure actuelle, puis faciliter l'organisation de vos idées sur la façon dont les choses vont changer pour les utilisateurs après le développement de votre solution.

La découverte est le moment essentiel pour examiner nos hypothèses et travailler à leur validation. Cela peut prendre la forme d'une analyse plus approfondie pour comprendre les règles métiers ou la réglementation externe. Vous devrez alors passer du temps avec les clients et les utilisateurs pour comprendre comment ils travaillent. Cela implique de développer des prototypes de votre solution et de les valider avec votre public cible. Et cela inclut également le développement de prototypes techniques pour évacuer les risques techniques.

> Ayez des conversations pour découvrir et explorer une solution viable de petite taille.

Célébrez chaque partie de la solution que vous pouvez mettre à la poubelle en toute sécurité, ou déposez-la dans le backlog d'opportunités pour qu'elle soit traitée ailleurs. Nos opportunités ont pu constituer de gros rochers, mais à l'intérieur de ces roches se trouvaient des diamants et des métaux précieux. Décomposez ces gros blocs pour en extraire les parties vraiment précieuses, et célébrez la mise au rebut de tous les déchets.

Tout au long de la découverte, vous pouvez choisir de développer de petites choses qui faciliteront votre apprentissage, notamment l'interface utilisateur ou les prototypes d'architecture.

Spike est un terme utilisé pour les parties de développement ou de recherche que nous effectuons avec un objectif explicite d'apprentissage. C'est un terme qui vient de la communauté Extreme Programming pour décrire le travail qui n'a pas un rapport direct avec le logiciel à livrer. Utilisez des spikes pour favoriser l'apprentissage de votre équipe quand elle développe quelque chose.

Une fois que vous êtes convaincu de posséder ce petit sous-ensemble de stories que vous allez développer et livrer à vos clients et vos utilisateurs, il est alors temps de se pencher sur la livraison. J'appelle backlog de release ce sous-ensemble de stories qui aboutit à une release de produit.

11.8 APPROFONDISSEZ LES DÉTAILS DE CHAQUE STORY AU COURS DE LA LIVRAISON

Nos opportunités peuvent avoir commencé sous la forme de gros rochers que les conversations de découverte ont décomposés pour en extraire les métaux précieux. Mais la livraison sera plus rapide et plus efficace si l'on continue à décomposer ces morceaux en parties les plus petites possibles, sans oublier que chaque partie doit toujours être quelque chose que l'on peut développer et dont on peut tirer des enseignements. Il va alors falloir beaucoup d'autres conversations pour approfondir tous ces détails.

J'ai imaginé une machine révolutionnaire qui décompose les rochers : d'un côté je charge ces gros rochers qui contiennent tous ces métaux précieux, et de l'autre côté je récupère de petites pierres d'une taille appropriée prêtes à entrer dans le prochain cycle de développement. J'appelle cette machine la *machine à atelier de stories*, et son nom décrit exactement ce que nous allons faire.

Nous allons avoir des discussions approfondies avec les développeurs et les testeurs et tout le reste de l'équipe qui va développer le logiciel, pour vraiment nous plonger dans les détails. Ce sont nos « dernières meilleures conversations », c'est-à-dire les conversations pendant lesquelles nous avons vraiment besoin de nous mettre d'accord sur les *confirmations*, ou les critères d'acceptation des petites parties du logiciel que nous développons, parce que l'étape suivante c'est justement le développement. Puisque nous savons que ce sont ces conversations qui décomposent les logiciels, nous allons les utiliser pour affiner nos stories à la bonne taille et les placer dans le prochain sprint ou la prochaine itération.

© Dunod – Toute reproduction non autorisée est un délit.

Animez des ateliers de stories pour approfondir les choses et discuter des détails, décomposer les stories et vous mettre vraiment d'accord sur ce que vous allez précisément développer.

J'aime bien appeler ces dernières meilleures conversations de stories des *ateliers de stories* car tout le monde sait que les réunions sont improductives alors que les ateliers sont conçus pour que le travail se fasse. Selon les besoins, ils peuvent pratiquement avoir lieu tous les jours. Parfois, ils ont lieu tous à la fois au cours d'une séance de planification. Dans Scrum, ils peuvent se tenir au cours de ce qu'on appelle une session d'*affinage de backlog*. Quel que soit le nom que vous donnez à ces discussions, l'essentiel est qu'elles aient lieu.

11.9 CONTINUEZ À DISCUTER PENDANT QUE VOUS DÉVELOPPEZ

La machine à atelier de stories alimente la suivante, la machine de livraison agile. Voici comment je l'imagine : d'un côté, nous plaçons ces petites stories de taille normale et de l'autre côté il ressort des bouts de logiciel fonctionnel et impeccable, ou quoi que ce soit dont votre story décrit la réalisation.

Quelle que soit l'ardeur que vous avez déployée, vos dernières meilleures conversations de stories ne vous permettront pas de prédire tout ce que vous allez apprendre une fois que vous aurez commencé à développer. Prévoyez d'avoir de nombreuses conversations de stories tous les jours et évoquez leur nécessité dans vos réunions quotidiennes.

Si vous êtes un développeur et que les détails apportés par les discussions de stories ne sont pas suffisants pour répondre à vos questions, vous devez tout de suite trouver quelqu'un à qui parler pour poursuivre votre discussion. Vous ne pouvez pas accuser les mauvaises spécifications ici. N'oubliez pas qu'avant de commencer, vous avez travaillé avec les autres pour identifier tout ce que vous devez savoir pour réaliser le développement. Mais nous sommes tous humains et c'est donc normal de rater certaines choses.

Si vous êtes PO, UX designer, analyste métier ou une de ces autres personnes qui participent au processus de décision de ce qu'il faut développer, n'ayez pas peur de vous lever de votre bureau pour aller voir comment se passe le développement. Je vous promets que dès que vous verrez quelque chose qui fonctionne, vous verrez quelque chose d'utile. Et il est plus que probable que la personne qui développe appréciera un minimum de feedback.

Continuez les conversations au fur et à mesure que vous développez pour fournir des réponses et émettre des commentaires sur ce qui est développé.

11.10 ÉVALUEZ CHAQUE PARTIE

Lorsque ces morceaux de logiciel fonctionnel sortent de la machine de livraison agile, c'est le moment pour les gens qui ont contribué à décrire ce qu'il faut développer et pour les personnes qui ont assuré le développement de faire une pause et d'examiner attentivement ce qui a été réalisé.

N'oubliez pas qu'il ne s'agit pas vraiment de machines et que les personnes avec qui vous travaillez ne sont pas les rouages d'une machine. De plus, tous ces morceaux de logiciel sont différents.

Arrêtez-vous et étudiez vraiment la qualité de la solution développée. Réfléchissez à la manière dont vous l'avez effectivement planifiée. Avez-vous vraiment terminé ce qui était prévu ? Est-ce qu'il a fallu beaucoup plus de temps ? Ou bien moins de temps ? Est-ce bien ce que vous aviez imaginé ? Parlez véritablement du bon fonctionnement

© Dunod – Toute reproduction non autorisée est un délit.

de la « machine ». C'est le moment de faire des ajustements ou des changements dans votre façon de travailler pour obtenir une meilleure qualité de manière plus prévisible.

> Réfléchissez souvent à la qualité du produit, à vos projets et à votre façon de travailler.

En Scrum, cette première phase d'évaluation s'appelle *revue et rétrospective de sprint*. Quels que soient les noms que vous donnez à ces phases d'arrêt, pour examen et réflexion, assurez-vous qu'elles aient lieu.

11.11 ÉVALUEZ AVEC LES UTILISATEURS ET LES CLIENTS

N'oubliez pas que ce que vous développez n'est en général pas pour vous. Vous allez devoir faire face aux utilisateurs et aux clients pour voir comment ils appréhendent le produit. Certaines de ces personnes n'ont vu qu'un prototype, qu'une armature ou qu'une description textuelle. Le fait de voir et de toucher quelque chose qui fonctionne vraiment leur permet d'évaluer si oui ou non ils ont le bon produit.

Mais chacun de ces petits morceaux sortis de la machine de livraison agile peut ne pas être suffisant pour forger leur opinion. Dans mon modèle mental, j'imagine que chacun de ces morceaux s'empile sur une balance. Il s'agit d'une de ces balances anciennes avec un contrepoids sur l'autre plateau. Sur mon contrepoids figure le mot *assez*, c'est-à-dire suffisamment pour tester avec les utilisateurs et les clients, et pour apprendre quelque chose.

Habituellement *assez* signifie un écran complet, ou un flux d'écrans, qui permet aux utilisateurs d'effectuer une tâche ou bien d'atteindre un objectif significatif. Et je ne veux pas me contenter d'une simple démonstration devant les utilisateurs. Je ne cherche pas non plus à ce qu'ils me disent leur satisfaction. Je cherche à apprendre, ce qui généralement prend la forme de formules du genre « *Ce n'est pas tout à fait ce qui convient* » et « *maintenant que j'utilise vraiment le produit, ce serait mieux si…* »

> Apprenez en testant des morceaux significatifs de logiciel fonctionnel avec les clients et les utilisateurs.

11.12 ÉVALUEZ AVEC LES PARTIES PRENANTES DE L'ENTREPRISE

Il y a probablement d'autres personnes dans votre organisation qui ont des intérêts particuliers dans le logiciel que vous développez. Il ne s'agit peut-être pas des gens qui vont l'utiliser tous les jours, mais ils sont concernés par le fait que le logiciel puisse être livré à ceux qui vont s'en servir dès que possible.

Faites une revue du produit pour leur montrer son état d'avancement et pour le replacer dans son contexte par rapport au plan d'ensemble. N'oubliez pas qu'ils ne sont probablement pas intéressés par les détails de l'agencement des petites parties du logiciel. Ce qui les intéresse, c'est la progression vers la solution viable minimum, parce que c'est la plus petite partie que nous pouvons livrer et dont on peut vraiment tirer parti dans le monde extérieur. C'est donc de cela qu'il faut leur parler. Partagez

© Dunod – Toute reproduction non autorisée est un délit.

avec eux les résultats des tests que vous avez effectués avec les utilisateurs ou les clients car cela les intéresse aussi.

Communiquez sur les progrès et la qualité auprès des parties prenantes à l'intérieur de votre organisation.

11.13 SORTEZ UNE RELEASE ET CONTINUEZ À ÉVALUER

J'imagine une dernière balance à la fin de tout cela. Sur cette balance j'ai empilé les parties que nous avons examinées ensemble, testées avec les clients et les utilisateurs et communiquées aux parties prenantes de ma société. Cette balance est très semblable à celle que nous avons utilisée auparavant pour évaluer le produit avec les utilisateurs et les clients. Comme avec l'autre balance figure sur le contrepoids le mot *assez*, mais cette fois-ci, cela signifie qu'il y a de parties pour sortir une release à l'attention des clients et des utilisateurs afin de produire l'impact recherché. Lorsque la balance est à l'équilibre, c'est le moment de déployer la release pour les clients et les utilisateurs.

Mais ne vous arrêtez pas là car vous avez encore quelque chose à apprendre. Si vous êtes comme Eric au chapitre 3, l'apprentissage est votre objectif principal, si bien que vous avez besoin d'utiliser des indicateurs pour savoir comment les gens utilisent votre produit. Vous aurez besoin d'avoir des conversations en tête à tête pour apprendre pourquoi ils utilisent ou n'utilisent pas votre produit. Ne vous contentez pas de vos prévisions, mais utilisez des indicateurs et des conversations pour savoir ce qui se passe véritablement.

© Dunod – Toute reproduction non autorisée est un délit.

Obtenez des indicateurs de vos entrevues en tête à tête avec les utilisateurs pour savoir si votre impact cible a été atteint.

S'il s'agissait d'un projet, vous l'auriez alors terminé puisqu'il a été livré. Mais vous venez de réaliser quelque chose. C'est un produit et la vie d'un produit *commence* à partir du moment où il est livré. Lorsque vous commencez à faire attention à ce que les gens font avec votre produit, je vous promets que vous trouverez des opportunités pour l'améliorer. Notez-les et injectez-les au début de ce modèle.

C'est le cycle *réel* de la vie, tout du moins de la vie d'une story.

Et cela fait beaucoup de blocs de pierre à casser. Mais, au fait, qui est censé casser toutes ces pierres ? Je suis heureux que vous ayez posé la question car c'est justement l'objet du chapitre suivant.

12

Les casseurs de blocs

Croire qu'il n'y a qu'une seule personne responsable de l'écriture des stories et de la conduite des conversations est une grave erreur que l'on rencontre souvent dans les pratiques agiles (dans Scrum, cette personne s'appelle le Product Owner). Il y a deux raisons principales qui expliquent pourquoi ce modèle ne fonctionne pas, et probablement toute une série d'autres raisons moins importantes.

- **Première raison** – Il y a trop de conversations à mener pour transformer une story (on part d'une vague idée pour arriver à des petites choses spécifiques à développer). Une seule personne ne suffit pas pour gérer toutes ces conversations. Et si vous définissez votre processus de telle sorte qu'une personne en particulier doive être présente dans toutes les conversations, vous constaterez rapidement que cette personne va constituer un goulet d'étranglement.
- **Deuxième raison** – Une personne seule ne peut réunir l'expertise requise et les différents points de vue nécessaires pour arriver à la meilleure solution. Il faut la collaboration de plusieurs personnes avec des compétences différentes pour véritablement parvenir aux meilleures solutions.

> Le PO,à lui tout seul, ne peut pas écrire toutes les stories et être présent à toutes les conversations.

Ne mésinterprétez pas mon propos : selon moi, dans un bon développement de produit, le PO est un meneur crucial. C'est lui qui maintient le produit et toute l'équipe dans la même direction.

Il existe une alternative qui est la conception en comité, un très bel anti-pattern où tout le monde a une voix égale pour décider ce qu'il faut faire. Dans un comité, quand on a du temps et des ressources pour réaliser une seule chose, on fait des compromis. Mon ex-épouse et moi-même faisions souvent cela quand il fallait choisir

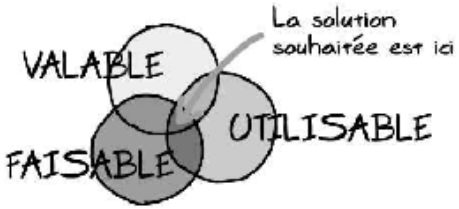
© Dunod – Toute reproduction non autorisée est un délit.

un restaurant. Elle voulait des fruits de mer, je voulais manger mexicain, et nous nous mettions d'accord sur une troisième solution. Lorsqu'un comité n'est limité ni par le temps ni par les ressources, il n'y a pas besoin de compromis et toutes les fonctionnalités sont développées. Vous avez déjà utilisé ce genre de logiciels : un produit avec d'innombrables fonctionnalités où votre plus gros problème est de trouver la fonctionnalité que vous souhaitez ou de vous rappeler comment l'utiliser.

Les PO efficaces s'entourent des personnes dont ils ont besoin pour prendre de bonnes décisions. Ils intègrent l'expertise et les opinions d'un grand nombre de personnes. Mais, au bout du compte, lorsque les ressources sont limitées ou que le succès du produit est en jeu, ils doivent prendre des décisions. Et il y aura toujours quelqu'un qui sera mécontent de cette décision. Mon ami Leisa Reichelt résume bien ceci : « *La conception en communauté n'est pas la conception en comité… la conception n'est jamais démocratique…* »[1].

12.1 VALABLE, UTILISABLE ET FAISABLE

Dans son livre, *Inspired: How to Create Products Customers Love* (SVPG Press), Marty Camara décrit la responsabilité d'un chef de produit quand il doit identifier qu'un produit est valable, utilisable et faisable. Quand j'ai lu ces mots, j'ai pensé à un simple diagramme de Venn où la solution désirée serait l'intersection de ce qui est valable pour notre entreprise et nos clients, utilisable par les utilisateurs et faisable étant donné le temps et les outils dont nous disposons.



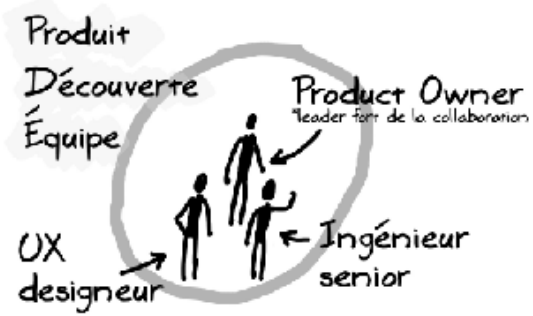
Mais ce qui n'est peut-être pas si évident que cela, c'est que pour vraiment identifier la solution au centre du diagramme, il faut une collaboration entre des personnes qui comprennent notre métier, nos clients, nos utilisateurs et la technologie que nous utilisons, et ne comprennent pas seulement ces choses, mais prennent aussi la responsabilité de la réussite de la solution. Ces personnes parlent en fait avec les parties prenantes, les clients et les utilisateurs ; ce sont eux qui en réalité conçoivent et testent les interfaces utilisateur ainsi que le code qui fait fonctionner le produit.

Vous souvenez-vous de cette idée fausse dans la communauté du développement agile où un seul PO ou chef de produit décide de ce qu'il faut développer ? Il est rare, voire impossible, qu'une seule personne possède les compétences métiers et les connaissances nécessaires en conception d'interface utilisateur et en ingénierie pour déterminer cette intersection entre le valable, l'utilisable et le faisable. C'est la raison pour laquelle la plupart des organisations efficaces utilisent de petites équipes de découverte transversales qui travaillent ensemble pour trouver la bonne solution. Comme nous l'avons évoqué au chapitre précédent, il faut vous représenter la découverte comme un travail de décomposition. C'est le travail que nous faisons quand nous transformons une story, qui n'est à la base qu'une vague idée, en quelque chose de petit et spécifique que l'on peut développer.

> Une petite équipe pluridisciplinaire dirigée par un PO orchestre le travail de découverte du produit.

La taille idéale d'une telle équipe est de deux à quatre personnes (c'est le bon nombre de personnes pour un dîner ou une conversation afin qu'ils puissent forger rapidement une compréhension mutuelle).

Cette équipe doit être dirigée par un PO ou un chef de produit qui possède une connaissance approfondie de la vision et de la stratégie de l'entreprise, ainsi que du marché couvert par son produit. Dans cette équipe de base, on trouve une personne qui comprend les utilisateurs, qui est à l'aise pour travailler avec eux afin d'en savoir plus sur leur façon de travailler, et qui peut dessiner et créer de simples prototypes d'interface utilisateur. Cette équipe inclut également un ingénieur senior de l'équipe qui va développer le produit. Celui-ci doit comprendre l'architecture actuelle du système et avoir une bonne perception des nouvelles approches d'ingénierie qui pourraient servir à résoudre des problèmes difficiles. Le véritable secret ici est que les solutions les plus innovantes viennent souvent de cet ingénieur qui perçoit à la fois le problème de l'entreprise et les problèmes des utilisateurs.



Une équipe de découverte soudée est un groupe puissant et réactif d'experts qui peuvent trouver des problèmes et valider des solutions rapidement. J'ai souvent

1. Leisa Reichelt a pour la première fois prononcé ce bon mot lors d'une conférence IXDA en 2009. Son point de vue est développé dans cet article : http://www.disambiguity.com/designbycommunity/.

© Dunod – Toute reproduction non autorisée est un délit.

entendu le terme *triade* pour décrire cette équipe de base. Lors de ma dernière visite récente chez Atlassian, qui est basée à Sydney, Sherif (dont nous avons fait la connaissance au chapitre 8) m'a montré trois sièges rapprochés. Il m'a expliqué que c'était l'emplacement où la triade s'asseyait. La zone alentour était remplie de bureaux et d'ordinateurs où le reste de l'équipe s'asseyait. Le terme *triade* est aussi utilisé lorsque l'équipe de découverte est composée de deux personnes, quatre personnes, voire plus, car on parle ici des trois préoccupations (valable, utilisable et faisable) et non pas des corps de trois personnes.

> Renforcez les PO avec une équipe de découverte qui inclut des experts en expérience utilisateur, en conception et en technique.

12.2 UNE ÉQUIPE DE DÉCOUVERTE A BESOIN DE BEAUCOUP D'AUTRES PERSONNES POUR RÉUSSIR

Une équipe de découverte efficace coordonne la collaboration non seulement avec l'équipe de développement, mais aussi avec les parties prenantes de l'entreprise, les experts du domaine, les clients et les utilisateurs finaux. C'est un travail difficile qui nécessite des compétences élevées en communication et en facilitation, en plus de l'expertise particulière que chaque membre de l'équipe apporte.

Maintenant, voici le véritable secret. Quelle que soit la taille du produit, il faut une équipe pour le développer. Pour que la vision du produit soit claire, un bon chef de produit est essentiel car il faut s'assurer que la solution développée par l'équipe soit cohérente et aide chacun à aller dans la même direction. Les meilleurs leaders font en sorte que tout le monde s'approprie le produit. Dans un environnement sain piloté par les stories, vous verrez en permanence beaucoup de conversations de stories. Et beaucoup d'entre elles se dérouleront sans que le chef de produit soit là.

12.3 LES TROIS AMIGOS

¡Three amigos! est le nom d'un médiocre western-comédie de 1986 avec Steve Martin, Chevy Chase et Martin Short. Quel est le rapport entre ce film et les stories ou le développement logiciel agile ? Il s'agit d'un trio qui se complète efficacement le temps d'une histoire.

Vous vous souvenez peut-être que *l'atelier de stories* est le terme que je donne à la dernière bonne conversation pendant laquelle nous décidons plus précisément ce qu'il faut développer. C'est à ce moment-là que les trois amigos entrent en scène.

Au cours de cette dernière bonne conversation, il faut vraiment examiner beaucoup de détails et de solutions de rechange pour la mise en œuvre, et nous avons donc besoin d'un développeur de l'équipe qui va créer le logiciel, idéalement un des développeurs qui va réellement travailler sur le projet.

Pour que ce petit morceau de logiciel puisse être considéré comme terminé, il devra être testé, si bien que nous aurons besoin d'un testeur dans la conversation. Un testeur, le *premier* amigo, apportera souvent un œil critique dans la discussion, en repérant les choses qui pourraient tourner mal plus tôt que prévu. Le testeur est souvent le meilleur au jeu du « que se passe-t-il si... ? ».

Et bien sûr nous aurons besoin de quelqu'un qui comprend ce que nous développons (pour qui et pourquoi) si bien qu'il nous faudra un membre de l'équipe de découverte du produit. Cette personne est le *deuxième* amigo.

À ce stade, nous n'introduisons pas souvent une nouvelle idée de feature car cela a probablement déjà été fait lors de la découverte. Maintenant nous sommes plus ou moins engagés à développer quelque chose, et il est donc important de comprendre précisément à quoi ce logiciel doit ressembler et comment il doit se comporter. La personne impliquée dans cette conversation est donc souvent un concepteur d'expérience utilisateur ou un analyste métier qui a travaillé sur ces détails. C'est notre *troisième* amigo.

Ce groupe travaillera sur les détails et se mettra d'accord sur les critères d'acceptation spécifiques de la story. Ce n'est pas le but de cette conversation de déterminer la meilleure estimation du temps qu'il faudra pour développer et tester le logiciel. Et c'est souvent dans cette conversation que nous allons prendre la décision de scinder la story en stories de développement plus petites (ces stories qui prennent de un à trois jours à développer et à tester).

© Dunod – Toute reproduction non autorisée est un délit.

Des conversations de stories ont lieu en permanence quand nous traduisons les idées en développement de logiciels. Dans toutes les discussions, orientez le sujet de la conversation sur ce qui est valable, utilisable et faisable. Incluez dans les conversations des personnes qui sont capables de parler de ces choses-là. Évitez la conception en comité en conservant un PO responsable du succès de son produit.

L'anti-pattern client-fournisseur

Il y a un anti-pattern remarquable qui entrave également l'utilisation des stories. En fait, il empêche les gens qui travaillent ensemble de faire correctement *quoi que ce soit*. C'est le redoutable anti-pattern client-fournisseur.

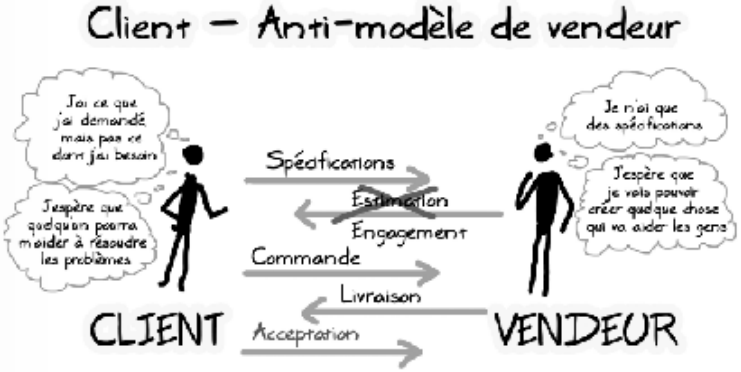
Dans cet anti-pattern, une personne prend le rôle de client au cours d'une conversation, tandis qu'une autre prend le rôle du fournisseur. C'est la tâche du client de savoir ce qu'il veut et d'expliquer les détails au fournisseur. C'est ce que nous appelons les « spécifications ». C'est la tâche du fournisseur d'écouter, de comprendre et ensuite de réfléchir à une approche technique pour livrer ce que le client a demandé. Le fournisseur donne ensuite son estimation, ce qui dans le jargon informatique vaut effectivement « engagement », raison pour laquelle les développeurs craignent souvent de fournir une estimation sans une étude approfondie.

Le reste de l'histoire est tristement prévisible.

De temps à autre, l'estimation est terriblement précise, et le client obtient ce qu'il voulait, en l'occurrence ce dont il avait besoin.

Mais, la plupart du temps, l'élaboration d'une solution dure plus longtemps que prévu par le fournisseur. La personne ayant le rôle de fournisseur peut avancer toutes sortes

© Dunod – Toute reproduction non autorisée est un délit.



d'excuses pour le retard, notamment le manque de détails des spécifications qui lui ont été fournies, ou tout simplement de « mauvaises spécifications ». Le client peut dénoncer une estimation inexacte, personne ne semblant remarquer qu'il s'agit là d'un paradoxe. Lorsque la solution est livrée, la personne ayant le rôle de client reçoit ce qu'il a demandé, et se met ensuite à utiliser le logiciel pour se rendre compte que ce n'est pas ce dont il a besoin. Il n'obtient pas l'impact espéré.

La vraie tragédie ici est que la personne dans le rôle du client comprend son problème bien mieux qu'elle n'est capable de prédire ce qui permettra de le résoudre. Et la personne qui comprend la technologie est souvent la plus qualifiée pour résoudre le problème parce qu'elle sait combien la technologie avec laquelle elle travaille peut être utile. Qui plus est, la plupart des experts en technologie sont sincèrement désireux d'aider. Ils veulent s'assurer que les choses qu'ils développent seront utiles.

Mais, dans l'anti-pattern client-fournisseur, les conversations sur les problèmes et les solutions sont remplacées par des discussions et des accords sur les spécifications. Personne ne gagne.

Un des objectifs des stories est de briser cet anti-pattern.

Beaucoup d'entre nous ont avec leur médecin un type de relation qui casse cet anti-pattern. La prochaine fois que vous irez chez votre médecin, essayez de lui fournir vos « spécifications ». Dictez-lui les ordonnances que vous souhaitez et planifiez les analyses que vous désirez. S'il est gentil, il va sourire et vous dira « C'est intéressant ; dites-moi où vous avez mal. »

Dans ma tête, j'imagine un continuum où d'un côté se trouve le mot *serveur*, et de l'autre côté le mot *médecin*. Essayez de faire en sorte que vos relations de travail soient plus du genre d'une bonne relation médecin-patient et beaucoup moins du genre serveur-client.

12.4 LE PO VU COMME UN PRODUCTEUR

Si vous travaillez dans un environnement informatique plus traditionnel, la notion de PO peut sembler déroutante. Par exemple, si vous aidez à développer des systèmes

critiques pour une banque, la banque sait que ses véritables produits sont les services bancaires qu'elle vend à ses clients. Le rôle d'un « chef de produit » dans une banque est de s'occuper d'un type spécifique de produit de crédit ou de compte bancaire. Les systèmes informatiques qui prennent en charge cette offre de service ne sont qu'une pièce du puzzle. Et souvent cette même infrastructure informatique prend en charge beaucoup de produits bancaires différents. Naturellement, la banque ne voit pas cette infrastructure comme un produit, et il n'y a souvent personne qui en soit propriétaire.

Dans ces types d'organisations, les analystes métiers sont souvent placés dans un rôle de « collecte des spécifications ». Ils vont servir d'intermédiaire entre les développeurs et les parties prenantes de l'entreprise, comme le chef de produit d'un produit bancaire ou d'un produit d'assurance. Lorsque ces entrepreneurs ont besoin de modifier l'infrastructure informatique qui prend en charge leur produit, ils travaillent avec l'analyste métier pour décrire ces changements. Et c'est le moment où ils peuvent prendre le rôle du client alors que l'analyste prend le rôle du fournisseur, ce qui permet à l'anti-pattern d'entrer en jeu.

Au détour d'une conversation, mon ami David Hussman m'a proposé une meilleure métaphore pour décrire la relation que l'analyste métier devrait avoir avec ses parties prenantes : c'est celle qu'un producteur de musique devrait avoir avec un groupe. Venant de David, c'est logique car il est à la fois gourou agile et ex-guitariste d'un groupe de heavy metal. Il a travaillé avec des producteurs et a été producteur lui-même. Dans cette relation, le groupe aborde l'industrie de la musique avec passion, et si possible avec talent, mais il ne connaît pas le business de la musique ou le processus d'enregistrement d'un album. C'est au producteur de s'occuper de cela et son rôle est d'aider le groupe à réaliser le meilleur enregistrement possible. Les producteurs à succès peuvent devenir à leur tour des artistes raffinés qui vendent beaucoup de disques.

En tant qu'analyste métier dans un contexte informatique, c'est votre travail. Prenez la vision de vos parties prenantes et aidez-les à réussir. Vous ne pouvez pas vous contenter d'être simplement un preneur d'ordre, et vous aurez besoin de vous comporter davantage comme un médecin. Cela signifie parfois que vous allez dire à vos parties prenantes des choses qu'ils ne veulent pas entendre. Mais, si vous êtes sincère tout en les aidant à réussir, ils vont le voir et valoriseront votre assistance.

> Lorsque vous agissez en tant que PO pour les idées d'autres parties prenantes, prenez le rôle d'un producteur qui les aide à réussir.

Il y a un anti-pattern potentiel qui consiste à faire prendre aux parties prenantes le rôle de PO. Je dis *potentiel* car cela peut fonctionner si l'entrepreneur a beaucoup d'aide et de soutien de la part des autres membres de l'équipe, le désir d'apprendre à faire le travail d'un PO, et du temps pour le faire... La tâche de PO n'est pas une responsabilité anodine et ne devrait pas être traitée comme quelque chose que l'on peut faire pendant son temps libre. Au lieu d'imposer une autre tâche à un entrepreneur, je vous recommande de trouver un producteur pour l'aider à réussir.

12.5 C'EST COMPLIQUÉ

Pour une idée qui à la base est simple, toute cette histoire de story est devenue bien confuse. Et si quelqu'un vous dit que le développement logiciel, ou d'ailleurs *tout* développement de produits, est facile, c'est un menteur.

Les stories sont beaucoup de choses à la fois. Nous utiliserons le mot pour désigner la fiche, le bout de logiciel que l'on développe et surtout le genre de conversations que nous avons pour prendre des décisions sur ce que nous devons développer. Les stories peuvent décrire de très grandes opportunités ou des parties presque insignifiantes qui, en elles-mêmes, ne sont pas nécessairement significatives pour les clients et les utilisateurs. Travailler avec des stories est un processus continu de conversations et de discussions pour décomposer de grandes choses en petites choses. Et à travers toutes ces conversations, on se concentre non seulement sur ce que nous pouvons développer, mais aussi sur le « pour qui » et le « pourquoi ». Le story mapping n'est qu'une des façons de nous aider à décomposer les grandes choses tout en focalisant le sujet de la conversation sur les personnes qui utilisent votre produit.

Si tout cela commence à faire sens, alors c'est que votre esprit a entamé cette nécessaire évolution de mentalité. Ce n'est pas une révolution d'utiliser des stories pour documenter des spécifications, mais un virage vers une collaboration plus efficace, pour se consacrer ensemble à la résolution des problèmes réels qu'ont les utilisateurs avec les produits que vous développez.

Et j'espère que vous serez d'accord pour dire que c'est une très belle chose.

13

On commence par les opportunités

Permettez-moi de souligner encore une fois l'analogie entre les stories et les blocs de pierre qu'il faut casser en morceaux plus petits. Au départ, il y a toujours un premier bloc à étudier attentivement pour décider si ça vaut le coup de le décomposer. Ce « bloc zéro » constitue ce que l'on appelle, dans nos flux de stories, une opportunité.

J'utilise le terme *opportunité* pour désigner une idée qui à notre avis permettra de résoudre un problème. Je ne suis pas juste un type optimiste qui ne voit que le verre à moitié plein, mais je considère simplement que c'est une mauvaise idée d'envisager que toute idée doit être incluse dans notre produit, car vous savez bien qu'il n'y a ni assez de temps ni assez de compétences disponibles pour développer tout ça. Et, même si vous aviez toutes les ressources nécessaires, vos clients seraient dépassés par toutes ces fonctionnalités.

13.1 TENEZ DES CONVERSATIONS SUR LES OPPORTUNITÉS

Quand on se confronte aux idées, elles sont souvent assez vastes, mais ce n'est pas toujours le cas. Dans le jargon des stories, on pourrait les désigner sous le terme *epics*, mais je préfère les appeler des *opportunités*. Peu importe comment vous les appelez, il s'agit toujours de stories et le but des premières conversations est de décider s'il faut les développer ou bien les abandonner. Pour chacune de ces opportunités, nous pouvons discuter des questions suivantes :

© Dunod – Toute reproduction non autorisée est un délit.

- **Pour qui sont-elles ?**
  À ce niveau, il s'agit souvent de différents groupes d'utilisateurs, de clients ou du marché cible.
- **Quels problèmes allons-nous résoudre ?**
  Pour chaque type d'utilisateur, nous discuterons des problèmes que nous allons résoudre. Nous aurons besoin d'évoquer la manière dont ces utilisateurs résolvent leurs problèmes jusqu'à présent à l'aide d'outils manuels, des produits de nos concurrents ou, pire encore, de notre produit, qui leur cause des soucis.
- **Ce que nous allons imaginer**
  Nous pouvons avoir quelques idées sur ce que le produit ou la feature devraient être et nous devons discuter de ces idées.
- **Pourquoi ?**
  C'est le bon moment pour discuter des bénéfices que pourrait retirer notre organisation en développant des logiciels pour ces utilisateurs. En général, la résolution des problèmes des utilisateurs ne suffit pas, et il nous faut aussi prendre en compte le retour sur investissement et savoir s'il est conforme à notre stratégie commerciale actuelle. Je ne dis pas que nous devons calculer le retour sur investissement, parce que quiconque en serait capable à ce stade aurait probablement plein d'autres choses en tête que des stories. Il faut simplement discuter en termes généraux des bénéfices que notre organisation pourrait escompter si nous prenions la décision d'effectuer ce développement.
- **Quelle taille ?**
  À ce niveau, même quand le champ couvert est très vaste, nous pouvons instinctivement commencer à donner une estimation du temps de développement, même si elle n'est pas très précise. Il est préférable d'examiner ce qu'il y a à faire et de le comparer à quelque chose que vous avez déjà fait : « *Cela ressemble à cette autre feature que nous avons placée dans la dernière release. Elle nous a pris deux semaines, et ça pourrait donc prendre environ le même temps.* » Afin de décider si nous devons entamer une discussion sur cette idée, il est intéressant de savoir si nous parlons de quelque chose qui va prendre des jours, des semaines ou des mois à développer.

La pile entière de ces stories est ce que j'appelle un *backlog d'opportunités*. Nous ignorons encore à ce stade si nous devons tout développer. Il faut identifier les opportunités en phase avec la stratégie commerciale de votre entreprise et censées résoudre les problèmes des clients et des utilisateurs. Il faudra ensuite avoir suffisamment de discussions pour pouvoir prendre une décision go/no go.

13.2 APPROFONDISSEZ, JETEZ OU RÉFLÉCHISSEZ

La décision « go » ne veut pas dire que l'on va développer cette chose, mais cela signifie que l'on va approfondir la discussion de découverte de l'opportunité. Au cours de la découverte, il y aura beaucoup plus de conversations, probablement avec d'autres personnes qui ne sont pas encore présentes. Si c'est une nouvelle feature ou un produit entièrement nouveau, il faudra approfondir les discussions avec les clients et les utilisateurs pour savoir comment ils résolvent leurs problèmes actuellement (idéalement, il faut leur parler directement). Ensuite, il faudra réfléchir et prototyper des solutions différentes. Cela engendrera beaucoup de discussions, alimentées par ce que votre équipe apprendra, qui permettront de prendre une décision sur ce qu'il faut précisément développer pour être utile aux clients, aux utilisateurs et à votre organisation. Et après tout ce travail, vous aurez encore la possibilité de rejeter cette idée.

La décision « no go » est un excellent résultat d'une discussion d'opportunité. Rappelez-vous qu'il y a toujours plus à développer qu'il n'y a de temps pour le faire. Après votre discussion, si l'opportunité ne semble pas prometteuse, jetez-la immédiatement. C'est une bonne idée d'associer à la discussion les personnes qui ont soutenu l'idée, en espérant qu'elles arrivent à la même conclusion que vous.

Il est possible que votre groupe n'ait peut-être pas suffisamment d'informations pour prendre une décision « go » ou « no go ». Dans ce cas, faites une liste de ce que vous avez à connaître et cherchez ensemble les renseignements dont vous avez besoin.

Si vous ne pouvez toujours pas prendre une décision, vous pouvez toujours la remettre dans le backlog d'opportunités pour en discuter plus tard. Cela s'appelle la procrastination et je m'y adonne souvent.

La matrice d'opportunité

En étudiant des opportunités pour des produits, j'avais utilisé le modèle d'évaluation d'opportunité de Marty Cagan comme point de départ. Depuis peu, je préfère utiliser une approche pilotée par matrice.

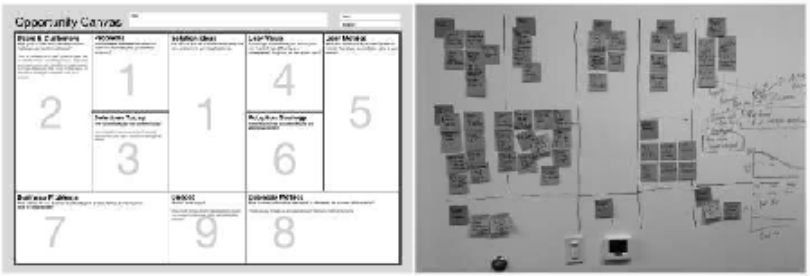
L'approche par matrice pour étudier les modèles économiques, telle qu'elle est décrite dans l'ouvrage d'Alexander Osterwalder et Yves Pigneur, *Business Model nouvelle*

© Dunod – Toute reproduction non autorisée est un délit.

génération : Un guide pour visionnaires, révolutionnaires et challengers (Pearson), est une méthode efficace pour la collaboration des groupes qui évaluent le démarrage d'une entreprise. Mais pour moi et pour la plupart des gens avec qui je travaille, nous ne cherchons pas à lancer une nouvelle entreprise ou un nouveau produit. Nous cherchons souvent à ajouter une fonctionnalité importante à notre produit. Cela ne devrait cependant pas nous empêcher d'utiliser une approche similaire par matrice pour évaluer les opportunités des produits.

Une matrice organise l'information dans l'espace : c'est grand et tout est rassemblé au même endroit si bien qu'un groupe peut voir et travailler avec la matrice, ce qui est difficile à faire avec un document imprimé, et ce type de représentation vous permet d'afficher les dépendances. Les informations sont adjacentes aux informations dont elles dépendent.

Voici à quoi ressemble une matrice :



Et voici à quoi ressemble le travail collaboratif pour recueillir des informations sous la forme d'une matrice :

On obtient de réels avantages à employer une approche par matrice :

– En voyant les points importants d'une opportunité en un seul coup d'œil.

– En visualisant les relations entre ces points importants.

– En créant un canevas grâce au travail collaboratif, au partage de la réflexion, au sentiment d'appartenance au projet on augmente l'apport de chacun.

En tant qu'équipe de découverte de produit, vous devez d'abord remplir la matrice en vous basant sur ce que vous comprenez actuellement. Impliquez les parties prenantes, les experts du domaine, ou quiconque susceptible d'apporter des informations importantes dans la conversation.

Utilisez des Post-it® parce que vous allez changer d'avis au fur et à mesure que la discussion progresse. Améliorez la matrice au fur et à mesure que vous en apprenez davantage.

Vous pouvez continuer avec la matrice, en commençant par la première case et en allant jusqu'à la neuvième. Toutefois, si vous n'avez pas de bonnes réponses pour une case, enregistrez maintenant ce que vous savez *vraiment*, ou bien vos hypothèses, et passez à la suite.

Remplissez la matrice en un seul flux et lisez-la dans un autre

Les cases de la matrice sont numérotées dans un ordre logique pour discuter d'une opportunité. Mais si vous partagez la matrice avec quelqu'un d'autre, vous pouvez la lire de gauche à droite et de haut en bas. Vous remarquerez que le flux de gauche à droite est extrait de la séquence « maintenant » à « plus tard » du modèle output versus outcome qui a déjà été présenté. Vous noterez également que le sens de haut en bas va des besoins des utilisateurs aux besoins de l'entreprise.

La matrice n'est pas un formulaire à remplir, mais plutôt un ensemble de sujets de discussion qu'il faut affiner de manière itérative pour en améliorer la compréhension. N'oubliez pas que « *la conception en communauté n'est pas la conception en comité* ». L'implication de nombreuses personnes aide tout le monde à apprendre plus vite, mais en fin de compte, la décision go/no go de poursuivre avec une opportunité appartient au PO. Les meilleurs PO demandent à leurs équipes de l'aider à prendre la décision et en général ils tombent d'accord.

Voici le flux des cases de la matrice d'opportunité.

1. Problèmes ou solutions

Idéalement, on devrait commencer par une vision claire d'un problème que l'on tenterait de résoudre. Toutefois, le monde est rarement idéal. On nous donne souvent une idée de fonctionnalité ou d'amélioration, puis il faut travailler à rebours pour comprendre le problème. Commencez par ce que vous avez.

Idées de solution

Listez le produit, la fonctionnalité ou les idées d'amélioration capables de résoudre les problèmes de votre public cible.

Problèmes

Quels sont les problèmes actuels de vos clients et utilisateurs potentiels auxquels répond votre solution ?

Si vous développez un produit de divertissement, comme un jeu ou un outil pour partager des trucs amusants sur un réseau social, il est possible qu'ils n'aient pas de réel « problème » à résoudre, mais simplement le désir d'être divertis.

2. Utilisateurs et clients

Quels types d'utilisateurs et de clients ont les problèmes traités par votre solution ? Recherchez les différences dans les objectifs des utilisateurs ou des utilisateurs qui affecteraient leur utilisation du produit. Séparez les utilisateurs et les clients selon des types basés sur ces différences. C'est une mauvaise idée de cibler « tout le monde » avec votre produit.

© Dunod – Toute reproduction non autorisée est un délit.

3. Solutions actuelles

Comment les utilisateurs traitent-ils leurs problèmes aujourd'hui ? Listez des produits concurrents ou des solutions de contournement qu'emploient vos utilisateurs.

4. Valeur utilisateur

Si vos utilisateurs-cible utilisent déjà votre solution, comment pourraient-ils alors faire les choses différemment ? Et dans quelle mesure cela leur sera-t-il bénéfique ?

5. Indicateurs utilisateur

Quels sont les comportements des utilisateurs que vous pourriez mesurer pour s'assurer qu'ils adoptent, utilisent et valorisent votre solution ?

6. Stratégie d'adoption

Comment les clients et les utilisateurs vont-ils découvrir et adopter votre solution ?

7. Problème de l'entreprise

Quels problèmes de votre entreprise seront résolus par le développement de ce produit, de cette feature ou de cette amélioration ?

8. Indicateurs entreprise

Quels sont les indicateurs de performance de l'entreprise qui seront impactés par le succès de cette solution ? Ces indicateurs sont souvent une conséquence du changement de comportement des utilisateurs.

9. Budget

Combien d'argent allez-vous budgétiser pour découvrir, développer et affiner cette solution ?

13.3 UNE OPPORTUNITÉ NE DEVRAIT PAS ÊTRE UN EUPHÉMISME

Je sais que votre entreprise n'utilise probablement pas les opportunités. En effet, si votre entreprise est du genre de celles avec lesquelles j'ai travaillé, vous avez une roadmap remplie de trucs que vous êtes censé développer. Vous pouvez même considérer que les opportunités sont vos « spécifications », et il est possible que le fait de prendre une décision « no go » sur une opportunité ne soit pas de votre ressort. La vérité est que nous ne devons vraiment pas transformer toutes ces belles idées en logiciel, quelle que soit la position hiérarchique de la personne qui a eu une idée géniale.

Utilisez cette première grande conversation de story pour vous vous faire une idée du travail dans lequel votre équipe est susceptible de s'engager. Même si la réponse à la question « go/no go » peut être un « go », assurez-vous qu'à l'issue de la conversation vous avez une compréhension mutuelle des problèmes à résoudre.

Si ce n'est pas à vous ni à un membre de votre équipe de valider cette opportunité, alors assurez-vous d'inclure dans la conversation les personnes décisionnaires. Si elles ne sont pas disponibles, ayez quand même la conversation et faites des hypothèses sur le « qui », le « quoi » et le « pourquoi », puis partagez ces hypothèses avec les décideurs. Je vous promets qu'ils vous corrigeront si vous avez mal compris. Le fait de discuter de leurs rectifications constitue véritablement le début de la conversation.

© Dunod – Toute reproduction non autorisée est un délit.

13.4 STORY MAPPING ET OPPORTUNITÉS

J'apprécie beaucoup les story maps, mais je n'en utiliserais pas pour gérer les opportunités car ce sont en général de plus gros morceaux.

Cependant, l'un des atouts des story maps est qu'elles offrent un moyen vraiment efficace de prendre du recul et d'avoir une vision d'ensemble du produit tel qu'il est actuellement. Créez une carte de votre produit actuel pour trouver les opportunités, ou pour étudier celles que vous avez déjà.

Essayez d'élaborer une carte simple et de très haut niveau de votre produit existant. Il s'agit d'une carte « actuelle » semblable à celle que vous avez créée au chapitre 5 sur les activités que vous faites quand vous démarrez votre journée (au fait, vous l'avez bien créée cette carte ?). On rencontre ces types de cartes sous diverses formes depuis un certain temps et on les appelle souvent *cartes de voyage (journey maps)*. Pour en faire une de votre produit actuel, cartographiez simplement le flux des utilisateurs au cours de leurs principales activités. Utilisez cette carte pour fournir le contexte de vos opportunités. Pour ce faire, ajoutez chaque opportunité dans le corps de la carte. Utilisez une couleur différente de Post-it® ou de fiche pour les distinguer clairement. Vous pouvez ainsi voir apparaître les points névralgiques des opportunités, ces endroits du flux de vos utilisateurs où il y a une densité plus élevée des idées et probablement des soucis pour vos utilisateurs.

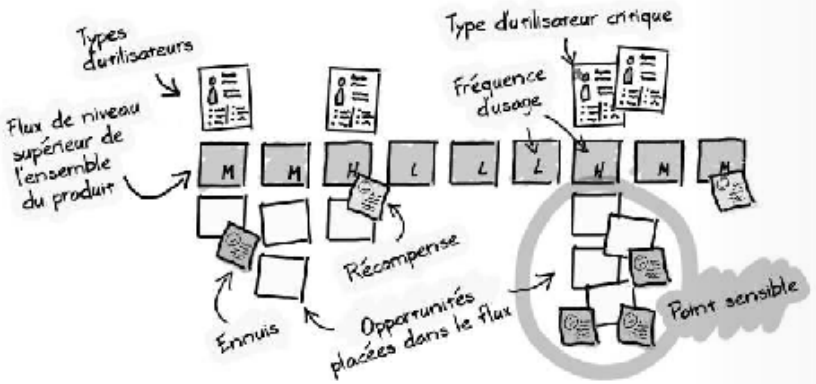
Regardez les utilisateurs engagés dans chaque activité et la fréquence de cette activité. Les opportunités qui concernent des activités avec une fréquence élevée et des utilisateurs importants sont probablement celles sur lesquelles vous devriez vous concentrer en priorité.

Vous pouvez utiliser cette même carte pour y ajouter des fiches pour les choses dont les utilisateurs se plaignent actuellement. Pour équilibrer les choses, examinez les parties du produit que les utilisateurs apprécient et incorporez-les dans la carte. Si vous trouvez des endroits où il y a beaucoup de soucis pour les utilisateurs, mais où vous n'avez pas encore identifié d'opportunités, vous devriez probablement en chercher.

Journey mapping et génération de concept

Ben Crothers, Atlassian

Étant donné que nous fournissons plus de dix produits différents, il faut nous assurer que nous les concevons, développons et améliorons d'une manière qui corresponde à la façon dont nos clients les utilisent ensemble, et non pas individuellement. Dans le cadre d'un projet pour découvrir comment nos produits pourraient mieux fonctionner ensemble, nous avons formé une équipe multidisciplinaire pour cartographier l'expérience du client de bout en bout (rechercher, évaluer, acheter, ainsi qu'obtenir de l'aide et ajouter d'autres produits et services).



Ce fut un travail énorme. Pour faciliter la décomposition, nous avons tout d'abord cartographié l'expérience à un niveau élevé, puis nous avons divisé en sous-groupes pour développer les grandes étapes de ce voyage. Au départ, nous l'avons fait en remplissant un mur avec les phases, les actions et les questions que traversent les clients, en effectuant un codage par couleur, puis nous avons ajouté des points problématiques, des opportunités et des hypothèses.

Nous avons beaucoup gagné en perception car nous avons été en mesure de retracer une story de bout en bout plutôt qu'une simple expérience basée sur une fonctionnalité partielle. Nous nous sommes rapidement rendu compte, par exemple, que certaines parties, comme l'installation des produits ou l'assistance, n'étaient pas rattachées à une partie d'un parcours linéaire et devaient être combinées pour être plus détaillées.

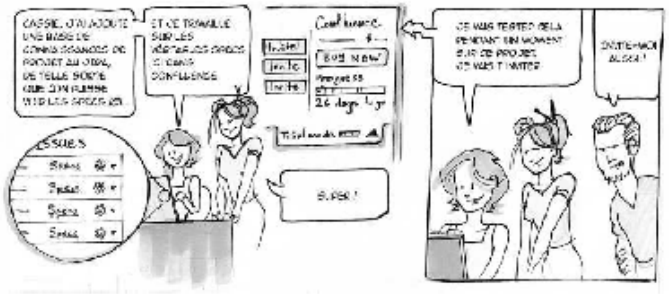
Les diverses équipes et autres parties prenantes ont été sollicitées pour étoffer les voyages qui ont été tous rattachés à différents points du squelette de haut niveau, de telle sorte que nous avons capturé et validé autant de connaissances que nous avons pu.

© Dunod – Toute reproduction non autorisée est un délit.

Il y a alors eu une explosion de concepts pour améliorer et réinventer les différentes parties des trajets détaillés. Tous ces concepts ont été concentrés et rédigés sur des fiches collées sur le mur en désordre le long du voyage.

Chaque membre de l'équipe a expliqué ses concepts, et nous avons tous voté sur l'efficacité des concepts avec des autocollants, en fonction de leur faisabilité, leur viabilité et de leur caractère souhaitable.

Nous étions alors en mesure de définir la vision générale et transversale de nos produits de l'usage idéal de nos clients, basée sur ces voyages et des concepts validés. Le voyage a également donné lieu à la création d'une bande dessinée de 20 pages, qui a transformé le voyage, ainsi que tous les personas, les scénarios et les concepts sous-jacents, en un storyboard unique, ce qui a facilité la communication à l'ensemble de l'organisation. Cela permet de rester informé des nombreuses améliorations de produits qui se poursuivent encore aujourd'hui.



Ce qui est vraiment intéressant pour nous, c'est que beaucoup de ceux qui sont à présent impliqués dans l'élaboration et l'affinage de ces améliorations ont d'abord participé à la génération des concepts. Bien que cet exercice ait été réalisé il y a plus de huit mois, cette conception et cette compréhension mutuelles rendent les choses beaucoup plus efficaces maintenant.

Hypothèses complexes avec remapping de répétition

Erin Beierwaltes et Aaron White

1. Une carte pour gouverner toutes les autres

Après les visites des clients, les interviews et les explorations, le PO dispose d'un « guide de voyage » dont il est persuadé qu'il trace le chemin vers le meilleur bénéfice pour le client. Maintenant, il est temps de partager cette carte avec d'autres personnes et de se mettre en marche.

2. Questionnez tout

Mais après quelques conversations, nous avons commencé à nous poser des questions sur ce que nous aurions oublié. Quelles sont les hypothèses simples qui pourraient avoir un impact plus important que prévu ? Nous avons besoin d'un moyen rapide pour contester nos hypothèses, trouver des failles, *et* améliorer la compréhension mutuelle parmi tous ceux qui font partie du projet.

3. Plantez le décor

Nous avons réuni l'équipe, assigné des personas (il y en a plusieurs) et décrit l'impact souhaité qui inciterait naturellement les personas à utiliser la partie du produit qui testerait nos hypothèses. Nous n'avons *pas* rédigé d'instructions étape par étape, mais une simple directive : « *Jouez le rôle de cette personne, avec cet objectif* ». Répétition !

4. Répétition

Pendant que les acteurs ont joué leurs rôles et essayé d'atteindre les objectifs assignés, quelques observateurs ont joué le rôle d'un auditoire silencieux.

Le PO a observé les chemins empruntés, répondu à quelques questions, en évitant d'inciter à prendre un chemin particulier. Le IxD (concepteur d'interactions) a observé les comportements, les commentaires et les réactions.

5. Reconstruisez la carte

On n'a pas montré la carte originale, mais au lieu de cela, on a amené tous les acteurs à créer une nouvelle carte qui décrit la manière dont ils ont réalisé leurs objectifs ; certains ont pris des chemins différents.

6. Déceptions et bonnes surprises

Après la reconstruction, nous avons demandé à chacun des acteurs de partager ses déceptions (les choses qu'il a contestées, qui l'ont frustré ou dérouté) et ses bonnes surprises (les choses qui lui ont paru astucieuses, intéressantes ou intuitives). Un sentiment réel et nouveau d'empathie, de compréhension et d'excitation a émergé dans la conversation.

7. Observations

Enfin, ce fut au tour de l'auditoire de poser des questions sur les comportements intéressants qu'il avait observés. Nous avons été surpris du nombre de choses dont les acteurs n'étaient pas conscients.

8. Bénéfice !!!

En seulement deux heures et demie, nous avons amélioré la compréhension partagée de la nouvelle solution, et seule une conversation permet cela. Les acteurs ont parlé avec une réelle empathie au nom des clients dont ils ont joué le rôle, et l'équipe de découverte a bien mieux compris ce qui fonctionnait et ce qui nécessitait plus d'expérimentations.

13.5 SOYEZ RIGOUREUX

Vous n'aidez personne si vous acceptez tout. Débarrassez-vous rapidement des opportunités qui n'offrent pas beaucoup d'espoir d'obtenir l'impact souhaité. Travaillez avec les parties prenantes de l'entreprise pour qu'elles puissent vous aider à prendre ces décisions.

Maintenant que vous avez pris une décision « go », c'est le moment de retrousser vos manches et de vous mettre au travail. C'est ce que nous allons voir au chapitre suivant.

© Dunod – Toute reproduction non autorisée est un délit.

14

Utilisation de la découverte pour améliorer la compréhension mutuelle

Des modèles simples qui décrivent le développement agile commencent souvent sur le côté gauche avec une grande liste, le *backlog de produit*. Aujourd'hui, je trouverais cela drôle si je ne savais pas que certaines personnes considèrent vraiment cela aussi simple. Pourtant, l'obtention d'un bon backlog de produit qui soit applicable nécessite beaucoup de travail car il ne va pas se matérialiser tout seul. Et il ne s'agit certainement pas du résultat de l'enregistrement d'une liste de choses que les gens veulent développer. C'est un processus délibéré de découverte axé d'abord sur l'apprentissage de nombreuses informations sur le « qui », le « quoi » et le « pourquoi ».

14.1 LA DÉCOUVERTE NE CONCERNE PAS LE DÉVELOPPEMENT

Le travail de découverte ne concerne pas le développement de logiciels. C'est l'apprentissage et l'élaboration d'une meilleure compréhension de ce que nous pourrions développer. Il s'agit de poser et de répondre à des questions telles que :

- Quels *problèmes* allons-nous vraiment résoudre ?
- Quelles solutions pourraient être *valables* pour notre organisation pour les clients qui achètent ou adoptent le produit ?
- À quoi ressemble une solution *utilisable* ?
- Qu'est-il *possible* de développer étant donné le temps et les outils dont nous disposons ?

© Dunod – Toute reproduction non autorisée est un délit.

C'est en posant ces questions sur une opportunité et en commençant à y répondre que débute votre première phase de décomposition des tâches. Tous les détails sur le produit ou la feature dont vous discutez deviennent les titres de stories plus petites. Et chacune de ces petites stories peut donner lieu à des discussions encore plus profondes et à des stories encore plus petites.

Toutes ces discussions de découverte ne se contentent pas d'engendrer plus de stories. N'oubliez pas que les discussions de story impliquent la création de beaucoup d'autres modèles simples qui représentent ce que nous comprenons. Nous aurons besoin de ces modèles pour que tout le monde se comprenne.

> Si la seule chose que vous créez lorsque vous voulez donner du sens à une grande opportunité ambiguë c'est de multiplier le nombre de stories plus petites, alors vous êtes probablement dans l'erreur.

14.2 QUATRE ÉTAPES ESSENTIELLES DE LA DÉCOUVERTE

Quand j'ai une grande idée, ou même une petite qui nécessite d'être clarifiée, je suis cette progression de discussions pour obtenir les détails nécessaires pour mieux comprendre s'il s'agit d'une solution qui mérite d'être développée :

1. *Formulez* l'idée d'un point de vue « business »
2. *Comprenez* les clients et les utilisateurs et la manière dont vous les aidez.
3. *Envisagez* votre solution.
4. *Minimisez et planifiez* pour identifier la solution viable la plus petite et comment la développer.

14.2.1 Formulez l'idée

Si vous utilisez vraiment un backlog d'opportunités, et tenez réellement une discussion d'opportunité pour prendre la décision de commencer la découverte, alors vous êtes dans le droit chemin. Les discussions de cadrage permettent de commencer la découverte ciblée. Impliquez les personnes qui vont travailler ensemble pour mieux comprendre cette opportunité.

Les discussions de cadrage permettent de définir les limites du travail que vous entreprenez. Si vous êtes clair sur les raisons pour lesquelles vous développez quelque chose et sur les personnes qui utiliseront votre produit, alors votre équipe pourra mieux arrêter les discussions sur les solutions qui ne résolvent pas le problème sur lequel vous vous concentrez.

14.2.2 Comprenez les clients et les utilisateurs

Utilisez les discussions sur les clients et les utilisateurs pour vous faire une meilleure idée des personnes qui se serviront de votre produit et qui en tireront parti. Faites participer ceux qui ont une connaissance profonde des utilisateurs et ceux qui ont besoin de l'acquérir.

Ébauchez des personas simples

J'aime créer des esquisses simples de *personas* avec une petite équipe de découverte pour forger la compréhension mutuelle de mes utilisateurs. Un persona est un exemple de votre utilisateur cible assemblé à partir de faits, et parfois d'hypothèses, que vous avez collectés sur vos utilisateurs. La création de personas nous aide à regarder le logiciel à travers les yeux de nos utilisateurs. Les personas sont des outils pratiques.

J'ai créé ce personnage simple avec un groupe de l'organisation caritative Mano a Mano, qui aide les gens en Bolivie à faire toutes sortes de choses, que ce soit la construction de routes, le soutien scolaire ou bien les soins médicaux. Notre discussion de ce jour portait sur les petits donateurs par Internet, le genre de personnes qui ne disposent pas d'une énorme somme d'argent, mais qui veulent que ce qu'ils *donnent* soit utilisé à bon escient. Nous nous attendions à ce que les gens comme Chuck trouvent l'association Mano a Mano sur le Web ou en entendent parler via Twitter ou Facebook.

Ensemble, nous avons créé ce persona à l'aide d'un tableau papier. C'est une activité amusante et rapide, avec beaucoup de gens qui s'interpellent et qui échangent des informations.

Maintenant, si vous êtes un designer UX expérimenté qui a déjà créé des personas, il se peut que vous vous sentiez mal à l'aise. Pour ceux qui n'ont pas cette expérience, les bons personas sont créés à partir de bonnes données recueillies dans le cadre de solides recherches. Les pros du UX design verront sans doute d'un mauvais œil des membres d'une équipe en train de crier et de griffonner des choses sur tableau papier, car cela ne semble pas rigoureux. Dans ces conditions, arrêtez de crier vos suppositions. Discutez de ce que vous savez et de ce que vous avez pu constater.

© Dunod – Toute reproduction non autorisée est un délit.

Racontez des stories. Impliquez dans la discussion les gens qui ont une expérience directe avec les utilisateurs. Si vous avez fait beaucoup de recherches, faites-en profiter les autres dans la discussion. Identifiez des détails plus pertinents pour l'opportunité que vous développez et mettez-les dans le persona. Filtrez le bruit. Lorsque vous avez terminé, ayez une discussion honnête sur le caractère hypothétique du persona.

« Nous avons déjà créé des personas. Ce sont de belles descriptions, que nous avons affichées sur le mur. » J'ai entendu cela de nombreuses fois. Mais soyez honnête avec vous-même. La plupart des gens n'ont pas lu ces descriptions, n'est-ce pas ? Et la moitié des personnes qui les ont *vraiment* lus l'ont fait juste pour s'en moquer. Peut-être suis-je cynique, mais je l'ai souvent constaté.

Construisez des personas de manière collaborative pour améliorer la compréhension avec l'équipe qui va développer le produit. Cela permet de prendre vraiment en compte les aspects les plus importants du persona.

> Créez ensemble des personas légers pour améliorer la compréhension et favoriser l'empathie au sein de l'équipe.

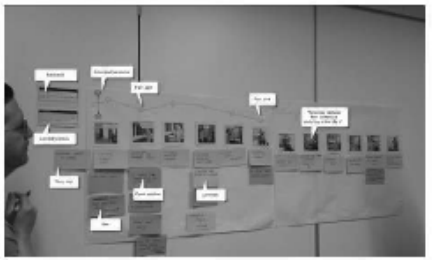
Je crée des personas simples pour chaque type d'utilisateur qui peut se servir de la feature dont nous parlons. Si je travaille vite, je peux juste lister différents types d'utilisateurs ou de rôles qui vont se servir du logiciel et noter quelques détails à leur sujet. Vous vous souvenez de Gary au chapitre 7 ? Une des piles de fiches juste à côté de lui représentait exactement cela : une liste d'utilisateurs et sous forme de puces, une liste inventoriant des informations pertinentes à leur sujet.

Créer des profils d'organisation

Si vous développez un produit susceptible d'être acquis par des organisations, par exemple, un logiciel de comptabilité, prenez le temps de répertorier différents types d'organisations et de collecter quelques détails à leur sujet, car ce sont vos clients. On appelle souvent *profil d'organisation* un type d'organisation avec quelques renseignements sur son fonctionnement. C'est mon ami Lane Halley qui m'a initié à la création d'exemples de profils d'organisation, un peu comme on crée un persona.

Cartographiez la manière dont les utilisateurs travaillent actuellement

Vous pouvez approfondir les choses et cartographier la façon dont vos utilisateurs travaillent actuellement sans votre produit, ou bien avec la version en cours de votre produit actuel. Si vous avez joué le jeu au chapitre 5, vous avez construit une story map sur la façon dont vous faites les choses actuellement. Le fait d'étudier la manière dont vos utilisateurs travaillent aujourd'hui aidera votre équipe de découverte à vraiment comprendre les problèmes qu'ils vont résoudre.



Ces photos de Duncan Brown du groupe Caplin montrent quelque chose qu'ils appellent *carte de voyage narrative*. C'est une carte qui décrit une story sur le côté « maintenant » du modèle « maintenant-et-plus tard » que j'ai présenté au début de ce livre (à la fin de l'introduction « À lire en premier »). Cela ne décrit pas notre solution idéale, mais la façon dont les gens atteignent leurs objectifs actuellement, en n'omettant pas les problèmes.

Le corps de la carte contient des faits, des observations, des ennuis et des succès. Lorsque vous cartographiez ce que vous comprenez de la situation actuelle, vous voyez des « points chauds », des zones du flux où il y a beaucoup de problèmes. Vous trouvez également des motifs de satisfaction, c'est-à-dire les succès qui couronnent les efforts de vos utilisateurs à la fin d'une série d'étapes. Vous pouvez développer de bons produits en évacuant les soucis et en multipliant les succès. Utilisez cette carte comme une introduction aux séances de remue-méninges sur les solutions, ou pour valider que la solution que vous avez en tête résout vraiment les problèmes.

© Dunod – Toute reproduction non autorisée est un délit.

14.2.3 Envisagez votre solution

Vous avez à présent clarifié les raisons pour lesquelles vous développez votre produit dans une perspective professionnelle ; vous avez étudié en profondeur les utilisateurs et les clients afin de mieux connaître leur monde. Il est temps d'imaginer l'avenir, pour envisager la solution et voir comment vos clients et vos utilisateurs cibles vont s'en servir.

Cartographiez votre solution

C'est le moment de gloire des story maps, tout au moins pour moi. J'utilise un story map pour imaginer la vie de mes utilisateurs avec la solution que nous développons. À la fois Gary et l'équipe de Globo.com (cf. les deux premiers chapitres) ont construit des cartes de cette manière-là. Comme nous l'avons déjà évoqué dans les premiers chapitres, les étapes empruntées par les utilisateurs de la story forment le flux de gauche à droite. Dans le chapitre 4, nous avons vu qu'il s'agit de tâches utilisateur, de courtes phrases verbales qui, quand elles sont lues de gauche à droite, décrivent la story. Les tâches de grain plus fin et les autres détails s'empilent verticalement sous chaque étape. S'il s'agit d'une longue story, concentrez les groupes d'activités pour créer une carte à trois niveaux.

Des mots et des images

Avez-vous déjà été dans la situation où vous décrivez une idée de produit à un développeur, et où vous êtes agréablement surpris de l'entendre dire « Oh, c'est simple et cela ne devrait pas prendre longtemps à développer » ? Mais quand il commence à développer, vous vous rendez compte que le développeur a imaginé quelque chose de beaucoup plus simple que ce que vous aviez en tête. Par exemple, vous avez décrit un site de vente en ligne et vous avez imaginé quelque chose comme eBay ou le Marketplace d'Amazon. En revanche, le développeur a imaginé quelque chose qui ressemble plus à Craigslist, et c'est pourquoi vous avez été satisfait de l'estimation du temps de développement. Au cours de ces dix dernières années, j'ai appris que les mots seuls ne suffisent pas.

> Visualisez votre interface utilisateur pour forger la compréhension mutuelle de la solution.

Si vous avez des designers UX dans votre équipe, c'est le moment qu'ils se mettent à faire des maquettes. Dessinez des écrans individuels et affichez-les au-dessus de la carte afin qu'ils soient visibles. Vous vous retrouverez avec quelque chose qui ressemble à un storyboard.

Visualisation de l'expérience totale

Josh Seiden, avec des illustrations de Demian Repucci

Un jour, j'ai reçu un appel de Robert, gestionnaire de design récemment embauché dans une startup du secteur éducatif. La société était dans les premiers stades d'un grand projet et elle embauchait à tour de bras pour respecter un échéancier serré afin de produire un système énorme. Il y avait cependant un défi à relever car elle avait de la difficulté à trouver comment aborder les énormes problèmes de conception auxquels elle était confrontée. Allais-je pouvoir l'aider ?

Quand je suis arrivé dans les bureaux quelques jours plus tard, Robert, à la fois fier et accablé, m'a fait faire le tour du propriétaire. La société avait embauché un grand cabinet de conseil pour l'aider à élaborer les spécifications de son projet, et les consultants avaient réalisé une quantité impressionnante de travail. Tous les murs des bureaux étaient recouverts de feuilles de papier brun, et chacune de ces feuilles était à son tour recouverte de fiches et de notes sur des Post-it® : des spécifications sous la forme de user stories. Il y en avait des milliers. Alors que Robert m'accompagnait le long des murs de stories, j'ai remarqué que toutes les stories étaient organisées par module fonctionnel : un mur pour l'éditeur de texte, un mur pour une application de scolarité. Et des modules de cours : un mur pour les sciences, un mur pour l'anglais. J'ai eu du mal à construire une image mentale du système.

Robert était en train de constituer une équipe de design et tentait aussi de segmenter le problème. Quand nous parlions des besoins de son équipe, nous nous sommes rendu compte que nous pourrions utiliser une story map pour organiser les milliers de user stories de façon à faciliter la conception et à aider les équipes de développement à fonctionner à partir d'une vision partagée.

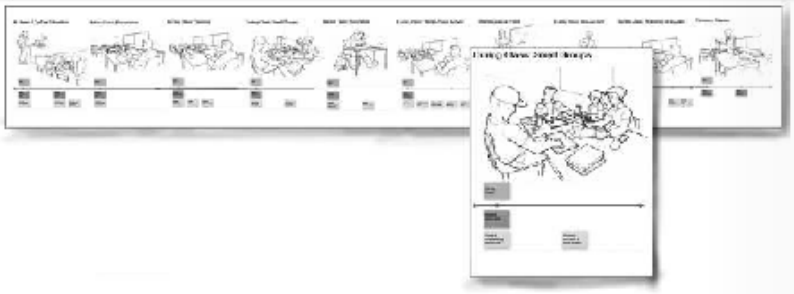
Par hasard, quelques semaines plus tôt, j'avais participé à un atelier animé par une équipe d'artistes utilisant des storyboards. Cet atelier visait à aider des entrepreneurs à formuler une vision d'avenir de leurs idées commerciales. Ces artistes qui travaillaient rapidement étaient assis avec les entrepreneurs, et dessinaient leurs stories en exprimant les idées sous la forme de storyboards (de mini bandes dessinées qui racontaient chaque story avec une grande clarté). J'ai décidé de combiner cette approche avec le story mapping et j'ai appelé l'un des artistes qui m'avait impressionné dans cet atelier, Demian Repucci.

Au cours des semaines qui ont suivi, Demian et moi-même avons rencontré Robert et son équipe, ainsi que les chefs de produit des diverses parties du système. Nous avons mis l'accent sur le workflow haut niveau, c'est-à-dire les principaux cas d'utilisation du système. Lors des réunions, Demian dessinait dans son carnet et j'utilisais des fiches et des notes sur des Post-it® pour décrire les cas d'utilisation sur les murs de la salle de conférences. Après les réunions, Demian retournait dans son studio pour illustrer les moments clés, et j'ai utilisé Omnigraffle pour produire des versions propres des story maps que nous avions rédigées lors de nos réunions.

Avec Robert, nous avons décidé que la chose la plus intéressante que nous pourrions produire pour l'équipe était la structure organisatrice, et nous avons donc réalisé une série d'affiches à imprimer sur du papier A3 et à coller sur les murs pour former la « colonne vertébrale » de la story map. Les équipes pourraient alors utiliser cela de façon indépendante pour organiser leurs user stories d'une manière nouvelle. Au lieu d'une vue centrée sur le module qui ne se prête pas au développement itératif, nous avons à présent une approche axée sur l'usage qui peut être décomposée en releases transversales aux modules.

© Dunod – Toute reproduction non autorisée est un délit.

L'approche Design Studio est une façon d'envisager l'expérience utilisateur en impliquant toute l'équipe. Design Studio est un moyen rapide, simple et collaboratif



d'impliquer un grand groupe dans une démarche volontaire d'idéation, ce qui est un mot pompeux pour dire que l'on doit produire le plus d'idées possible. Vous allez vite apprendre qu'aucune personne n'a les meilleures idées et, qu'au contraire, les meilleures idées sont souvent une combinaison de plusieurs idées de personnes différentes ainsi que des discussions complémentaires pour les étayer. Design Studio (et d'ailleurs tout processus simple de formation des idées) est souvent le contraire de ce que la plupart des gens font, moi y compris : se lancer avec la première idée qu'on juge opérationnelle. La première description que j'ai lue de Design Studio a été rédigée par Jeff White et Jim Unger (http://portal.acm.org/citation.cfm?id=1358650) et je me suis demandé pourquoi je n'avais pas toujours fait cela. J'ai par la suite appliqué cette méthode avec des équipes de développement, des parties prenantes et même des clients et des utilisateurs finaux.

Quelle que soit l'approche que vous utilisez, combinez des idées, affinez-les pour aboutir à une compréhension mutuelle de ce à quoi pourrait ressembler le logiciel.

Il n'est pas toujours agréable de constater que la visualisation de votre solution vous aidera à détecter les choses que vous avez oubliées dans la carte. Vous allez peut-être devoir ajouter, modifier ou réorganiser la carte pour prendre en charge ce que vous avez visualisé. Ne vous inquiétez pas : c'est une bonne chose.

Recette Design Studio

Design Studio est une approche rapide et collaborative de l'idéation. Il y a beaucoup de façons de réaliser cela correctement, mais voici ma recette simple :

1. *Invitez un groupe de personnes* dont vous appréciez les opinions et les idées, et dont vous aurez besoin de l'acceptation et de la compréhension pour développer le produit. Un groupe de huit à douze personnes est idéal.
2. *Décrivez le problème que vous voulez résoudre*. Critiquez le travail que vous avez fait pour formuler l'opportunité. Critiquez les personas et les cartes « maintenant » qui décrivent la façon dont les gens travaillent actuellement. Critiquez toutes les cartes des solutions que vous avez commencées, mais ne soyez pas trop bavard. S'ils adhèrent trop à vos idées, vous pourriez passer à côté de certaines de leurs bonnes idées.
3. *Partagez éventuellement des exemples et vos trouvailles*. Évoquez et montrez d'autres produits similaires qui sont de bons exemples. Évoquez et montrez des produits qui ne sont pas identiques, mais qui recèlent de bonnes idées qui pourraient être exploitées.
4. *Tout le monde dessine !* Donnez à chaque participant du papier, des stylos, et éventuellement des modèles de croquis ainsi qu'une limite de temps (de 5 minutes à 60 minutes ; personnellement, une durée de 15 minutes me convient parfaitement).
5. *Partagez des idées en petits groupes*. J'aime former des groupes de quatre personnes si bien qu'avec un groupe de douze, vous pouvez former trois groupes de quatre. Partagez votre meilleure idée individuellement et faites commenter les membres de l'équipe. Formez les participants à donner leur avis sur la façon dont les solutions traitent le problème car ils ont plutôt tendance à dire s'ils aiment ou non la solution. Entraînez-les à exploiter les idées des autres. Continuez individuellement pendant un temps déterminé (en général, 30 minutes sont une durée qui fonctionne bien).
6. *Demandez à chaque groupe de consolider leurs meilleures idées* en une seule solution exprimée sous la forme d'un dessin. C'est la partie la plus difficile et il faut bien compter 15 à 30 minutes pour cela.
7. *Demandez à chaque groupe de partager leurs meilleures idées consolidées* avec l'ensemble du groupe. Discutez de tout cela.
8. *Remerciez tout le monde et ramassez les croquis et les idées*. Vous-même, votre UX designer ou votre équipe de découverte devrez exploiter ces documents pour créer une maquette finale et consolidée de l'interface utilisateur. N'oubliez pas la maxime de mon amie Leisa Reichelt : « le design en communauté n'est pas le design en comité ». Vous aurez beaucoup de bonnes idées qui s'opposent et quelqu'un devra prendre des décisions difficiles.

Les deux premières photos sont publiées avec l'aimable autorisation de Edmunds.com, et la troisième avec l'aimable autorisation de Lane Halley.

© Dunod – Toute reproduction non autorisée est un délit.

Validez le caractère complet des choses

Combler les détails est une des choses dans lesquelles notre esprit excelle. Par exemple, quand on voit deux images d'une bande dessinée, notre cerveau peut imaginer ce qui s'est passé entre les deux cases. Il s'agit d'un truc intéressant pour les bandes dessinées, les romans et les films, mais quand on l'applique au logiciel, nous imaginons souvent les fonctionnalités les plus intéressantes tout en négligeant les choses nécessaires qui se passent au milieu. Si l'on file la métaphore cinématographique, c'est un peu comme si on parlait seulement des poursuites en voiture et des fusillades en laissant de côté l'histoire qui explique pourquoi toutes ces actions ont eu lieu.

Décrire toute l'étendue des stories de vos utilisateurs en utilisant une carte vous aide à ne pas oublier de parler de ces détails essentiels qui assurent le lien. Généralement, vous verrez que la fonctionnalité vraiment importante à laquelle vous pensiez

nécessitait un paramétrage par l'utilisateur au début de la story, ce qui engendrait des changements dans les rapports et les notifications plus tard dans la story. Cela peut avoir encore des conséquences pour vos autres idées de nouvelles fonctionnalités. Par exemple, les administrateurs peuvent avoir à gérer des problèmes de sécurité, ou les gestionnaires peuvent vouloir contrôler la façon dont la fonctionnalité est utilisée par leur personnel.

Validez les problèmes d'ingénierie

Revenons à notre métaphore cinématographique : si vous voulez faire ce film, vous aurez besoin commencer à penser à la manière et à l'endroit où vous aller tourner ce film. Vous devrez réfléchir aux types d'effets spéciaux qui seront nécessaires. À un moment donné, vous aurez besoin de dépasser le scénario pour examiner les détails techniques de fabrication de votre film.

Une story map de votre logiciel est utile pour le même genre de discussion que vous auriez pour faire un film. Discutez de la carte de votre solution avec des ingénieurs et des architectes avant d'aller trop loin. Avoir une vision d'ensemble les aide à réfléchir à des contraintes d'ingénierie plus importantes qui pourraient engendrer des solutions imparfaites. Ils peuvent vous avertir que votre solution peut sembler intéressante, mais qu'elle est impossible à développer compte tenu de votre architecture actuelle et du temps dont vous disposez. Ils peuvent souvent suggérer d'autres façons de faire les choses qui fourniront à vos utilisateurs une expérience équivalente, mais qui seront moins coûteuses à développer.

Ces ingénieurs d'une grande compagnie d'assurances ont parlé devant cette carte pendant un bon moment. Au cours de leur conversation, ils ont détecté un problème à propos de leur produit sur cette grande carte, et le moteur des règles métier de leur produit a dû être modifié. Le fait d'avoir une vision d'ensemble les a aidés à se confronter directement à la complexité des choses. Ils ont utilisé ces connaissances pour planifier ce qu'il fallait développer en premier afin d'atténuer les risques.

Jouez à « Que se passe-t-il »

Vous avez imaginé la solution du point de vue de l'utilisateur et visualisé l'expérience utilisateur. Prenez le temps de discuter de ce qui se passe sous l'interface utilisateur. Parlez des règles métier délicates, de la validation des données complexes et des systèmes pénibles de back-end ou des services auxquels vous devrez vous connecter. Ajoutez des stories à la carte qui identifient ces parties ou assurez-vous de prendre des notes sur les stories que vous avez.

C'est aussi le bon moment de passer en revue avec beaucoup d'autres personnes les informations dont vous disposez. Partagez ce que vous-même et votre groupe avez mis au point. Je vous promets que des gens poseront beaucoup de questions qui commencent par « que se passe-t-il... ». J'aime ces personnes, même si la confrontation n'est pas toujours facile, mais j'apprécie qu'elles m'aident à penser aux choses difficiles avant que je ne tombe dessus plus tard, quand l'apprentissage sera encore plus douloureux.

La métaphore cinématographique fonctionne vraiment ici. Si je devais faire un film, je voudrais un scénario et un storyboard, c'est-à-dire les esquisses des scènes clés qui m'aident à imaginer le film. Si je devais investir dans un film, j'aurais besoin de voir au moins cela pour avoir une image claire de ce que les scénaristes et les réalisateurs ont en tête. Si j'ai aimé le film, à ce stade, je vais avoir besoin d'en apprendre un peu plus sur son coût et sur sa faisabilité.

J'espère qu'ils utiliseront ces esquisses pour réfléchir plus profondément au film qu'ils vont tourner. J'aimerais qu'ils pensent aux lieux de tournage, aux types de plateaux, aux accessoires et aux effets spéciaux dont nous aurions besoin. En tant qu'important investisseur de Hollywood, j'aurais besoin du scénario, du storyboard et de beaucoup d'autres détails qui permettent d'élaborer un projet approximatif ainsi qu'une estimation. Tout cela serait nécessaire pour établir un budget et un calendrier précis de tournage du film.

C'est le genre de choses dont vous aurez besoin avant de commencer à développer votre solution.

Idées, exemples et trajets

David Hussman, DevJam

Beaucoup de gens compliquent trop le processus de découverte alors qu'il peut être assez simple tout en restant encore puissant. Dépassez la certitude mythique des « spécifications » et prenez un moment pour explorer la découverte d'idées à l'aide d'exemples et de trajets qui vous serviront de guide.

– *Utilisez le guide simple suivant :* 1) proposez une idée de produit à explorer, 2) sélectionnez des personnes susceptibles de tirer profit de cette idée, 3) créez une collection d'exemples utilisant l'idée de produit et 4) servez-vous de ces exemples pour créer une carte et les trajets qu'ils devraient emprunter. N'oubliez pas qu'en tant

© Dunod – Toute reproduction non autorisée est un délit.

Carte utilisant des exemples et des journeys

que créateur du produit, vous avez pris la responsabilité de créer des expériences enrichissantes et pas simplement plus de fonctionnalités.

– *Les idées n'ont pas besoin d'être brillantes*. Assurément, vous voulez un très bon produit, mais souvent les idées réputées brillantes sont vouées à l'échec, alors que d'autres, qui sont moins éclatantes, gagnent en légitimité quand on explore leur utilisation dans un contexte significatif.

– *La sélection des passagers n'est pas sorcier*. Ne compliquez pas outre mesure la sélection. Si vous ne savez pas par où commencer, faites simplement une liste des personnes susceptibles de bénéficier de vos idées. Formulez vos idées de manière naturelle afin qu'elles prennent vie et survivent dans la bouche et l'esprit de la communauté de développement du produit. Après les avoir un peu retravaillées, choisissez une personne avec laquelle vous allez travailler et ne vous inquiétez pas de savoir s'il s'agit de la bonne personne. Soyez prêt à apprendre de l'exploration et évitez de vous inquiéter de la pertinence de votre sélection car vous avez probablement commis des erreurs.

– *Créez une liste variée d'exemples*. C'est à ce moment-là que beaucoup de gens perdent le contrôle et tombent accidentellement dans la complexité. Commencez par un exemple simple ou évident, l'exemple le plus concret étant le meilleur. Prenez ensuite un exemple complexe et n'ayez pas peur de placer la barre haut. Vous êtes en train de créer une série de contraintes, et pas de promettre la lune. Encore une fois, prenez un exemple complexe précis. Si vous cherchez le degré de précision à obtenir et que vous êtes courageux, remplacez l'exemple en question par un test, et vous aurez fait un pas de plus vers un module de validation automatique.

– *Ajoutez d'autres exemples qui vont du plus évident au plus complexe*, puis démarrez simplement. Décrivez la story du passager à l'aide de l'exemple simple qui vous servira de guide. Racontez-la à quelqu'un d'autre et aidez-vous de cette personne pour enregistrer les stories pendant le trajet. Où se trouvent vos utilisateurs lorsqu'ils commencent ? Qu'est-ce qui les pousse à s'engager ? Que font-ils précisément ? Comment est-ce que cela se termine ? Utilisez les divers exemples pour explorer les différents trajets de vos passagers.

– *Choisissez les trajets qui vous aideront à apprendre*. Le fait de se soucier de savoir où commencer est encore un autre facteur de complexité. Parcourez la carte en sélectionnant des trajets susceptibles de vous en apprendre le plus sur votre public et leurs besoins. Encore une fois, si vous avez peur de ne pas faire la bonne sélection, il est fort possible que vous soyez dans le vrai. La meilleure façon de savoir et le meilleur investissement dans l'apprentissage, c'est de sélectionner quelques trajets, de les développer et de regarder les gens s'en servir, en personne ou par l'intermédiaire d'analyses en temps réel.

– *Éviter le piège de du produit mal adapté*. La différence entre ce dont les gens ont *vraiment* besoin, et ce dont vous pensez qu'ils ont besoin constitue le royaume du produit mal adapté. En utilisant le procédé énoncé ci-dessus, vous pouvez apprendre plus rapidement en contexte en développant et en validant un voyage à la fois.

Ce n'est pas encore le moment de faire la fête

Grâce à cette étape du flux de découverte, vous devez avoir commencé à décrire les parties de votre solution sous la forme de stories individuelles, chaque morceau faisant partie de la grande opportunité. Si vous êtes comme moi, vous avez utilisé une story map pour organiser toutes les parties. Si vous êtes plus malin que moi, vous avez inventé une meilleure façon d'organiser toutes les parties, et si tel est le cas, vous devez tout de suite m'écrire. Si vous n'êtes pas très malin, vous vous retrouverez simplement avec une grande pile. Ou pire encore, une personne aura écrit un long document de spécifications qui embrouillera tout ce que vous avez appris. S'il vous plaît, ne faites pas cela !

Quel que soit le résultat obtenu, vous en êtes au moment où la plupart des gens font la fête parce qu'ils ont « terminé les spécifications ». Ne soyez pas comme eux car vous avez une dernière étape à accomplir et c'est la plus importante.

14.2.4 Minimisez et planifiez

Vous avez imaginé une solution en mots et en images et, à ce stade, votre équipe peut ressentir une certaine fierté. Mais l'un des gros problèmes avec la découverte se produit quand nous travaillons tous ensemble pour identifier une solution fabuleuse comportant beaucoup de choses accessoires que nous aimons tous.

Je sais ce que vous pensez. *Pourquoi serait-ce un problème ?*

Le problème survient généralement quand nous nous efforçons que tout le monde soit content, et que dans le même temps, nous ne parvenons pas à nous concentrer sur le petit outcome cible spécifique. Le résultat est une solution qui est beaucoup plus grande qu'elle ne devrait l'être.

N'oubliez pas que notre objectif est de minimiser la quantité à développer (notre output) et de maximiser les bénéfices que nous en retirons (l'outcome et l'impact).

© Dunod – Toute reproduction non autorisée est un délit.

Votre opportunité va se décomposer en de nombreuses stories potentielles plus petites que nous n'allons certainement pas toutes développer. Sinon, cela ne servirait à rien de minimiser l'output, n'est-ce pas ?

> Si vous n'abandonnez pas plus d'idées que vous n'en gardez, vous n'accomplissez probablement pas le travail de découverte correctement.

Il y a toujours trop

J'en suis désolé, mais c'est vraiment le cas. Si vous êtes dans le développement de logiciels depuis un certain temps, vous savez déjà probablement cela. J'ai essayé pendant plus d'une dizaine d'années de faire semblant que ce n'était pas vrai, puis j'y ai renoncé et vous devriez en faire autant. Mais ne vous inquiétez pas : vous avez quelques outils qui vous aideront à identifier ce qu'il est possible de développer en fonction du temps et des ressources humaines dont vous disposez.

Au chapitre 3, vous avez étudié la démarche de Globo.com qui utilise une carte pour faire face à une échéance serrée. L'entreprise s'est focalisée sur cette date limite, a pris le problème à bras-le-corps et a identifié un outcome qui lui a permis de réussir. Puis elle a découpé son backlog, c'est-à-dire qu'elle a abandonné toutes les stories qui n'étaient pas nécessaires pour atteindre cet outcome. Il s'agissait de son hypothèse de solution viable minimum. Et il ne s'agissait pas d'une version médiocre, ni faite ni à faire de la grande idée, mais au contraire d'une version vraiment géniale qui était prête pour les élections brésiliennes. Les gens de Globo pensaient que ce serait un atout précieux pour leur entreprise, leurs annonceurs, les réseaux de télévision et leurs utilisateurs. Ils avaient pensé à tous leurs utilisateurs et ils espéraient avoir une solution utilisable. Et en découpant en tranches leur carte, ils ont trouvé une solution qu'il a été possible de développer compte tenu du temps et des équipes qu'ils avaient. L'intersection du valable, de l'utilisable et du faisable d'un ensemble précis de clients, d'utilisateurs et d'utilisations constitue une solution viable.

> *Viable* signifie un succès pour une stratégie spécifique d'entreprise, des clients ciblés et des utilisateurs.

Nous pouvons penser au-delà de cette première release viable à la deuxième, puis aussi à la troisième. Mais nous savons qu'une fois que cette première release est publiée, le monde aura changé, ce qui est une bonne chose. Mais cela signifie aussi que nous aurons besoin de repenser les releases suivantes à la lumière du nouveau monde que nous avons créé.

Le secret de la hiérarchisation des priorités

Rapprochez-vous parce que je vais vous chuchoter ce secret…

Peu de gens savent cela, ou tout du moins ils se comportent comme s'ils ne le savaient pas, mais peut-être jouent-ils les imbéciles simplement pour perturber les gens.

© Dunod – Toute reproduction non autorisée est un délit.

Si vous développez en agile depuis un certain temps vous avez peut-être entendu l'expression « priorisation des stories par leur valeur métier ». L'énoncé est vrai, en quelque sorte, mais quand vous voyez les mots « valeur métier», vous êtes censé placer derrière ces mots quelque chose de spécifique. C'est là où votre équipe de découverte a besoin de dire précisément de quelle valeur il s'agit.

Penchons-nous à nouveau sur Mad Mimi. Gary avait besoin de trouver au plus vite un produit qui serait viable dans un marché spécifique, avant qu'il ne soit à court d'argent. Pour Gary, *viable* signifiait qu'il trouverait un public qui apprécierait son produit et serait prêt à payer pour l'acquérir. Et il pourrait commencer à augmenter l'audience de son produit et, par conséquent, les revenus qu'il en obtiendrait.

C'est cet objectif commercial combiné avec des contraintes financières qui ont obligé Gary à se concentrer sur des utilisateurs précis et leurs activités spécifiques qu'il prendrait en charge. Gary avait encore de grands espoirs de créer une interface commerciale de l'industrie musicale, mais il décida de se concentrer d'abord sur le manager de groupes assurant la promotion de ses clients en envoyant des courriels publicitaires aux fans. Après avoir réalisé cela, la fonctionnalité spécifique sur laquelle il devait se concentrer devint évidente.

Si vous m'avez écouté attentivement, vous avez appris le secret de la hiérarchisation des priorités.

L'avantage métier spécifique met l'accent sur des utilisateurs spécifiques, leurs objectifs et les activités dans lesquelles ils vont s'engager avec votre produit. En mettant l'accent sur les activités, cela permet de se concentrer sur les caractéristiques et les fonctionnalités que les utilisateurs doivent exploiter avec succès.

Pour Mad Mimi, Gary a pris la décision délibérée de se concentrer sur la satisfaction des managers de groupes assurant la promotion de leurs clients. *Il a choisi de se concentrer sur la valeur spécifique*. Il n'a pas utilisé ce terme ambigu de « valeur commerciale ». Il a rempli ce vide avec ce qui avait de la valeur à ses yeux.

L'erreur de hiérarchisation des priorités qui font de la plupart des gens consiste à essayer de hiérarchiser d'abord les features.

> Fixez les priorités des objectifs métiers spécifiques, des clients et des utilisateurs, puis leurs buts. Avant de prioriser les features, priorisez les impacts.

La prochaine fois que vous entendrez quelqu'un demander quelle feature a la priorité la plus élevée sans discuter des objectifs commerciaux, des utilisateurs cibles et de leur utilisation, c'est le signal pour commencer à poser des questions. Essayez de ne pas être trop prétentieux car tout le monde ne connaît pas ce secret.

14.3 ACTIVITÉS DE DÉCOUVERTE, DISCUSSIONS ET ARTEFACTS

Il y a beaucoup d'activités et d'artefacts que votre équipe peut créer au cours de la découverte. Voici un rapide tableau qui va vous donner quelques points de départ pour le genre de choses que vous pouvez faire, mais ne faites pas toutes ces choses parce qu'il y en a probablement trop. Et ne vous contentez pas de ces choses car il existe sans doute des pratiques équivalentes, voire meilleures, qui conviennent mieux à vos compétences et au contexte. En tous les cas, il est hors de question de s'asseoir dans une pièce et d'écrire beaucoup de stories car ce serait totalement déraisonnable.

Formuler l'idée

Utilisez ces discussions pour étudier pourquoi votre entreprise développe le logiciel, pour quels clients et pour savoir comment vous allez mesurer le succès du projet.	– Liste des problèmes commerciaux traités – Indicateurs commerciaux impactés – Courtes listes de clients et d'utilisateurs particuliers – Indicateurs permettant de mesurer si les gens utilisent et apprécient cette nouvelle feature – Risques importants et hypothèses – Discussions avec les parties prenantes et des experts du domaine

Comprendre les clients et les utilisateurs

Utilisez des discussions et faites des recherches pour comprendre les clients et les utilisateurs, leurs besoins, et leur manière actuelle de travailler.	– Listes des rôles et descriptions des utilisateurs – Croquis de profils utilisateurs simples ou de personnes – Profils d'organisation simples – Story maps sur la manière dont les gens font les choses actuellement (cartes des trajets) – Recherches sur l'utilisateur et observation pour combler nos lacunes

Envisager des solutions

Mettez l'accent sur des clients et des utilisateurs particuliers, et envisagez ensuite des solutions qui les aideront. Visualisez les solutions en mots et en images. Validez les solutions avec les clients et les utilisateurs.	– Story maps – Cas d'utilisation et scénarios utilisateur – Maquette de l'interface utilisateur et storyboards – Prototypes de l'interface utilisateur – Croquis de conception architecturale et technique – Prototypes d'architecture ou techniques – Beaucoup de collaboration avec les membres de l'équipe, les utilisateurs, les clients, les parties prenantes et les experts du domaine

Minimiser et planifier

Identifiez ce que vous croyez être une solution viable minimum. Faites une assez bonne estimation pour définir un budget qui permette la livraison de la solution. Créez un plan de développement qui minimise les risques.	– Story maps pour décomposer – Estimation utilisée pour définir un budget de développement

14.4 LA DÉCOUVERTE SERT À CE QUE TOUT LE MONDE SE COMPRENNE

Vous souvenez-vous d'avoir travaillé sur un projet de logiciel où personne ne comprenait très bien la vision d'ensemble ? Vous souvenez-vous du moment où l'équipe a appris à mi-parcours du développement qu'une grande partie du travail n'avait pas été prévue ? Quand ce genre de choses m'est arrivé dans le passé, nous avons souvent trouvé des réponses au sein de l'équipe et avec les collaborateurs externes avec lesquels nous travaillions. Nous aurions facilement pu prévoir les problèmes que nous avions rencontrés si nous avions simplement été sur la même longueur d'onde.

L'histoire de Gary au chapitre 1 est un peu similaire car Gary et son équipe de développement n'ont pas partagé leur compréhension de la vision d'ensemble. Même Gary, qui avait la vision du produit en tête, n'avait pas une idée claire de la taille et de la complexité de son produit. La visualisation de son produit sous la forme d'une série de modèles simples l'a aidé, ainsi que son équipe, à obtenir la même vision d'ensemble.

Pour certaines choses que vous développez, cela peut suffire que tout le monde soit sur la même longueur d'onde et qu'il y ait consensus sur qui sont vos clients et vos utilisateurs, et sur la vision d'ensemble de la solution que vous avez en tête. Mais je dois vous avertir : l'hypothèse à partir de laquelle vous développez est probablement fausse. Ne vous inquiétez pas : je veux vous entretenir de quelques stratégies qui font merveille dans la pratique de la découverte, et ce sera l'objet du prochain chapitre.

© Dunod – Toute reproduction non autorisée est un délit.

15

Utilisation de la découverte pour l'apprentissage validé

Je vous ai quelque peu induit en erreur.

Certains d'entre vous ont peut-être lu le chapitre précédent, voire plusieurs autres, et vous avez lentement atteint un état critique parce que vous savez ce que j'ai laissé de côté. Croyez bien que j'en suis désolé.

Les histoires que j'ai racontées à propos de MadMimi.com et Globo.com sont toutes les deux incomplètes car en vérité elles utilisent des conversations de découverte pour identifier ce qu'elles croyaient être une solution viable minimum. Mais que ces solutions soient viables ou non dans la réalité n'était qu'une hypothèse. En fait, toute cette démarche n'est qu'une hypothèse tant que nous n'avons pas livré le produit et observé ce qu'en font nos clients et nos utilisateurs. Les premières conversations de découverte, ainsi que les story maps, facilitent la formulation d'une bonne hypothèse de départ. Mais pour ces deux entreprises, cela a marqué le début d'un voyage beaucoup plus long pour véritablement découvrir un produit viable.

Cela m'amène à l'une des plus grandes erreurs que commettent les gens : croire que leur solution viable minimum sera couronnée de succès.

15.1 NOUS AVONS TORT LA PLUPART DU TEMPS

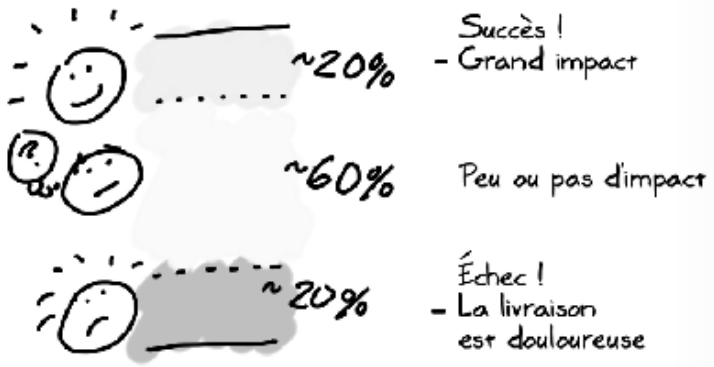
Je suis aussi coupable que la plupart des gens de croire que mes grandes idées seront couronnées de succès. Pour être franc, j'ai livré par le passé beaucoup de solutions dont je pensais qu'elles auraient beaucoup de succès, mais cela n'a pas été le cas. Elles n'étaient pas non plus des échecs lamentables, mais elles n'apportaient pas un grand bénéfice. Dans ces cas-là, mon entreprise et moi-même avons appris à détourner le regard. Je n'étais pas le seul responsable, et nous avions *tous* pensé que les

© Dunod – Toute reproduction non autorisée est un délit.

fonctionnalités que nous ajoutions apportaient une plus-value. Mais en fin de compte, nous avons ajouté telle ou telle fonctionnalité que très peu de gens ont utilisée, la plupart l'ignorant, et nous savions que nous aurions à assurer leur support pendant toute la vie de notre produit.

J'ai une conviction personnelle, qui n'est pas étayée par des recherches scientifiques ni des études, mais que je me suis forgée par l'observation de mes propres échecs et du travail réalisé avec d'autres sociétés : très peu de ce que nous développons est réussi ou fournit l'impact réel que nous espérions. Je dirais qu'environ 20 % sont une réussite. Il y a aussi 20 % des choses que nous développons qui sont de véritables échecs, c'est-à-dire des solutions qui donnent lieu à un impact négatif. J'ai constaté que de nombreux organismes sortaient une nouvelle version améliorée de leur site Web et voyaient leurs ventes diminuer. Ou bien ils publiaient une nouvelle version de leur produit et leurs clients demandaient un retour à l'ancienne version après l'avoir testée. C'est de ce genre d'échec dont je parle.

Mais ce sont les 60 % du milieu qui ne sont ni un succès ni un échec qui constituent un gros problème. Ce sont les choses pour lesquelles nous dépensons beaucoup de temps de développement et de création, et qu'à la fin nous regrettons.



Des recherches du Standish Group, publiées dans les rapports « Chaos », expliquent qu'entre 64 et 75 % des fonctionnalités sont rarement, voire jamais utilisées[1]. Et, selon la source que vous étudiez, de 75 à 90 % des startup développant des logiciels connaissent un échec[2].

Quand on y pense, tout cela est assez décevant et il n'est donc pas étonnant que la plupart des organisations fassent semblant que cela fonctionne.

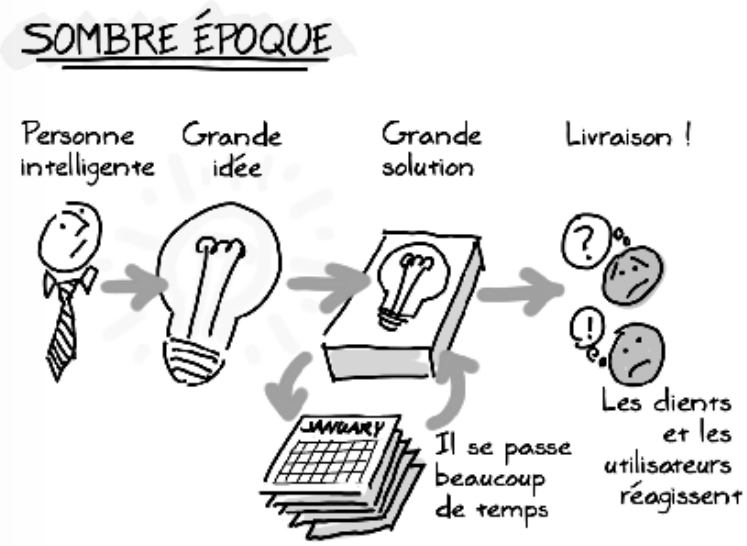
1. Jim Johnson, Président du Standish Group, « ROI, It's Your Job » (keynote), troisième Conférence internationale sur l'Extreme Programming, Alghero (Italie), du 26 au 29 mai 2002.
2. Deborah Gage, « The Venture Capital Secret: 3 Out of 4 Start-Ups Fail », *Wall Street Journal*, 20 septembre 2012, http://on.wsj.com/UtgMZI

© Dunod – Toute reproduction non autorisée est un délit.

15.2 LA SOMBRE EPOQUE

Dans une sombre époque, j'avais pris l'habitude de travailler un peu comme ça. J'arrivais avec une idée géniale, ou en réalité, quelqu'un d'autre, comme mon P.-D.G. ou un client important, me confiait *sa* bonne idée. Je me mettais à l'ouvrage pour concrétiser cette idée et mon équipe et moi-même nous la développions. Cela prenait toujours deux fois plus de temps que prévu, et c'est un problème que nous traiterons dans les prochains chapitres. On livrait le produit, puis on faisait la fête, et parfois, on faisait la fête avant la livraison, mais quoi qu'il en soit, nous en avions terminé avec le produit.

Malheureusement, les problèmes arrivaient très vite. Habituellement, les clients se plaignaient car ce que nous avions livré ne fonctionnait pas de la façon escomptée. Parfois, ils ne se plaignaient pas du tout et on découvrait plus tard que c'était parce que personne n'utilisait vraiment le produit. On passait alors beaucoup de temps à faire semblant de croire que nous avions remporté un grand succès. Pour certains d'entre vous, cela peut décrire la façon dont votre entreprise fonctionne aujourd'hui, et je vais être honnête avec vous : il m'arrive encore souvent de retomber dans ces travers. Mais ne le dites à personne car je suis censé être un expert...



Il existe cependant de meilleures solutions.

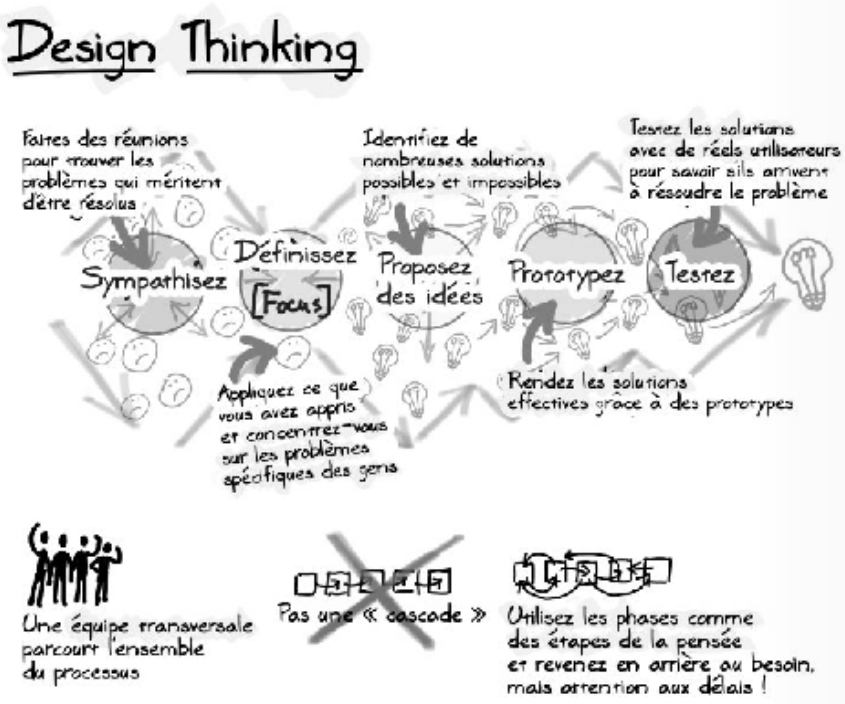
15.3 *EMPATHIZE, FOCUS, IDEATE, PROTOTYPE, TEST*

Il y a plusieurs années, un client potentiel m'a contacté et m'a demandé si je pouvais l'aider à adopter une méthode appelée *Design Thinking*. Ce client employait un

processus agile classique et il le faisait vraiment bien, en ce sens où il livrait ce qui était prévu et avec un bon niveau de qualité. Mais il avait appris que « plus vite vous livrez de la merde, plus il vous en reviendra en retour ». En termes plus élégants, le client avait appris qu'il y avait peu de corrélations entre la quantité de logiciel développé et l'effet obtenu.

Lorsque ce client m'a appelé, j'étais reconnu en tant que spécialiste de la conception de l'expérience utilisateur et du développement agile. Je me suis dit : « *je suis un designer et je pense, je dois donc être capable de faire du Design Thinking* ». Mais je me trompais et ce n'était pas ce que voulait dire le client. Heureusement pour moi, je n'ai pas dit ce que je pensais à voix haute.

Design Thinking est une méthode de travail décrite pour la première fois par une société appelée IDEO et qui a été reprise plus tard et enseignée par la d.school de l'Université de Stanford. De nos jours, cette méthode est enseignée dans plusieurs universités et utilisée dans de nombreuses sociétés dans le monde entier.



Un processus de Design Thinking comporte plusieurs étapes qui, comme je l'explique, paraissent à l'évidence de bonnes idées. Mais, en pratique, la plupart des gens ont tendance à faire exactement le contraire. Il n'y a donc rien d'étonnant à ce que les choses se passent mal aussi souvent.

La première étape d'une approche de Design Thinking se nomme *empathize* (faire preuve d'empathie, s'identifier à quelqu'un). De la part d'un processus de conception, on s'attendrait plutôt à ce que cette étape s'appelle recherche, mais *empathize* signifie que l'on doit comprendre réellement ce que cela fait d'être un utilisateur de votre produit. Pour ce faire, vous devez vous rendre là où se trouvent les utilisateurs, les rencontrer, les regarder travailler et idéalement travailler à leurs côtés. Bien entendu, si vous développez des logiciels pour des chirurgiens, personne ne s'attend à ce que vous deveniez un apprenti chirurgien, mais vous devez faire de votre mieux pour comprendre leur manière de travailler. Il est important de se rappeler qu'en dehors de la recherche traditionnelle, en particulier avec les approches quantitatives passives, on obtient des données et pas toujours de l'empathie.

> Parlez directement aux clients et aux utilisateurs. Aidez-les à relever des défis en y participant vous-même.

L'étape suivante s'appelle *define* (définir). Avec le travail réalisé au cours de l'étape *empathize*, nous en apprenons beaucoup, mais nous devons donner un sens à tout cela afin de bâtir une compréhension partagée. Pour ce faire, on pratique beaucoup la collaboration pour décrire des stories, partager et distiller ce que nous avons appris. Ensuite, on choisit des personnes et des problèmes spécifiques sur lesquels on se concentre.

Utilisez ici des story maps pour cartographier la manière dont les gens travaillent, en incluant les détails de ce que vous avez vu et appris. Mettez l'accent sur les problèmes que rencontrent les utilisateurs et les gains qu'ils recherchent. Utilisez des personas simples pour créer un utilisateur type qui synthétise bien ce que vous avez appris. Choisissez des problèmes spécifiques sur lesquels vous vous focaliserez.

> Ne vous concentrez vraiment que sur un seul problème ou une quantité réduite de problèmes et décrivez-les spécifiquement.

L'étape suivante est *l'idéation*[1]. Si vous avez été attentif dans le dernier chapitre, nous avons évoqué une pratique simple appelée design studio. C'est un exemple d'une bonne approche de l'idéation. Dans la pratique courante des entreprises où la première personne qui arrive avec une idée viable est considérée comme un *winner*, cela semble une perte de temps de prendre en compte un grand nombre d'idées possibles.

J'aime utiliser une story map comme toile de fond de l'idéation. Utilisez une carte qui montre les problèmes, les gains et les autres informations sur les utilisateurs, puis incorporez toutes vos idées sous la forme d'un brainstorming. Écrivez directement les idées de solutions sur des fiches ou des Post-it® pour les injecter sur la carte où la solution est la plus pertinente.

1. NdT : L'idéation représente, d'un point de vue psychologique, la formation et l'enchaînement des idées.

© Dunod – Toute reproduction non autorisée est un délit.

Proposez, de manière délibérée, plusieurs solutions possibles aux problèmes des clients et des utilisateurs.

L'étape suivante consiste à *prototyper*. Aujourd'hui, nous savons probablement tous ce qu'est un prototype, mais nous négligeons souvent d'en créer dans la précipitation du développement des produits. C'est vraiment dommage car de petits investissements sous forme de simples prototypes papier nous aident à réfléchir à notre solution en nous y confrontant directement. En construisant des prototypes simples sur papier, ou des outils simples de prototypage, cela nous permet d'abandonner beaucoup d'idées qui ne fonctionnent pas. En commençant à simuler l'acte même d'utiliser un produit, cela nous aide à continuer à pratiquer l'idéation afin de trouver des idées pour améliorer encore la solution.

Créez des prototypes simples pour explorer vos meilleures solutions. Faites évoluer les prototypes à un niveau de fidélité qui permette aux utilisateurs et aux clients de savoir si la solution résout vraiment leur problème.

La dernière étape consiste à *tester*. Cela ne signifie pas que vous devez vérifier s'il y a des bugs, mais que vous devez savoir si votre solution résout vraiment le problème de quelqu'un. Vous pourriez être surpris d'apprendre qu'une solution peut résoudre un problème, même si elle comporte des bugs. Quand vous avez un prototype dont vous pensez qu'il résout les problèmes sur lesquels vous avez choisi de mettre l'accent, faites tester ce prototype aux gens qui vont utiliser votre produit. Il ne s'agit pas d'en faire une démonstration ni de le vendre, mais les utilisateurs potentiels doivent certifier que le prototype qu'ils pourraient utiliser peut résoudre l'un de leurs problèmes. Ils ont donc besoin de l'utiliser pour accomplir une tâche réelle.

Présentez vos solutions aux gens qui vont acheter ou utiliser votre produit. Ne vous attendez pas à un succès immédiat. Procédez par itération pour améliorer votre produit.

En plus de ces cinq étapes, l'approche par Design Thinking est une façon de travailler qui met l'accent sur les petites équipes pluridisciplinaires qui collaborent rapidement en utilisant des modèles simples, des esquisses et des méthodes sommaires de documentation et de communication. Cela devrait vous faire penser à l'équipe de découverte et aux autres collaborateurs que j'ai décrits au chapitre 12. Vous devriez reconnaître leur façon de travailler qui met l'accent sur la compréhension mutuelle.

En utilisant des éléments du Design Thinking, cela contribue à nous faire vraiment prendre conscience des problèmes que nous allons résoudre, et non pas à régler des problèmes imaginaires. Le prototypage et le test des solutions avant de se lancer dans de gros investissements de développement contribuent à valider que nous créons des solutions que les gens pourront vraiment utiliser.

Mais le Design Thinking, quand on ne pratique que cette seule approche, peut engendrer des problèmes.

© Dunod – Toute reproduction non autorisée est un délit.

15.4 COMMENT GÂCHER LES BONNES CHOSES

Les processus de conception existent depuis longtemps, ainsi que le Design Thinking, en tant qu'approche généraliste de conception. Et un processus de conception mené à bien peut être une amélioration considérable par rapport aux périodes sombres. Mais ne confondez pas un processus avec une compétence. On peut de façon prévisible faire échouer un processus de conception. Si vous avez vu un bon processus de conception prendre beaucoup trop de temps et produire de mauvais résultats, alors vous vous êtes peut-être dit que ce genre de processus ne fonctionnait pas, mais en fait il ne s'agit pas du processus.

Voici quelques bonnes façons de gâcher un processus de conception :

- Démarrez sans bien formuler les besoins de l'entreprise et des clients cibles. Cela va rendre difficile la hiérarchisation des priorités et la détermination d'une bonne solution.
- Passez beaucoup de temps à faire des recherches approfondies et à donner du sens à ce que vous avez appris. Vous ne manquerez jamais de choses à apprendre, alors pourquoi vous arrêter ? Vous imposer des délais aurait été une bonne idée.
- Ne passez pas de temps à parler aux gens ni à apprendre des choses sur eux. Après tout, nous avons beaucoup d'informations, et nos idées de solution sont vraiment excellentes. Nous avons juste besoin de nous mettre à leur conception.
- Ne vous concentrez pas sur le problème spécifique et tentez de résoudre de nombreux problèmes pour un maximum de personnes. Plus vous allez résoudre de problèmes, mieux ce sera, n'est-ce pas ? Sauf que les grands problèmes se traduisent souvent par de grandes solutions. Et tenter de résoudre un problème pour des personnes ayant des besoins contradictoires peut se traduire par une solution qui ne satisfait personne.

- Envisagez plusieurs solutions, mais ne demandez qu'aux véritables designers de contribuer aux idées de solution, parce que ce sont les seuls à être formés pour avoir de bonnes idées.
- Ne perdez pas de temps à prendre en compte plusieurs solutions, parce que c'est votre idée qui est la bonne.
- Créez un prototype magnifique qui a l'air tout à fait réel, mais qui ne fonctionne pas assez bien pour que les clients et les utilisateurs puissent vraiment l'utiliser. Après tout, quand ils le voient, ils nous disent qu'il a l'air parfait.
- Arrivez à vous convaincre, ainsi que les autres, que cette solution bien documentée et conçue de manière professionnelle va fonctionner correctement. Après tout, vous avez suivi un processus de conception rigoureux, alors qu'est-ce qui pourrait mal se passer ?
- Ne vous souciez pas des coûts de développement. C'est la bonne solution et peu importe le prix car il est justifié.
- Lorsque vous livrez la solution aux clients et aux utilisateurs et que vous ne constatez pas les résultats attendus, trouvez le défaut dans le processus. Ou mieux encore, trouvez une personne ou un groupe que vous pourrez tenir pour responsable.

Je sais, je suis ici un peu d'humeur chagrine, mais je suis pourtant un fervent partisan de l'utilisation des processus de conception. Et, curieusement, je trouve que je suis souvent celui qui s'en plaint. Je peux aussi confesser que je me suis rendu coupable de presque toutes les erreurs énumérées ci-dessus. Mais, au cours des dernières années, j'ai trouvé une astuce qui améliore les méthodes de conception classiques.

15.5 COURTES BOUCLES D'APPRENTISSAGE VALIDÉ

Eric Ries est l'auteur d'un livre intitulé *The Lean Startup* (Crown Business). Dans son livre, Eric décrit comment il est tombé dans le piège que j'ai décrit précédemment comme étant la « sombre époque ». En tant que directeur technique d'une startup, il a aidé son entreprise à développer ce qu'ils croyaient être un produit à succès. Mais leur clientèle cible et les utilisateurs ne l'ont pas vu de cette manière-là et ils ont en fait réagi pour la plupart avec un mélange de commentaires agréables, de critiques et d'apathie pure et simple. Ce n'était certainement pas l'effet qu'ils recherchaient.

Steve Blank était l'un des consultants de l'entreprise d'Eric et il avait écrit un livre intitulé *The Four Steps to the Epiphany* (K & S Ranch) dans lequel il affirme que la première chose que vous devez développer n'est pas un produit, mais des clients. Il décrit un processus de validation progressive du fait que vous avez trouvé des clients qui sont intéressés par une solution, pour ensuite valider que les solutions que vous avez en tête sont les solutions qu'ils vont acheter, utiliser et promouvoir. Blank appelle cela un *processus d'apprentissage validé*.

La plus grande contribution d'Eric Ries au développement de produits est la simplification et la traduction sous forme de produits de cette pensée qui est résumée dans ce mantra simple : *développer-mesurer-apprendre*. Eric a mis l'accent sur la réduction du temps nécessaire pour parcourir ce cycle d'apprentissage simple. L'un des plus grands défauts des processus de conception traditionnels est qu'ils prennent beaucoup de temps pour l'apprentissage et la conception, tellement de temps que vous devenez prisonniers de ces solutions et que vous n'arrivez pas à valider que ces solutions apportent vraiment les bénéfices escomptés. Alors qu'un processus de conception classique peut prendre des semaines ou des mois pour valider une idée de solution, un processus Lean Startup prend en général à peine quelques jours.

Ce que recouvre un nom

Je dois vous dire que j'aime la plupart des éléments de la méthode Lean Startup, mais que je n'apprécie pas son nom. Elle n'est pas aussi mince[1] que cela et les concepts sont beaucoup trop importants pour être utilisés simplement par des startup.

Lean fait référence à l'utilisation du *Lean Thinking* et aux principes décrits il y a quelques décennies par Toyota ; Le *Lean Thinking* est aujourd'hui utilisé dans beaucoup d'autres contextes, y compris le développement de logiciels. Il y a énormément de bonnes idées dans le *Lean Thinking* et Lean Startup reste à la surface des choses.

Eric essaye d'argumenter que les startup existent même au sein des plus grandes entreprises, où il y a des contextes de risque élevé et d'incertitude qui incitent à adopter un mode de gestion de startup. Mais j'ai la conviction que si vous vous dites qu'il n'y a pas beaucoup de risques ou d'incertitude dans un projet, vous allez tout droit à l'échec. Il y a toujours une certaine part de risque, et les stratégies d'apprentissage décrites dans un processus Lean Startup sont très utiles dans la plupart des contextes. Il n'y a pas besoin d'essayer de se justifier pour avoir besoin de se comporter comme une startup.

15.6 COMMENT LEAN STARTUP MODIFIE LA CONCEPTION DE PRODUIT

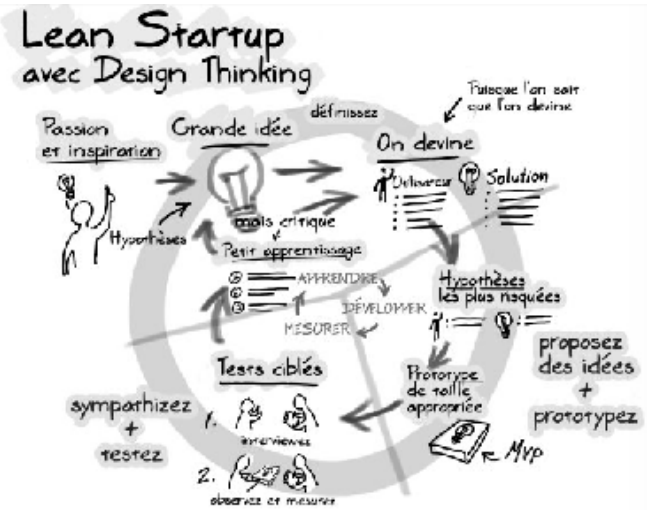
À la sombre époque, nous avions de grandes idées, nous les développions et nous espérions que tout irait pour le mieux.

Si nous avions tenté de sortir de ce piège en utilisant un processus de conception rigoureuse, nous aurions fait de notre mieux pour mettre de côté nos grandes idées et puis nous avons approfondi nos recherches pour comprendre les problèmes que nous allions résoudre.

Voici comment je recommande de faire les choses aujourd'hui avec le Lean Startup.

© Dunod – Toute reproduction non autorisée est un délit.

1. Ndt : en anglais, *lean* signifie mince ou maigre.



15.7 COMMENCEZ PAR DEVINER

Oui, vous avez bien lu : il faut deviner.

À la sombre époque, vous auriez deviné, mais vous auriez fait semblant que ce n'était pas le cas. Dans un processus de conception, vous ne vous seriez pas permis de deviner, mais vous l'auriez quand même fait, alors il faut arrêter de faire semblant que l'on ne devine pas.

En fait, il ne s'agit pas simplement de deviner car c'est un mélange de passion, d'expérience et de perspicacité, avec juste une bonne dose de devinette, qui permet de faire avancer les choses. J'obtiens mes hypothèses et mes suppositions à partir de mes utilisateurs, généralement en dessinant des prototypes simples. Je décris la manière dont je pense qu'ils travaillent aujourd'hui en créant de simples story maps « maintenant ». Je réalise cela de manière collaborative avec d'autres personnes qui ont une réelle expérience avec les utilisateurs et les clients. Et, dans certaines situations, j'implique directement les clients et les utilisateurs. En fait, un grand nombre des suppositions de notre équipe n'en sont pas du tout. Mais ce n'est pas non plus exactement comme le travail de recherche que nous avons l'habitude de faire. Nous allons passer quelques heures ou quelques jours à faire ce genre d'activités, mais jamais des semaines ou des mois.

Après avoir établi une compréhension partagée avec les personnes qui utiliseront le logiciel et s'être concentré sur les problèmes que nous allons résoudre, nous faisons des suppositions sur la solution. Nous allons utiliser les principes du Design Thinking, et envisager délibérément plusieurs solutions. Mais nous allons essayer de converger rapidement vers ce que nous pensons être notre meilleure solution. Parfois, nous ne réussirons pas à choisir une solution unique et nous en choisirons deux. Nous ne nous en préoccupons pas trop, parce que nous savons que de toutes les façons, nous sommes probablement dans l'erreur.

15.8 DONNEZ UN NOM À VOS HYPOTHÈSES RISQUÉES

Étant donné que nous avons fait de nombreuses suppositions sur nos utilisateurs et leurs défis actuels, nous allons donner un nom à ces suppositions. Plus précisément, nous travaillerons ensemble pour établir une liste des choses dont nous pensons qu'elles sont vraies, mais si nous trouvons que ce n'est pas le cas, alors nous aurons besoin de tout repenser.

Nous ferons la même chose avec notre solution. Nous allons réfléchir à la manière dont nous croyons que les utilisateurs l'accueilleront et à la manière dont ils s'en serviront. Nous allons former une hypothèse sur la façon dont nous pensons qu'ils se comporteront avec notre solution. Nous aborderons également les risques techniques qui constitueraient une menace pour la faisabilité de notre solution.

En fonction de la liste des risques et des hypothèses concernant nos clients, nos utilisateurs et notre solution, nous allons pouvoir identifier ce que nous pensons être les risques les plus importants.

15.9 CONCEVEZ ET DÉVELOPPEZ UN PETIT TEST

C'est le moment où les choses deviennent *vraiment* différentes.

À la sombre époque, nous aurions planifié et développé un produit complet. Dans un processus de conception classique, nous aurions prototypé la totalité du produit, ou en tous les cas sa plus grande partie. Mais en utilisant une approche Lean Startup, où notre objectif est d'apprendre quelque chose aussi rapidement que possible, nous ferons notre possible pour créer un prototype le plus petit possible. Dans de nombreux cas, ce sera même difficile de l'appeler un prototype.

Voici l'exemple d'Ithaka, organisation à but non lucratif où certains de mes amis travaillent. Ils ont développé un produit appelé JSTOR et si vous avez fréquenté une université au cours des dix dernières années, vous avez probablement utilisé JSTOR à la BU pour trouver des articles et des livres.

Les étudiants qui se servaient du produit voulaient l'utiliser facilement depuis n'importe où, que ce soit dans un cybercafé, à la maison, ou bien en voyage. Mais accéder à JSTOR en dehors de l'université n'était pas simple pour les étudiants. Il leur fallait définir un nom d'utilisateur et un mot de passe pour pouvoir accéder à distance à toutes les ressources auxquelles leur université s'était abonnée. L'équipe de JSTOR avait déjà une solution, mais elle était un peu difficile à utiliser. Ils ont alors voulu tester une nouvelle façon de faire les choses.

Ils ont émis les hypothèses suivantes sur les étudiants :

- Ils travaillent dans des cybercafés ou dans leur chambre

© Dunod – Toute reproduction non autorisée est un délit.

- Ils ne savent pas qu'ils peuvent accéder à JSTOR à partir de ces emplacements
- Ou, s'ils le savent, ils pensent que c'est difficile

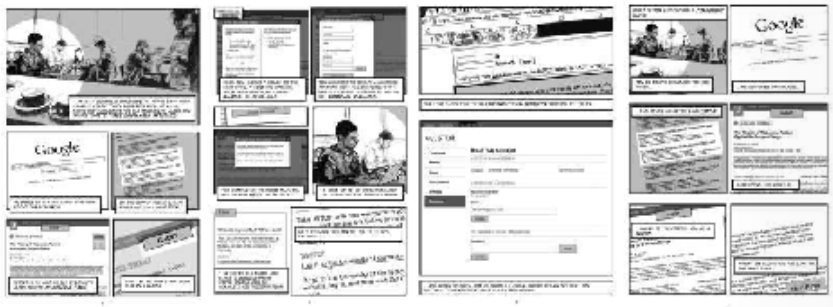
Ils ont émis ces hypothèses sur leur solution :

- La nouvelle solution serait facile à apprendre
- Il serait évident pour les étudiants qu'ils obtiendraient un accès complet à JSTOR sans devoir être à la bibliothèque

Pour tester leurs hypothèses, l'équipe de JSTOR n'a pas eu besoin de développer un logiciel du moins à ce stade. Ils devaient parler aux étudiants pour leur demander où ils se trouvaient précisément lorsqu'ils faisaient des recherches en utilisant JSTOR. L'équipe avait besoin de confirmer que les étudiants rencontraient bien les problèmes qu'elle avait imaginés, et en fonction de cela, que les étudiants pensaient que la solution de JSTOR résoudrait ces problèmes.

L'équipe avait prévu de parler à beaucoup d'étudiants, et pour faciliter la communication avec les étudiants et rendre plus cohérente la description du problème et sa solution, elle a créé une bande dessinée de conception. Si vous n'en avez jamais vu, une bande dessinée de conception est exactement comme son nom l'indique : une bande dessinée de quelques pages, mais, au lieu de représenter des superhéros luttant contre des êtres malfaisants, elle présente de véritables personnes réglant un problème réel avec votre idée de la solution.

Voici quelques pages extraites de la bande dessinée de conception de JSTOR (avec l'aimable autorisation d'Ithaka, © 2014 Ithaka, tous droits réservés) :



Le test que l'équipe a conçu nécessitait de passer un peu de temps à interviewer les étudiants pour connaître les problèmes qu'ils rencontraient alors. Puis ils examineraient le scénario avec les étudiants pour voir si leur solution pourrait résoudre les difficultés. Ils n'ont pas développé un prototype complet. Ils voulaient d'abord s'assurer que leur solution serait vraiment utilisable, et ils ne pouvaient pas apprendre cela à partir d'une bande dessinée. Ils avaient également eu quelques soucis techniques qui ont nécessité la programmation d'un prototype à des fins de test. Mais rien de cela n'avait d'importance si les étudiants n'avaient pas le problème que l'équipe cherchait à résoudre et s'ils ne réagissaient pas bien à la solution proposée.

Cette plus petite solution possible à tester est ce que le Lean Startup appelle un *produit viable minimum*. Eric Ries sait pertinemment que ce n'est pas un produit complet, mais que votre objectif étant l'apprentissage, il s'agit du plus petit produit que l'on puisse développer pour apprendre.

15.10 MESUREZ EN EXÉCUTANT VOTRE TEST AVEC LES CLIENTS ET LES UTILISATEURS

Présentez le test aux clients et aux utilisateurs. Au cours de la phase de travail initiale, cela signifie habituellement que l'on planifie des entrevues et que l'on passe du temps avec les utilisateurs. Si vous créez une solution grand public, vous pouvez vous rendre à l'endroit où sont vos clients et les utilisateurs, les arrêter et les interviewer. J'ai ainsi accompagné des gens avec qui je travaille dans des centres commerciaux, des cafés-restaurants et des attractions touristiques.

JSTOR a recruté des étudiants et a passé entre 30 et 60 minutes à les interviewer pour apprendre comment ils travaillaient et pour confirmer que ses suppositions correspondaient bien aux problèmes à résoudre. Puis ils ont fait lire aux étudiants la bande dessinée de conception pour obtenir leur réponse à l'idée de la solution.

15.11 REPENSEZ VOTRE SOLUTION ET VOS HYPOTHÈSES

Après avoir exécuté votre test plusieurs fois, vous allez commencer à obtenir des résultats prévisibles. Si vous vous fourvoyez complètement, vous l'apprendrez souvent assez rapidement. Revenez sur ce que vous avez appris et reprenez les faits sur la façon dont vos utilisateurs travaillent actuellement, afin de repenser votre solution. Puis, repensez vos hypothèses sur les utilisateurs et les solutions afin de concevoir votre prochain test.

Après les tests réalisés par JSTOR, l'équipe a appris que certains étudiants n'avaient pas les problèmes qu'elle avait anticipés. Cela aurait normalement dû être de mauvaises nouvelles parce que nous détestons tous nous tromper, mais, dans une démarche Lean Startup, c'est une excellente nouvelle. C'est excellent parce qu'ils ont trouvé qu'ils avaient tort, après seulement quelques jours de réflexion et de travail, et non pas après des semaines de développement d'un logiciel par toute une équipe.

Si vous utilisez ce type d'approche, votre plus grand défi sera de prendre l'habitude de vous réjouir de ce que vous apprenez plutôt que d'avoir peur de vous tromper.

> Dans une démarche Lean Startup, le plus gros échec est souvent d'échouer dans l'apprentissage.

© Dunod – Toute reproduction non autorisée est un délit.

- Ils ne savent pas qu'ils peuvent accéder à JSTOR à partir de ces emplacements
- Ou, s'ils le savent, ils pensent que c'est difficile

Ils ont émis ces hypothèses sur leur solution :

- La nouvelle solution serait facile à apprendre
- Il serait évident pour les étudiants qu'ils obtiendraient un accès complet à JSTOR sans devoir être à la bibliothèque

Pour tester leurs hypothèses, l'équipe de JSTOR n'a pas eu besoin de développer un logiciel du moins à ce stade. Ils devaient parler aux étudiants pour leur demander où ils se trouvaient précisément lorsqu'ils faisaient des recherches en utilisant JSTOR. L'équipe avait besoin de confirmer que les étudiants rencontraient bien les problèmes qu'elle avait imaginés, et en fonction de cela, que les étudiants pensaient que la solution de JSTOR résoudrait ces problèmes.

L'équipe avait prévu de parler à beaucoup d'étudiants, et pour faciliter la communication avec les étudiants et rendre plus cohérente la description du problème et sa solution, elle a créé une bande dessinée de conception. Si vous n'en avez jamais vu, une bande dessinée de conception est exactement comme son nom l'indique : une bande dessinée de quelques pages, mais, au lieu de représenter des superhéros luttant contre des êtres malfaisants, elle présente de véritables personnes réglant un problème réel avec votre idée de la solution.

Voici quelques pages extraites de la bande dessinée de conception de JSTOR (avec l'aimable autorisation d'Ithaka, © 2014 Ithaka, tous droits réservés) :

Le test que l'équipe a conçu nécessitait de passer un peu de temps à interviewer les étudiants pour connaître les problèmes qu'ils rencontraient alors. Puis ils examineraient le scénario avec les étudiants pour voir si leur solution pourrait résoudre les difficultés. Ils n'ont pas développé un prototype complet. Ils voulaient d'abord s'assurer que leur solution serait vraiment utilisable, et ils ne pouvaient pas apprendre cela à partir d'une bande dessinée. Ils avaient également eu quelques soucis techniques qui ont nécessité la programmation d'un prototype à des fins de test. Mais rien de cela n'avait d'importance si les étudiants n'avaient pas le problème que l'équipe cherchait à résoudre et s'ils ne réagissaient pas bien à la solution proposée.

Cette plus petite solution possible à tester est ce que le Lean Startup appelle un *produit viable minimum*. Eric Ries sait pertinemment que ce n'est pas un produit complet, mais que votre objectif étant l'apprentissage, il s'agit du plus petit produit que l'on puisse développer pour apprendre.

15.10 MESUREZ EN EXÉCUTANT VOTRE TEST AVEC LES CLIENTS ET LES UTILISATEURS

Présentez le test aux clients et aux utilisateurs. Au cours de la phase de travail initiale, cela signifie habituellement que l'on planifie des entrevues et que l'on passe du temps avec les utilisateurs. Si vous créez une solution grand public, vous pouvez vous rendre à l'endroit où sont vos clients et les utilisateurs, les arrêter et les interviewer. J'ai ainsi accompagné des gens avec qui je travaille dans des centres commerciaux, des cafés-restaurants et des attractions touristiques.

JSTOR a recruté des étudiants et a passé entre 30 et 60 minutes à les interviewer pour apprendre comment ils travaillaient et pour confirmer que ses suppositions correspondaient bien aux problèmes à résoudre. Puis ils ont fait lire aux étudiants la bande dessinée de conception pour obtenir leur réponse à l'idée de la solution.

15.11 REPENSEZ VOTRE SOLUTION ET VOS HYPOTHÈSES

Après avoir exécuté votre test plusieurs fois, vous allez commencer à obtenir des résultats prévisibles. Si vous vous fourvoyez complètement, vous l'apprendrez souvent assez rapidement. Revenez sur ce que vous avez appris et reprenez les faits sur la façon dont vos utilisateurs travaillent actuellement, afin de repenser votre solution. Puis, repensez vos hypothèses sur les utilisateurs et les solutions afin de concevoir votre prochain test.

Après les tests réalisés par JSTOR, l'équipe a appris que certains étudiants n'avaient pas les problèmes qu'elle avait anticipés. Cela aurait normalement dû être de mauvaises nouvelles parce que nous détestons tous nous tromper, mais, dans une démarche Lean Startup, c'est une excellente nouvelle. C'est excellent parce qu'ils ont trouvé qu'ils avaient tort, après seulement quelques jours de réflexion et de travail, et non pas après des semaines de développement d'un logiciel par toute une équipe.

Si vous utilisez ce type d'approche, votre plus grand défi sera de prendre l'habitude de vous réjouir de ce que vous apprenez plutôt que d'avoir peur de vous tromper.

> Dans une démarche Lean Startup, le plus gros échec est souvent d'échouer dans l'apprentissage.

© Dunod – Toute reproduction non autorisée est un délit.

Dans une démarche Lean Startup, *développer* signifie développer l'expérience la plus petite possible. *Mesurer* signifie que l'on recueille des analyses à partir de logiciels fonctionnels, des observations directes au cours d'entrevues ou des tests de prototypes en tête à tête, ou les deux. *L'apprentissage* est ce que l'on fait avec des informations. C'est la refonte de nos hypothèses et la révision de ce que nous croyons être la future meilleure solution.

15.12 STORIES ET STORY MAPS ?

Vous demandez peut-être où se trouvent les stories et les story maps dans tout cela, et vous avez raison. Tout au long d'une approche d'apprentissage validé, vous allez constamment rédiger des stories sur vos utilisateurs, ce qu'ils font et leurs motivations. Vous utiliserez des story maps pour raconter de plus grandes stories sur la manière dont les gens travaillent, et sur la manière dont vous imaginez qu'ils utiliseront votre solution. Quand viendra le temps de créer des prototypes, vous utiliserez des stories et des conversations de stories pour vous mettre d'accord précisément sur l'aspect du prototype développé, et sur ce que vous devrez vérifier pour dire que le prototype est terminé. Une fois que vous aurez compris que les stories sont une façon de travailler, vous vous rendrez compte qu'il est difficile de dire quand vous les utilisez et quand vous ne les utilisez pas.

Il y a cependant une grande différence dans la façon dont nous utilisons des stories pendant la découverte. Habituellement, lorsque nous les utilisons, nous parlons du logiciel que nous avons l'intention de développer et de mettre en production avec les développeurs, les testeurs et beaucoup d'autres personnes. Nous travaillons très dur pour nous assurer que nous avons acquis une compréhension partagée. Nous rentrons dans les détails pour estimer le temps de développement. Habituellement, nous parlons de plusieurs stories pour nous mettre d'accord sur ce que nous pouvons développer dans un sprint de deux semaines ou une itération. Mais la façon dont nous travaillons au cours de la découverte est plus rapide car nous espérons construire des prototypes simples en quelques heures, et non pas en quelques jours. Et nous espérons que même les prototypes que nous développons en les codant avec des données réelles ne prendront que quelques jours, et pas des semaines. Nous développons pour apprendre, et nous nous attendons à ce que la plupart de nos idées échouent, ou au minimum, aient besoin de quelques ajustements pour réussir. Nous nous concentrons donc sur une collaboration permettant un consensus rapide avec un minimum de formalités.

Au cours de la découverte et de l'apprentissage validé, vous pouvez rédiger des stories constamment, décomposer vos idées et votre travail en petits morceaux constructibles pour convenir exactement de ce qu'il faut développer. Vous allez travailler si vite que vous ne vous rendrez même pas compte que vous utilisez des stories, mais ce sera pourtant le cas.

© Dunod – Toute reproduction non autorisée est un délit.

16

Affinez, définissez et développez

Alors, que fait-on maintenant ? Si les stories servent uniquement à organiser et à faciliter les discussions dans le but de produire du logiciel, notre activité va sembler se résumer à d'interminables conversations.

16.1 DES FICHES, DES CONVERSATIONS, ENCORE DES FICHES, ENCORE DES CONVERSATIONS...

La première série de conversations vous a aidé à comprendre une opportunité. Vous avez parlé des utilisateurs de votre produit et imaginé comment ils s'en serviraient pour accomplir quelque chose d'intéressant. Vous avez approfondi suffisamment vos conversations pour décomposer la grande opportunité en stories assez petites pour que vous puissiez dire lesquelles doivent figurer dans la prochaine release du produit, et lesquelles sont finalement moins importantes. Vous avez collecté ces stories dans un backlog de release.

Si vous êtes intelligent, et je sais que vous l'êtes, vos conversations ont détaillé l'interface du logiciel, son comportement et son intégration dans l'architecture logicielle existante. Lors de ces conversations, vous avez été attentif aux risques. Vous avez découpé les stories en tranches que vous pourrez développer afin d'en apprendre plus sur le produit, et plus rapidement. Et comme vous êtes intelligent, vous avez segmenté votre backlog de release pour commencer avec des stories d'apprentissage, poursuivre avec celles de développement, et terminer par les améliorations.

Mais maintenant que tout cela est bien concret, il est temps d'avoir nos dernières *bonnes* conversations.

16.2 DÉCOUPAGE ET POLISSAGE

Nous voulons commencer à développer certains éléments, et nous savons que le développement du logiciel décrit par les stories se déroulera en douceur et de manière prévisible à la condition de décrire de manière concise et précise ce que nous voulons créer. Mais, après toutes ces conversations, les stories dont nous disposons sont un peu brutes de décoffrage. Nous n'en avons probablement pas parlé suffisamment en détail pour comprendre précisément ce qu'elles sont et ce qu'elles ne sont pas, et savoir vraiment combien de temps elles prendront à être développées. Mais nous avons une machine magique qui va arranger tout cela.

Je propose que vous imaginiez une petite machine au design élégant. Nous allons déposer depuis notre backlog de release des stories brutes dans un grand entonnoir situé sur le côté gauche. Ensuite nous entendrons 'quelques bruits de grincements et de cliquetis à l'intérieur de la machine. Et d'un petit goulot situé sur le côté droit sortiront des petites pépites bien polies. Ces pépites sont ce que les membres de l'équipe peuvent ramasser et utiliser de manière prévisible pour développer du logiciel de qualité.

De l'extérieur, cette machine semble magique, mais à l'intérieur, votre équipe a des discussions sérieuses sur le découpage et le polissage des stories. Le mécanisme spécial et secret qui se cache à l'intérieur de cette machine s'appelle un *atelier de stories*.

Comme vous vous en souvenez peut-être à la lecture du chapitre 11, les ateliers de stories sont de petites conversations productives où les bonnes personnes travaillent ensemble pour décrire les stories une dernière fois et pour décider de ce qu'ils vont finalement choisir de développer. Ce sont des conversations approfondies de stories qui donnent lieu à des confirmations. Nous obtenons finalement notre troisième C du flux *carte*[1]*-conversation-confirmation*. Et c'est ce C-*là* qui nous aide vraiment à décomposer et à polir les pépites.

16.3 ANIMATION DES ATELIERS DE STORIES

Vous aurez besoin d'un petit groupe qui inclut un développeur, un testeur et des gens qui comprennent les utilisateurs, ainsi que l'aspect et le comportement de l'interface utilisateur (des concepteurs d'interface utilisateur ou des analystes métiers dans certaines organisations). Il est préférable que le groupe soit assez petit (de trois à cinq personnes) pour travailler efficacement avec un tableau blanc.

1. Ndt : comme nous l'avons déjà indiqué au chapitre 6, nous traduisons habituellement le mot anglais *card* par fiche pour éviter la confusion avec le terme anglais *map* que nous traduisons par carte. Cependant, pour garder les 3 C, nous avons choisi, dans ce contexte, de traduire *card* par carte, mais il s'agit bien d'une fiche.

© Dunod – Toute reproduction non autorisée est un délit.

Il s'agit d'un atelier, et non pas d'une réunion. *Réunion* est le terme qui est devenu un euphémisme pour désigner une collaboration improductive. Un atelier de stories doit comporter de nombreuses discussions productives, des gens qui réagissent, des dessins et des croquis sur un tableau blanc. Il est important de travailler ensemble pour décider exactement ce qui sera développé. Nous devons sortir de cette conversation avec une solide compréhension mutuelle, et nous aurons besoin d'espace pour mener à bien ces conversations en mots et en images.

Dans toutes les conversations précédentes, nous avons évoqué les détails, mais nous nous sommes contentés du niveau de profondeur nécessaire pour prendre les décisions du moment. Les décisions qu'il faut prendre à présent se concentreront sur la réponse à la question suivante : qu'allons-nous exactement développer ?

C'est au cours de cette conversation que vous allez découvrir que votre story est trop grande. J'entends par là qu'elle est plus grande que la taille idéale que nous souhaiterions obtenir (quelques jours de développement, voire moins). D'accord, elle n'est pas *toujours* trop grande, mais si vous supposez qu'elle le sera, alors vous serez agréablement surpris quand ce ne sera pas le cas. Heureusement, vous disposez dans la salle des bonnes personnes pour vous aider à décomposer cette story en stories plus petites qui pourront être livrées, testées et présentées dans le produit que l'équipe cherche à développer.

Recette de l'atelier de stories

Un atelier de stories permet d'affiner la compréhension et de définir plus précisément le travail de l'équipe de développement. L'atelier est une conversation à propos du produit, agrémentée de nombreuses photos et données, qui aide l'équipe à prendre des décisions et à aboutir à une confirmation : les critères d'acceptation de ce que l'on choisira de développer.

En préambule à l'atelier, informez l'équipe des stories qui feront l'objet de l'atelier. Affichez-les sur un mur, ou diffusez-les d'une autre manière. Laissez aux membres de l'équipe le choix de participer à l'atelier.

Limitez le nombre de participants à l'atelier pour rester productif. De trois à cinq personnes suffisent.

Invitez les bonnes personnes. Pour que cette conversation soit efficace, invitez :

– quelqu'un qui comprend les utilisateurs et le fonctionnement de l'interface (souvent le PO, un professionnel de l'utilisateur expérience ou un analyste métier) ;

– un ou deux développeurs qui connaissent le code source auquel vous allez ajouter vos stories, parce qu'ils vont mieux comprennent ce qu'il est possible de développer ;

– un testeur qui aidera à tester le produit, parce qu'il pourra poser les questions délicates afin que vous preniez en considération les aléas que d'autres personnes, souvent trop optimistes, n'envisagent pas.

Vous pouvez ici impliquer d'autres personnes et d'autres rôles, mais n'oubliez pas que la bonne taille pour une bonne conversation est le nombre idéal de convives pour un dîner.

Vous pouvez constater qu'une personne peut avoir deux casquettes. Par exemple, je vois souvent la combinaison analyste métier-testeur dans certains services informatiques. Mais, si toutes les problématiques ne sont pas prises en compte, faites une pause dans l'atelier et essayez de trouver une personne de l'équipe qui puisse s'occuper des problèmes non couverts.

Prenez en considération toutes les options. Utilisez les conversations pour approfondir les questions suivantes :

– qui est précisément l'utilisateur ?

– comment pensons-nous exactement qu'il va utiliser le produit ?

– quel sera l'aspect précis de l'interface utilisateur ?

– comment le logiciel va-t-il exactement se comporter dans cette interface utilisateur (règles métier et validation des données) ?

– comment pourrions-nous, dans les grandes lignes, développer le logiciel ? Car nous avons besoin de prédire le temps nécessaire au développement, et comme heureusement, nous faisons à ce stade des choses qui se rapprochent de la réalité, nous pouvons produire une estimation plus précise.

N'oubliez pas que nous ne devons prendre en considération que ce qui est absolument nécessaire. Si la discussion débouche sur des solutions qui sont coûteuses ou complexes, prenez du recul et discutez des vrais problèmes à résoudre et des alternatives envisageables.

Mettez-vous d'accord sur ce qu'il faut développer. Après avoir passé suffisamment de temps en conversations pour s'être bien compris, il faut répondre aux questions suivantes :

– Qu'allons-nous vérifier pour nous 'assurer que le logiciel est terminé ?

– Comment allons-nous faire une démonstration du logiciel lors de la recette lorsque nous l'analyserons ensemble ?

Parlez et documentez. Utilisez des tableaux blancs ou des tableaux papier pour dessiner, prendre des exemples et envisager des options. Ne laissez pas vos décisions s'envoler. Enregistrez-les sur un tableau où tout le monde pourra les voir. Photographiez les notes et les dessins pour les transcrire plus tard.

Prenez des exemples. Dans la mesure du possible, employez des exemples précis de ce que font les utilisateurs, des données qu'ils saisissent, des réponses du logiciel, ou n'importe quel exemple qui prendra le mieux en compte votre story.

Décomposez et amincissez. Quand vous discuterez des détails du développement, vous trouverez souvent que les stories sont plus volumineuses que ce que vous souhaitez placer dans un cycle de développement. Travaillez en groupe pour fractionner les grandes stories, ou « amincir » les stories en supprimant les options inutiles.

Votre atelier ne fonctionne pas quand...

– personne ne participe (une personne décrit ce qui est exigé, et tout le monde l'écoute sans réagir) ;

– lorsque vous vous concentrez uniquement sur les critères d'acceptation sans poser les questions du « qui », du « quoi » et du « pourquoi » ;

– lorsque vous ne parvenez pas à prendre en considération les options tant du point de vue fonctionnel que du point de vue technique.

16.4 SPRINT OU PLANIFICATION DES ITÉRATIONS ?

Certains praticiens agiles tiennent ces conversations critiques de stories au cours de sessions de planification, comme la planification des itérations ou la planification des sprints. Cela fonctionne assez bien si vous travaillez avec des équipes qui travaillent efficacement ensemble et qui participent à la discussion avec une bonne compréhension de leur produit. J'ai procédé de cette façon-là pendant des années avec les équipes avec lesquelles j'ai collaboré.

Mais l'une des principales récriminations que j'entends de la part des équipes agiles est que ces réunions de planification sont souvent longues et difficiles à suivre. À un certain moment, tout le monde est d'accord sur ce qu'il faut développer, même s'ils n'ont pas une compréhension mutuelle, car ils ont envie que la réunion prenne fin.

Revenir de loin

Nicola Adams et Steve Barrett, Assurances RAC, Perth, Australie

Ma première incursion dans l'environnement d'une équipe de projet agile en tant qu'analyste métier a été une leçon difficile sur la puissance de la collaboration par rapport à la parole écrite.

Nicola Adams

Le contexte

Nicola, analyste métier expérimentée, qui avait l'habitude de l'approche traditionnelle du développement logiciel, raconte son voyage initiatique vers les méthodes agiles au sein des Assurances RAC, à Perth, en Australie occidentale. Son rôle consistait à faire comprendre à l'entreprise le domaine qui posait problème et à travailler avec le service informatique pour documenter les spécifications fonctionnelles afin de gérer la livraison. Voici l'illustration du flux de communication :

© Dunod – Toute reproduction non autorisée est un délit.

Elle avait mis l'accent sur le détail des spécifications afin qu'il n'y ait plus d'inconnue. Se rendant compte que « les développeurs ne lisent pas », des stratégies d'atténuation ont été utilisées (par exemple, des procédures guidées des spécifications) avec un succès variable. En général, il s'écoulait un bon moment après l'écriture des spécifications pour que la connaissance puisse être utilisée dans le développement et les tests.

Que s'est-il passé au début ?

Son affinité naturelle pour les spécifications écrites n'a pas facilement été abandonnée. Le concept de remplissage des spécifications au dos d'une fiche a été difficile à comprendre. Comment Nicola pouvait attendre des développeurs et des testeurs qu'ils fournissent la fonctionnalité requise, dont elle se sentait responsable, s'ils n'avaient pas suffisamment d'informations ? L'accent a été mis sur la création de stories narratives, peu différentes des spécifications fonctionnelles sauf en termes de taille ; le flux de communication n'a pas été modifié.

Les préparatifs de la session d'élaboration de Nicola comprenaient :

– la collecte des spécifications avec les parties prenantes ;

– une analyse approfondie des besoins et des données ;

– la création de stories narratives (de une à cinq pages chacune) pour documenter les spécifications, la conception de la solution et les critères d'acceptation ;

– la présentation des stories à l'équipe à l'aide d'un projecteur et des questions.

Malheureusement, les résultats 'n'ont pas été pas merveilleux. Les sessions d'élaboration étaient ternes et sans intérêt, la majorité de l'équipe ne participant pas. En outre, Nicola a estimé qu'elle avait manqué de temps pour préparer des stories, et l'équipe a largement ignoré les critères d'acceptation lors de la livraison.

Après une session d'élaboration, Sam, un expert du domaine, a dit au PO : « S'il s'agit d'un projet agile, je ne veux pas participer ! ».

Il y avait donc un problème à résoudre !

Qu'est-ce qui a changé ?

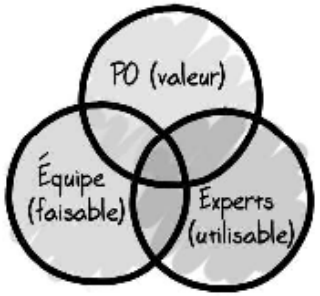
Steve, le chef de projet, a animé une rétrospective d'équipe pour se pencher sur le problème. Cette rétrospective a engendré un certain nombre d'actions importantes, notamment l'abandon des stories narratives, l'inclusion de l'équipe métier et de l'équipe de livraison dans l'élaboration des stories, et le respect d'une cadence fixe pour l'élaboration des stories et l'affinage du backlog.

Nicola est allée au-delà de la simple application des actions car elle a pleinement respecté l'objectif. La session suivante d'élaboration de stories a été radicalement différente de ce qui avait précédé.

L'équipe ne s'ennuyait plus et participait au fur et à mesure qu'ils s'investissaient dans les stories narratives projetées à l'écran. Ils étaient à présent rassemblés autour de modèles et d'artefacts visuels, absorbés dans de réelles conversations de stories impliquant le PO, les experts du domaine et l'équipe de livraison.

© Dunod – Toute reproduction non autorisée est un délit.

Le flux de communication avait changé. Nicola n'était plus l'intermédiaire entre l'entreprise et le service informatique ; elle était à présent un facilitateur qui permettait aux conversations de circuler entre ceux qui comprenaient la valeur de l'entreprise, ceux qui accompagnaient les utilisateurs pour savoir ce qui serait utilisable et l'équipe de livraison qui savait ce qui était faisable :



L'équipe métier et l'équipe de livraison appréciaient ce nouveau format et participaient à présent activement, une compréhension mutuelle autour des problèmes à résoudre s'étant installée ; les convergences et les divergences de groupe ont permis à l'équipe d'aboutir à des solutions optimales en respectant les contraintes et Nicola a senti baisser la pression et a disposé de plus de temps.

Nicola revenait de loin !

16.5 LES FOULES NE COLLABORENT PAS

La réunion de planification des sprints est devenue un tel cauchemar que de nombreuses équipes choisissent judicieusement de tenir ces discussions de stories dans les jours qui précèdent la réunion. Elles sont souvent programmées dans leurs agendas sous le nom de réunions de préplanification ou d'affinage de backlog, Mais, trop souvent, c'est la même séance de torture qu'ils haïssaient pendant la réunion de planification qui est déplacée à un autre jour. Et pour couronner le tout, les membres de l'équipe sont invités à faire une pause dans leur travail de production du moment pour assister à ce pensum. Il n'est guère étonnant qu'ils ne soient pas motivés.

Le problème n'est pas que les conversations de stories soient difficiles, même si dans certains cas, elles peuvent être assez dures, mais toutes les conversations sont rendues plus difficiles quand on tente d'y inclure trop de gens. Si un grand nombre de participants ne sont pas intéressés ou pas motivés, cela ne marchera pas. Vous voyez les gens dont je veux parler, ceux qui ne se cachent même pas pour jouer avec leurs smartphones sous la table.

Autorisez les membres de l'équipe à ne pas participer à ces conversations. Si plus tard ils se plaignent des décisions qui ont été prises, assurez-vous de les inviter à venir la prochaine fois.

Si tout le monde veut participer, essayez d'utiliser le modèle de collaboration en vase clos, comme celui qui est décrit dans l'encadré suivant. De cette façon, les personnes intéressées peuvent passer faire un tour, participer s'ils le veulent et quitter la réunion s'ils constatent qu'ils ne vont rien rater d'important.

Pattern de collaboration en vase clos (fishbowl)[1]

Si vous avez des gens qui veulent vraiment participer à la conversation, mais que leur nombre est trop important pour que la conversation reste efficace, essayez d'utiliser ce pattern. Cela va leur permettre de participer à la réunion sans trop impacter le résultat. Souvent, ils trouvent que le fait d'être là n'est pas aussi important qu'ils le pensaient. Au fil du temps, vous verrez qu'ils seront heureux de laisser les autres discuter des détails et d'en savoir plus sur les résultats dans une conversation ultérieure.

Le processus fonctionne de la manière suivante : trois à cinq personnes travaillent ensemble en vase clos avec un tableau blanc ou un tableau papier.

Les autres personnes présentes dans la pièce peuvent les observer mais n'ont pas le droit de parler. Ils sont à l'extérieur du premier cercle.

Si quelqu'un du cercle extérieur veut participer, il peut rentrer dans le cercle restreint, mais pour cela, une des personnes doit simultanément sortir du cercle restreint.

1. NdT : *fishbowl* signifie bocal en anglais, mais certains auteurs parlent de cercle excentrique : http://www.multibao.org/multibao/contributions/contributions/cercle_excentrique.md

De cette façon, la conversation demeure productive, et les autres personnes sont tenues informées et restent impliquées. C'est aussi un excellent moyen pour les apprenants de se mettre au niveau sans ralentir les travaux.

16.6 DÉCOMPOSEZ ET AMINCISSEZ

Vous vous souvenez de la discussion sur les gâteaux du chapitre 8 ? C'est le moment de décomposer ces gâteaux en mignardises les plus petites possible. C'est à ce moment, quand nous avons sous la main les développeurs, les testeurs et les autres personnes qui vont vraiment créer le logiciel, qu'il faut imaginer comment nous pourrions décomposer la story.

Un logiciel ressemble à un assez grand document ou à un livre. Si vous écriviez un livre, vous ne tenteriez pas de tout rédiger en même temps, mais vous écririez un chapitre à la fois. C'est d'ailleurs comme cela que je procède et Peter, mon éditeur compétent et au soutien indéfectible, révise ce que j'ai écrit pour y apporter des corrections et formuler des suggestions.

Mais pour autant, le chapitre n'est pas « terminé », loin s'en faut.

Je vais revenir sur le manuscrit pour déterminer l'emplacement des illustrations. J'ai besoin de savoir si je dois ajouter des notes, des références, un glossaire ou un index. Puis d'autres relecteurs réviseront chaque chapitre pour y apporter la touche finale. J'ai divisé naturellement le travail pour le faire par itération afin que je puisse voir l'ensemble du livre se concrétiser au plus tôt.

© Dunod – Toute reproduction non autorisée est un délit.

Vous lisez le présent chapitre et si vous le lisez maintenant, c'est qu'il est entièrement achevé, du moins nous l'espérons. Si je devais réfléchir à mes critères d'acceptation définitifs, j'établirais la liste suivante :

- rédigé et compréhensible par moi ;
- rédigé et compréhensible par mes éditeurs ;
- agrémenté par des illustrations qui aident les lecteurs à visualiser les points importants ;
- agrémenté par un index que les lecteurs peuvent utiliser pour rechercher des termes à l'intérieur des chapitres ;
- agrémenté par un glossaire que les lecteurs peuvent consulter pour rechercher des définitions de termes introduits dans les chapitres.

Cela représente beaucoup de travail. Même si j'intègre à présent toutes ces choses dans le premier jet de mon manuscrit, je me rends compte que j'ai beaucoup de travail à réaliser. Mais je ne veux pas faire tout ça avant de passer au chapitre suivant, parce que je voudrais voir comment se présente l'ensemble du livre. Alors je vais le diviser en « mignardises », petites pièces complètes qui ne sont pas prêtes à être livrées, mais qui me renforcent dans l'idée que je suis sur la bonne voie au fur et à mesure que j'avance dans la rédaction de cet ouvrage.

Je vais décomposer mon travail en stories de la manière suivante :

- affiner, définir et concevoir mon premier brouillon ;
- affiner, définir et concevoir mon deuxième brouillon amélioré ;
- affiner, définir et concevoir avec des illustrations ;
- affiner, définir et concevoir en incorporant les commentaires des réviseurs ;
- affiner, définir et concevoir en créant un index ;
- affiner, définir et concevoir en créant un glossaire ;
- affiner, définir et concevoir le projet final.

Chaque story passe à travers les étapes que je dois accomplir (avec l'aide de Peter, mon éditeur) pour terminer chacune de ces versions plus petites et apporter des améliorations au chapitre. Vous pouvez alors constater que quand chaque étape est terminée, le chapitre s'affine progressivement et s'approche de la version définitive. En théorie, vous pourriez lire chacune des versions intermédiaires une fois que la première story de la liste est terminée, mais je ne vous infligerai pas cela…

Enfin, vous avez peut-être remarqué que la liste des stories plus petites ressemble de très près aux critères d'acceptation des chapitres. C'est ici le côté magique des choses. C'est la discussion sur les critères d'acceptation qui révèle comment nous pouvons décomposer le travail en parties plus petites que l'on peut créer et inspecter tout au long du parcours.

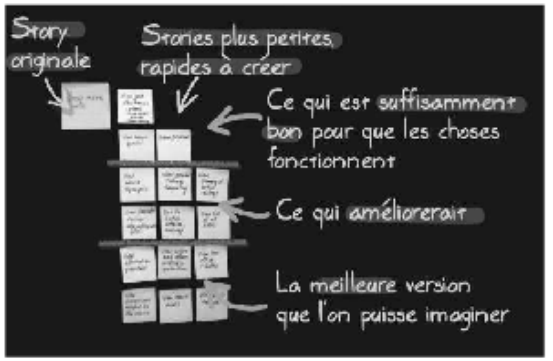
Il est important d'inspecter votre travail en cours de route afin de pouvoir l'évaluer et y apporter des corrections de trajectoire. Vous auriez pu voir ici l'exemple vraiment stupide que j'avais initialement écrit, mais vous ne le verrez jamais parce que je l'ai ré-examiné, puis je l'ai supprimé.

© Dunod – Toute reproduction non autorisée est un délit.

Dans un processus de logiciel traditionnel, cette pratique du « ré-examiner et supprimer » serait synonyme de mauvaises spécifications, mais quand vous avez votre casquette de développeur agile, il s'agit simplement d'apprentissage et d'amélioration itérative.

Jouez au jeu du bon-mieux-meilleur

Finalement, une de mes techniques préférées pour fractionner des stories est le jeu du bon-mieux-meilleur. Voici le résultat après avoir joué avec une grande story et des Post-it® :



Suffisamment bon pour l'instant

Prenez une story et commencez par une discussion sur ce qui serait juste assez bon, voire à peine suffisant, et sans doute pas assez bon pour que les utilisateurs ou les clients apprécient le produit. Notez les caractéristiques qui le rendraient suffisamment bon et traitez-les sous la forme de stories séparées et plus petites.

Si l'on prend l'exemple du site Web IMDb.com (http://imdb.com) qui est une base de données de films, on discute de la story « Voir les infos du film ». Nous avons imaginé un écran où l'on pourrait regarder les informations détaillées du film afin de décider si l'on veut le regarder. Lorsque nous avons discuté de ce qui serait suffisamment bon, nous sommes parvenus à une liste de choses simples :

– afficher les informations de base : titre, note, réalisateur, genre et ainsi de suite ;
– voir l'affiche du film ;
– visionner la bande-annonce.

Mieux

Puis on demande ce qui pourrait améliorer le produit. Pour notre exemple de base de données de films, nous avons établi la liste suivante :

– lire le synopsis d'un film ;
– lire les notes attribuées par les membres ;
– lire les notes attribuées par les critiques ;
– afficher la liste de tous les acteurs.

Meilleur

Enfin, on demande ce qui serait vraiment fabuleux. N'ayez pas peur de demander des choses folles ici. N'oubliez pas qu'il ne s'agit pas de spécifications, mais simplement d'envisager des options. Des choses intéressantes peuvent surgir parfois de ces discussions, des choses qui pourraient rendre le produit vraiment fabuleux et qui sont étonnamment bon marché à mettre en œuvre. Pour notre exemple de base de données de films, nous nous sommes retrouvés avec la liste suivante :

– regarder d'autres bandes-annonces ou des vidéos sur le film ;

– lire des anecdotes sur le film ;

– lire des actualités sur le film ;

– voir et participer aux discussions sur le film.

Vous pouvez voir combien cette progression de petites stories facilite la construction de la story « Voir les infos du film » car on part de quelque chose qui me permet juste de voir que cela fonctionne pour arriver à des stories qui me permettront d'améliorer le produit pour le rendre vraiment fabuleux. Si je devais développer cette feature, je ferais d'abord les bases de toute l'application avant d'améliorer les choses, pour ensuite réaliser le meilleur. En procédant de la sorte, je suis plus sûr de respecter les délais.

Quand vous avez vraiment de bonnes discussions de stories, à la fin d'un *atelier de stories*, vous devriez avoir des stories bien dimensionnées qui s'appuient sur une copieuse documentation et des modèles, ainsi que des critères d'acceptation qui décrivent la manière dont vous allez vérifier que cette story est bien terminée. Parfois, il faudra plusieurs ateliers s'appuyant sur un peu de recherches extérieures, des analyses et un travail de conception pour arriver à se mettre d'accord, mais tout ceci est normal. La décomposition et le polissage prennent du temps et il faut savoir être patient.

Recette de planification d'un cycle de développement

Les processus agiles, comme Extreme Programming et Scrum, utilisent un *développement limité par le temps*, où chaque cycle de développement commence par une séance de planification et s'achève par une revue. Dans de nombreuses entreprises, ce sont les réunions les plus détestées car elles peuvent être longues et pénibles, et au moment où les membres de l'équipe quittent la réunion, ils sont souvent prêts à accepter n'importe quoi tout simplement pour mettre un terme à la réunion. Il ne faut pas être un génie pour imaginer que la qualité de la planification laisse à désirer.

Mais il n'y a pas de raison pour que cela se passe comme ça.

Voici une recette simple qui devrait vous aider à éviter les pires problèmes.

Préparez

Choisissez les stories un ou deux cycles à l'avance. Si vous êtes le PO, rencontrez régulièrement votre équipe de base du produit pour discuter des progrès des solutions en cours. Choisissez les stories sur lesquelles vous voulez travailler par la suite.

Animez un atelier anticipé. Prenez le temps de travailler avec des membres de l'équipe produit avant la séance de planification. Approfondissez les détails, décomposez les grandes stories et envisagez plusieurs options. Reportez-vous à la story de Mat Cropper au chapitre 7 : il attache énormément d'importance aux séries d'ateliers de stories d'une demi-heure où il prépare les développeurs et les testeurs à la planification.

Invitez l'ensemble de l'équipe et les autres personnes dont vous pourriez avoir besoin pendant le futur cycle de développement.

Planifiez

Commencez par examiner l'objectif principal du prochain cycle. Vous avez choisi de travailler certaines stories. Comment ce groupe de stories facilite l'avancement de la solution que vous essayez de livrer ?

Révisez les stories que vous allez discuter. Ne rentrez pas dans les détails ici, mais suffisamment pour donner à chacun une vue d'ensemble. Reportez-vous à la story de Nicola et de Steve dans ce chapitre. Regardez le mur devant lequel se trouve Nicola : il y a beaucoup de mots et d'images pour aider les membres de l'équipe à se faire une idée.

Définissez une limite de temps pour que l'équipe de livraison puisse procéder à sa propre planification. Rappelez-vous que les foules ne collaborent pas. Et les personnes développant et testant ce logiciel doivent vraiment réfléchir à la création de leurs recettes pour créer ces stories, tout comme Sydnie l'a fait au chapitre 10. Donnez à votre équipe une heure pour se séparer en petits groupes et travailler ensemble sur les stories. Si vous êtes PO, concepteur d'interface utilisateur ou analyste métier, restez à proximité. Observez si vous le souhaitez, mais soyez prêt à répondre aux questions qui les aideront à avancer rapidement.

En petits groupes, créez une planification pour chaque story. N'oubliez pas les trois amigos que j'ai évoqués au chapitre 12. Assurez-vous que les petits groupes fonctionnent comme ça. Et, en tant qu'équipe de développement, décidez du nombre de stories qui peuvent être terminées avec succès au cours du cycle de livraison. N'oubliez pas de prendre en compte les vacances et les jours de repos. J'ai eu une fois une équipe qui m'a dit que leur planification ne serait pas respectée à cause des vacances de Noël, comme si ce jour férié sortait de nulle part et les avait pris par surprise.

Tous ensemble, mettez-vous d'accord sur la planification. À la fin du temps imparti, et après que l'équipe a réalisé la planification de chaque story, ses membres auront besoin de partager leur planification, mais pas tous les détails parce que ce serait très ennuyeux, même pour eux. Ce qui est important, c'est que l'équipe soit au clair sur ce qu'ils croient pouvoir réaliser dans le cycle. Ils doivent prendre au sérieux cet accord et leur planification, surtout s'ils veulent que les autres les considèrent comme fiables.

Se mettre d'accord peut prendre un peu de temps, surtout si tout le travail qui doit être réalisé n'est pas faisable dans le temps imparti au développement. Il est heureux que vous connaissiez quelques astuces pour décomposer encore plus finement certaines stories. Dans ce cas-là, essayez de faire passer la story de l'état « mieux » à l'état « suffisamment bon », ce qui devrait faire l'affaire.

Célébrez. Vous avez terminé. Par le passé, nous aimions réaliser la planification dans l'après-midi, et on se débrouillait pour terminer un peu avant l'heure de quitter le bureau, puis on faisait la fête. Le lendemain, on était revigoré et prêt à commencer à travailler sur la planification que nous avions créée ensemble.

© Dunod – Toute reproduction non autorisée est un délit.

16.7 UTILISEZ VOTRE STORY MAP PENDANT LA LIVRAISON

Utilisez une carte pour forger la compréhension mutuelle avec votre équipe de livraison. J'entends souvent des membres d'une équipe travaillant en méthode agile dire combien ils aiment le travail collaboratif et combien ils se sentent productifs parce que toutes les semaines ou toutes les deux semaines ils peuvent présenter un logiciel fonctionnel. Mais, ensuite, ils complètent leurs propos par des déclarations du genre : « 'Mais c'est comme si j'avais perdu la vision d'ensemble. Je ne vois plus que les petites parties du produit que nous développons. » Utilisez une carte pour donner à l'équipe la visibilité de l'ensemble du produit ou de la feature sur laquelle vous travaillez. Ils feront de meilleurs choix de conception tactique et prendront de meilleures décisions relatives au développement s'ils comprennent le contexte général.

16.8 UTILISEZ UNE CARTE POUR VISUALISER LES PROGRÈS

Quand vous commencez à développer votre release du produit, la carte constitue un bon tableau de bord visuel pour afficher ce qui a été développé et ce qui ne l'a pas été.

Certaines équipes enlèvent les stories détaillées du corps de la carte quand elles les mettent en chantier. De cette façon, quand elles regardent la carte, elles ne voient plus que ce qui reste à développer.

D'autres équipes préfèrent laisser la carte en place et à l'aide de stylos ou de Post-it® en couleur, elles indiquent les stories qu'elles ont terminées. Quand elles prennent du recul et regardent la carte, elles voient une visualisation de ce qui est fait et de ce qui reste à faire.

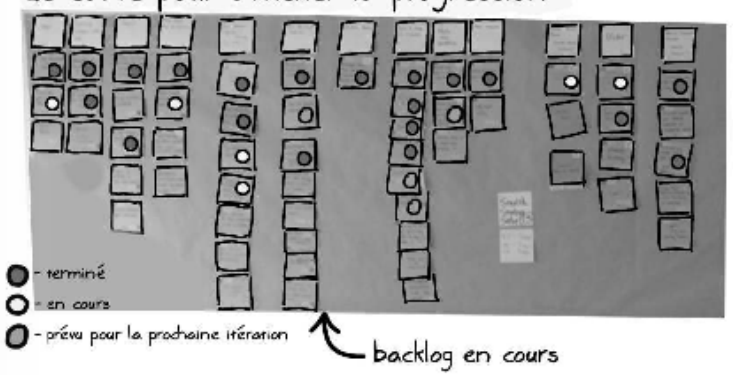
La carte permet d'identifier les stories suivantes à développer. Chaque semaine, les PO évalueront l'avancement des travaux en cours de développement et prendront des décisions sur les choses importantes sur lesquelles il faut se concentrer à l'avenir. Lorsque la carte visualise les progrès, il est alors un peu plus facile de rechercher les zones qui ont besoin de plus d'attention. C'est un peu comme si vous étiez un peintre : si vous pouvez prendre du recul et embrasser du regard la totalité de la toile, il est plus facile de savoir par où commencer la tâche suivante.

16.9 UTILISEZ DES CARTES SIMPLES AU COURS DES ATELIERS DE STORIES

Au cours de chaque cycle de développement, vous allez identifier sur la carte les stories sur lesquelles vous allez travailler par la suite. Vous allez placer ces stories dans ces dernières bonnes conversations que vous aurez pendant les ateliers de stories.

© Dunod – Toute reproduction non autorisée est un délit.

Stories étiquetées dans un backlog sous forme de carte pour afficher la progression



Une carte simple est un moyen de visualisation facile à construire au cours de cet atelier. Vous ne pouvez cartographier que les trois ou quatre étapes que l'utilisateur suit au cours de la feature dont vous discutez. Mais le fait d'être en mesure de pointer sur le mur des Post-it® qui indiquent le flux contribue à fluidifier la discussion. Lorsque vous commencez à discuter des critères d'acceptation, écrivez-les sur des fiches et ajoutez-les à la minicarte. En fin de compte, vous aurez une visualisation simple qui s'appuie sur la conversation que vous avez eue dans cet atelier.

Visualisez le fonctionnement de votre backlog

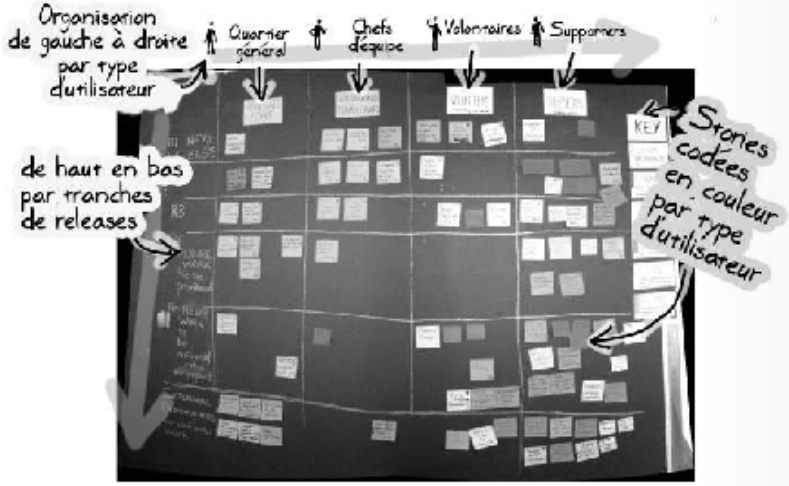
Chris Gansen et Jason Kunesh, Tableau de bord de la campagne d'Obama

Comme l'a prouvé la campagne électorale d'Obama en 2008, Internet a changé pour toujours la politique. La stratégie Internet de Barack Obama a joué un rôle prépondérant dans son investiture lors de la primaire, puis dans son élection. La stratégie pour l'élection de 2012 devait fournir des outils de prise en charge de collectes de fonds pour le grand public, tout en utilisant la technologie comme un « démultiplicateur de force » pour contrer l'énorme quantité d'argent des adversaires qui entraient dans la course. Alors que nous avons utilisé des outils comme Basecamp et Pivotal Tracker pour suivre le travail de nos équipes pour soutenir le Tableau de bord de campagne d'Obama 2012, afin que tout le monde puisse vraiment comprendre ce qui se passait, nous avons aussi utilisé des murs couverts de Post-it®. Vous vous demandez sans doute pourquoi nous avons pris le temps de faire joujou avec tout un tas de Post-it® sur le mur. En voici la raison.

Il y avait deux cultures différentes qui travaillaient ensemble dans le même espace. Nous étions un certain nombre dans notre coin qui dévissions les ampoules pour pouvoir travailler dans une obscurité relative, qui apportions nos propres claviers et qui portions des casques gigantesques pour étouffer le bruit des interviews à la télévision et des applaudissements qui rythmaient le bureau qui servait de salle de presse ; et qui portions aussi des T-shirts de groupes de rock au lieu de costumes. Et puis il y avait

tous les types en costard qui avaient travaillé sur les campagnes précédentes avaient une vision traditionnelle du développement logiciel et disaient : « *Nous décrivons les caractéristiques de ce que nous voulons, et vous devez les développer pour nous et pour hier* ». Ils n'étaient pas habitués à recevoir les choses de manière progressive et à les voir évoluer et s'améliorer de manière itérative, et ils n'avaient certainement pas l'habitude de faire les compromis nécessaires pour obtenir les choses dont ils avaient vraiment besoin. Il s'agissait d'une campagne électorale, et il n'y avait pas de possibilité de repousser la date de livraison, sauf en faisant voter une loi par le Congrès ! Au lendemain de l'élection, c'est terminé que nous ayons fini ou non. Et il n'y avait aucun moyen pour qu'ils obtiennent tout ce qu'ils avaient imaginé.

Le Président Obama en costume et Jason en polo, en train de faire la démonstration de produit la plus stressante de sa vie.



Le premier tableau.

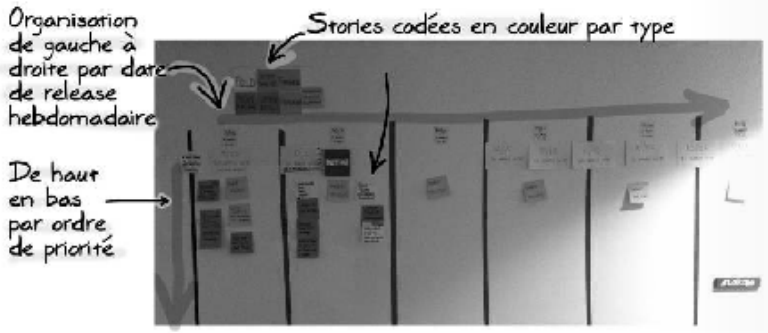
© Dunod – Toute reproduction non autorisée est un délit.

Lorsque nous avons commencé, nous avons utilisé quelques approches de mapping de base pour décrire aux gens qui allaient utiliser notre système les différentes sortes de choses qu'ils auraient besoin de faire. Nous pourrions ensuite organiser les travaux en releases au fil du temps. Cela a parfaitement fonctionné, mais il était difficile pour les personnes menant la campagne de vraiment s'impliquer sur le moyen terme, et sur la façon dont les gens travailleraient dans un an. Leurs esprits étaient concentrés sur ce qui devait être fait *maintenant*. Et ils faisaient beaucoup d'hypothèses sur ce que les bénévoles et les animateurs feraient. Ils ont dû faire d'autres hypothèses, en particulier parce que nous étions en train de réinventer la façon dont ils travaillaient.

Après sept mois, nous avons expédié un petit produit viable minimum, juste suffisant pour livrer une version à nos premiers utilisateurs dans l'Iowa. C'est alors que tout a changé. La première réaction a été mauvaise parce que ce que nous avions livré n'était manifestement pas ce que les gens avaient imaginé. Il y avait beaucoup de choses importantes qui manquaient et il avait des bugs. Nous avons entendu un grand nombre de « *Pourquoi est-ce que cela ne fonctionne pas du premier coup ?* ». Nous leur avons assuré que nous allions réparer ce qui ne marchait pas et améliorer les choses à un rythme hebdomadaire, mais ils ne pouvaient pas vraiment nous croire tant qu'ils n'avaient rien vu. Mais comme les choses se sont améliorées rapidement, cela a vraiment permis d'établir la confiance.

C'est là qu'intervient le grand mur couvert de Post-it®. Nous avons utilisé Pivotal Tracker et Basecamp, mais toutes les autres personnes n'avaient pas à utiliser ces outils. Nous avions besoin de présenter de façon totalement transparente ce sur quoi nous travaillions et ce qui allait être livré. Notre mur de stories a été organisé de gauche à droite par semaine du calendrier, et de haut en bas par priorité. Nous étions tous concentrés sur le temps. Il y avait une grosse horloge sur le mur qui comptait à rebours les minutes jusqu'à l'élection. Tous savaient que certaines choses étaient maintenant moins importantes et que d'autres devenaient plus critiques au fur et à mesure que le jour de l'élection se rapprochait. Toutes les stories sur le mur étaient codées par couleur en fonction de l'activité (travail sur le terrain, motivation des équipes, inscription des électeurs, participation électorale). Il y a beaucoup de violet sur la photo illustrant ce mur parce qu'on est encore loin de l'élection et que l'activité de motivation des équipes, représentée par les Post-it® violets, est plus importante que les autres choses comme le taux de participation du moment.

Le compte à rebours.



Releases hebdomadaires de gauche à droite, avec une priorité de haut en bas, codées par couleur par zone du produit.

Lorsque nous ajoutions des choses au tableau, nous le faisions avec les gens qui dirigeaient la campagne. Nous parlions de ce qui était déjà prévu pour cette semaine et des risques encourus. Nous parlions du véritable problème résolu derrière l'idée de la fonctionnalité. Ensemble, nous décidions ce qui était important par rapport à toutes les autres choses sur le mur. Quand viendrait le temps de développer le logiciel, le développeur responsable travaillerait directement avec la partie prenante qui en savait le plus et ils collaboreraient pour déterminer les détails. Parfois, tout ce qu'il fallait pour se mettre d'accord, c'était beaucoup de gribouillis sur un tableau blanc. D'autres fois, nous prenions un jour et développions quelques prototypes simples de l'interface utilisateur.

Ce grand mur visuel de Post-it® était essentiel pour construire un pont qui reliait les personnes qui développaient le logiciel et les gens qui en avaient besoin. C'est ce dont ils avaient besoin pour visualiser ce qui se passait et pour participer activement à la prise de décisions.

17

Les stories sont comme des astéroïdes

Si vous n'êtes plus tout jeune, vous vous souviendrez avec émotion d'avoir joué à un des premiers jeux vidéo qui s'appelait *Astéroïdes*. Restez attentif un moment car je vous promets que cela a un rapport avec ce qui nous intéresse.

Dans ce jeu, vous êtes représenté par un petit vaisseau flottant dans l'espace intergalactique, mais vos déplacements sont restreints par un champ d'astéroïdes énormes, et vous devez vous frayer, à coup de canons, un chemin pour survivre ! Si vous tirez sur un gros astéroïde, il explose en plusieurs astéroïdes plus petits. Et pour rendre les choses encore plus compliquées, ces astéroïdes plus petits se déplacent plus rapidement et dans des directions différentes, si bien qu'il est plus difficile de leur échapper. Si vous tirez sur un de ces petits astéroïdes, il va se décomposer en astéroïdes encore plus petits qui iront encore plus vite dans des directions *différentes*. Très vite, l'écran est rempli d'astéroïdes de toutes tailles se déplaçant dans toutes les directions possibles. Heureusement, lorsque vous tirez sur les astéroïdes les plus petits, ils explosent complètement et éclaircissent le terrain.

Une très mauvaise stratégie consiste à tirer sur tous les gros blocs qui se décomposent en petits rochers. L'écran se remplit alors d'innombrables rochers qui bougent dans tous les sens, et c'est une mort rapide assurée.

Une très mauvaise stratégie de gestion du backlog consiste à décomposer toutes les grandes stories afin qu'elles soient assez petites pour tenir dans le prochain cycle de développement. Votre backlog va se remplir avec beaucoup de petites stories qui partent dans tous les sens, et vous allez mourir. Bien entendu, vous n'allez pas vraiment mourir, mais vous allez être enterré vivant par un grand nombre de difficultés inutiles. Tout le monde va se plaindre de perdre la vision d'ensemble au milieu de tous ces petits détails.

Il faut donc décomposer progressivement les stories et au bon moment.

À chaque discussion de story et au moment de décomposer, vous aurez ces objectifs en tête :

1. *Pour les opportunités*, vous allez vous demander pour qui elles sont, quels problèmes elles résolvent, et si elles sont bien alignées avec votre stratégie d'entreprise. Il peut alors être judicieux de décomposer les opportunités volumineuses.
2. *Au cours de la découverte*, vous allez discuter de façon détaillée de qui utilise le produit, mais aussi du « pourquoi » et du « comment ». L'objectif de l'équipe est d'envisager un produit qui soit valable, utilisable et faisable. Vous allez décomposer beaucoup de blocs ici, mais heureusement vous transférerez seulement le nombre minimal de stories dont vous avez besoin dans le backlog de release qui décrit une version viable minimum du produit.
3. *Lors de la planification d'une stratégie de développement*, vous allez discuter des risques, ceux qui découlent des attentes des utilisateurs et de ce que ces derniers vont adopter, mais aussi ceux qui découlent des problèmes de faisabilité technique. Vous allez décomposer les blocs en pensant à l'apprentissage, en développant ce dont vous avez d'abord besoin, pour en apprendre le plus possible.
4. *Lors de la planification du prochain cycle de développement*, vous aurez vos ultimes discussions pour décider exactement de ce qu'il faut développer et vous mettre d'accord sur la manière de vérifier que le logiciel est bien terminé. Chacune de ces décisions permet une décomposition plus fine des stories.

Si vous avez essayé d'avoir ces quatre conversations au cours de la même séance, cela a donné lieu à une conversation longue et épuisante. Il faudrait une grande variété de personnes pour peser sur les différents aspects, mais vous avez retenu de mes avertissements et de votre propre expérience que les grands groupes de personnes ne travaillent pas efficacement ensemble, du moins pas quand tout le monde est dans la même pièce en même temps. C'est pourquoi nous décomposons les stories progressivement, au fil du temps, et en diversifiant les conversations.

17.1 RÉASSEMBLAGE DES BLOCS DÉCOMPOSÉS

Dans le jeu *Astéroïdes*, vous devez être très prudent car vous ne pouvez pas recomposer les astéroïdes sur lesquels vous avez tiré et que vous avez décomposés. En revanche, vous pouvez réassembler des stories qui ont été décomposées.

Pour éviter qu'un backlog ne soit rempli que de petites stories, vous pouvez prendre une série de stories qui vont ensemble et écrire tous leurs titres sur une seule fiche sous la forme de liste à puces. Si vous résumez ces titres sous un seul titre que vous écrivez sur une nouvelle fiche, vous obtenez une grande story.

C'est quelque chose d'assez fantastique si vous y réfléchissez bien. La fiche et le titre qui figure dessus sont un lien tangible vers de nombreuses idées. Les idées sont beaucoup plus malléables que des blocs de roche ou de lourds documents. Nous oublions parfois que le développement de logiciels est aussi un travail intellectuel. Quand nous l'oublions et qu'au contraire nous nous focalisons sur les documents et les processus, ce travail se transforme en quelque chose d'aride et d'administratif. Et quand je travaille avec des personnes qui gèrent un énorme backlog rempli de petites stories, cela semble terriblement administratif.

Regroupez les petites stories pour nettoyer votre backlog

Je suis souvent confronté à des équipes produit qui ont des backlogs contenant des centaines d'éléments. Et, comme prévu, ils me disent qu'ils luttent pour définir les priorités de leurs backlogs. Quand j'examine leurs backlogs, ils sont souvent remplis de nombreuses petites stories. Si l'on doit évoquer chacune d'entre elles pour prendre une décision de priorisation, cela peut prendre des heures, voire des jours dans certains cas, si bien qu'il vaut mieux s'abstenir.

Si vous jouiez à *Astéroïdes*, vous auriez perdu, mais comme ce n'est pas le cas, essayez de regrouper vos petites stories pour en faire des stories plus grandes :

© Dunod – Toute reproduction non autorisée est un délit.

1. Si vos stories sont dans un backlog électronique, écrivez-les sur des fiches ou des Post-it®. Quel que soit l'outil que vous utilisez, il doit être en mesure d'imprimer ou d'exporter vers une feuille de calcul. J'utiliserais un simple publipostage dans un programme de traitement de texte pour créer des étiquettes pour toutes les stories et ensuite je les collerais sur une fiche ou je les imprimerais directement sur les fiches.

2. Demandez de l'aide aux membres de l'équipe qui comprennent le système. Réservez une salle avec beaucoup d'espace sur les murs ou sur les tables pour travailler.

3. Donnez à chacun une série de fiches de stories et demandez-leur de commencer par les placer sur les tables ou de les coller sur les murs.

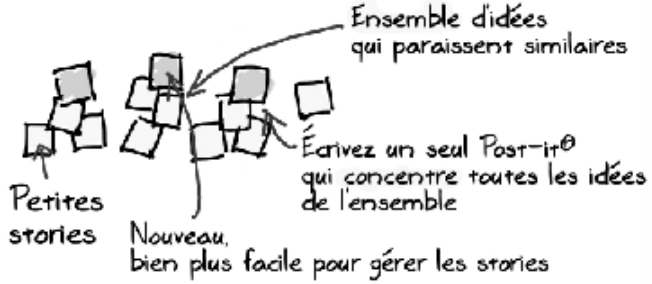
4. Lorsque vous voyez une fiche qui semble similaire à celle que vous placez, regroupez-les. Ne réfléchissez pas trop sur la signification du mot « similaire » et allez-y à l'instinct.

5. Procédez de la sorte en silence, au moins pour commencer. Vous constaterez que ce sont les conversations qui ralentissent les choses, et il est bon d'apprendre à utiliser ce pattern et votre langage corporel pour communiquer.

6. Déplacez et réorganisez n'importe quelle carte à votre guise. C'est le modèle de tout le monde, ce qui signifie que personne ne possède la position d'une fiche. Si quelque chose ne vous paraît pas en place, déplacez-le. Si quelqu'un n'est pas d'accord, il peut le remettre à sa place et c'est à vous de demander des explications.

7. Après avoir regroupé les choses, prenez une fiche de couleur différente et créez un titre pour chaque tas. Sur cette fiche, écrivez un meilleur nom de story, qui explique le lien entre toutes ces fiches. Si vous avez écrit un résumé qui s'intitule « Améliorations de l'interface utilisateur », c'est peut-être trop vague, et « Amélioration de la saisie et de la modification des commentaires » serait préférable, en supposant que ces modifications de l'interface utilisateur s'appliquent à des commentaires.

8. Les résumés deviennent vos nouvelles stories qui sont plus grandes. Les autres fiches deviennent des puces de leur description. Replacez-les dans votre backlog. Ou, s'ils peuvent être reportés, replacez-les dans votre backlog d'opportunités.



Cela fonctionne à merveille pour les backlogs volumineux composés de nombreux petits éléments. C'est aussi parfait pour les listes de bugs importantes. Vous savez bien qu'il y a toujours beaucoup de bugs qui ont une priorité moins élevée et qui ne sont jamais corrigés... Regroupez-les avec les autres bugs de priorité plus élevée dans la même zone du système. Quand un développeur va corriger les bugs de priorité élevée, il est souvent simple de corriger en même temps les bugs moins importants. Vos clients et vos utilisateurs vous en seront reconnaissants.

17.2 N'ABUSEZ PAS DU MAPPING

J'entends souvent des gens qui essaient de comprendre le story mapping dire que c'est vraiment « too much ». Quand je leur demande ce qui les dérange, ils me répondent qu'ils ont créé une très grande carte de la totalité de leur système pour discuter d'une seule feature. Ils ont raison : c'est « too much » et il ne faut pas faire cela.

Cartographiez uniquement ce qui vous suffit pour rédiger une story sur votre feature.

Par exemple, j'ai travaillé avec une entreprise qui faisait quelques modifications de la fonctionnalité de commentaires de son logiciel de GED collaboratif. L'équipe a cartographié l'édition de documents à un niveau très élevé, et ils n'ont utilisé que quelques fiches pour cela. Quand ils sont arrivés à la zone de commentaires, ils ont ajouté plus de fiches qui résumaient ce que leur produit faisait à ce moment-là en utilisant beaucoup de puces sur une seule fiche. Puis ils ont commencé à discuter des changements qu'ils aimeraient faire, en ajoutant beaucoup de fiches pour tous les détails et les options qu'ils envisageaient.

Lorsque vous ajoutez une feature dans un produit existant, cartographiez un peu avant l'endroit où la feature commence dans votre story, et un peu après l'endroit où elle se termine. Ne cartographiez pas la totalité du produit.

N'oubliez pas que les story maps prennent en charge les conversations au sujet de vos utilisateurs et de vos idées de produits. Voici une bonne règle de base à suivre : si vous n'avez pas besoin d'en parler, vous n'avez pas besoin d'en faire une story map.

17.3 NE VOUS PERDEZ PAS DANS LES DÉTAILS

J'ai décrit cette étape de décomposition des blocs et je vous ai même encouragé à traiter ces blocs comme les astéroïdes de cet ancien jeu tournant sur Atari, afin que vous ne soyez pas tentés de les décomposer trop vite. Derrière toutes ces stratégies, l'hypothèse est qu'un grand nombre de stories auxquelles nous sommes confrontés sont grandes. Mais, en fait, pour beaucoup d'entre elles, ce n'est pas le cas. Après avoir livré un produit ou une feature à des utilisateurs, vous constaterez immédiatement que beaucoup de petites choses sont totalement évidentes (il s'agit des choses auxquelles vous vouliez penser avant la livraison et que vous avez oubliées). En tous les cas, c'est ce qui m'arrive souvent. Pour ces petites choses je n'ai pas de discussion d'opportunité, et je n'ai pas rassemblé un groupe pour faire la découverte du produit, parce qu'il est évident pour tout le monde que ces choses-là doivent être faites. Je place toutes ces choses dans un backlog de release et, dès que possible, j'organise un atelier avec les membres de l'équipe afin que nous puissions les développer. Il en va de même pour les bugs et pour beaucoup de petites améliorations.

© Dunod – Toute reproduction non autorisée est un délit.

Sauvez le monde, mais petit à petit

Voici de nouveau mon ami Sherif de la société Atlassian. Il m'expliquait que les membres de son équipe produit travaillaient constamment sur de nombreuses petites corrections et améliorations. Comme ils s'inquiètent beaucoup du confort de leurs utilisateurs, ils corrigent beaucoup de petits bugs et des imperfections qui leur rendent la vie difficile. Ainsi, sur le mur d'une équipe travaillant sur un produit nommé GreenHopper, il y a une série de barres verticales. Chaque fois qu'un membre de l'équipe corrige un de ces petits bugs, il fait une marque sur le mur. Il semblerait que 47 de ces petits bugs seront corrigés dans la prochaine release. Si vous utilisez Confluence ou JIRA, vous pouvez les remercier...

18

Tirez un enseignement de tout ce que vous développez

Si vous avez un profil de développeur classique, vous pensez que vous avez terminé votre travail lorsque vous avez fini de développer le logiciel. Mais le développement agile et ses stories est conçu pour l'apprentissage. Nous passons beaucoup de temps, avant de développer quoi que ce soit, à nous assurer que nous devons le faire et à nous mettre d'accord sur ce qu'il faut développer. Et quand le développement est terminé, nous examinons une nouvelle fois le résultat et nous nous demandons si nous avons bien fait de développer ce produit et s'il est suffisamment bon.

Nous allons parler de toutes les occasions que vous avez de vous instruire une fois le développement achevé.

18.1 DÉBRIEFEZ EN ÉQUIPE

Revenons en arrière au moment où vous célébrez l'achèvement du logiciel. À la fin d'un cycle de développement et de tests, il est normal de faire la fête. Vous avez transformé quelques idées, beaucoup de discussions et des croquis en un véritable logiciel fonctionnel. Il aurait fallu beaucoup plus de temps si l'on avait employé un processus traditionnel basé sur des spécifications, et votre équipe éprouverait probablement un sentiment beaucoup moins grand d'appropriation du résultat.

Après les félicitations réciproques, il est temps de réunir l'équipe et d'examiner honnêtement ce que vous avez réalisé. Si vous êtes honnêtes avec vous-mêmes, vous allez probablement trouver certaines choses à améliorer dans le logiciel. Pour chacune de ces choses, vous allez écrire une nouvelle story et l'ajouter à votre backlog de release. Vous allez décider s'il s'agit de changements à effectuer immédiatement ou s'ils peuvent être reportés à un peu plus tard.

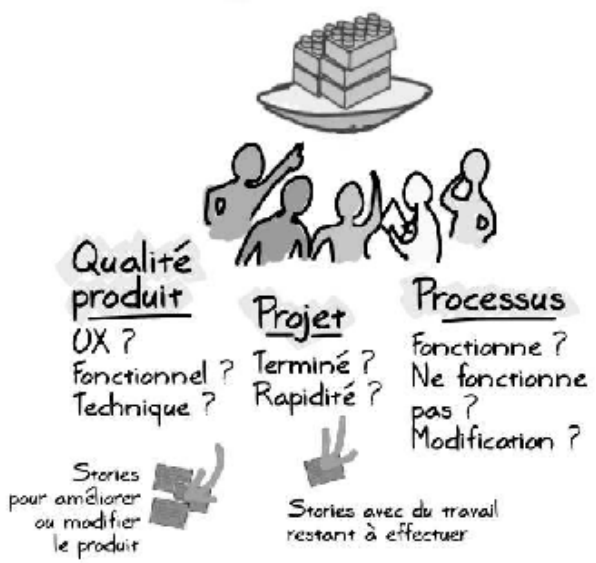
© Dunod – Toute reproduction non autorisée est un délit.

Dans Scrum, cela s'appelle une *revue de sprint*. Si vous utilisez Scrum, vous avez peut-être entendu que tout le monde était le bienvenu à cette revue, mais je vous suggère d'opérer différemment.

L'équipe, qui a travaillé en étroite collaboration, a discuté une dernière fois sur les stories, s'est mise d'accord sur ce qu'il fallait développer, et a travaillé à la réalisation du produit. Elle a besoin d'un moment et d'un endroit où elle puisse discuter en toute liberté de ses travaux. Ce que les personnes extérieures à l'équipe, y compris la direction de l'entreprise, pensent du produit est important et l'équipe a besoin d'entendre leurs opinions. Mais ces personnes n'ont pas participé aux conversations qui ont abouti au consensus sur les détails de ce qu'il fallait développer. Elles ne participaient pas non plus aux discussions qui ont permis d'élaborer des plannings détaillés de développement du logiciel. Et elles n'ont pas travaillé aux côtés de l'équipe quand toutes ces discussions et ces accords ont permis d'aboutir à un logiciel fonctionnel. Nous devons d'abord évaluer si nous avons réussi à développer ce que nous avons envisagé au niveau de qualité espéré et dans les délais prévus. Vous pouvez réaliser cela au cours d'une *revue de produit en équipe et de réflexion*.

Recette de la revue de produit en équipe

L'équipe qui a travaillé pour comprendre les stories et élaborer un planning à court terme pour leur développement, doit s'arrêter et réfléchir à la qualité de son travail. Animez un court atelier pour y parvenir.



Limitez le nombre de participants à cet atelier pour n'y inclure que les personnes qui ont travaillé ensemble pour planifier le travail. Invitez le PO et les autres membres de l'équipe de produit, ainsi que les développeurs, les personnes chargées de l'assurance qualité et toute personne qui a participé activement à la livraison du produit fonctionnel. Oui, je vous confirme que je suis d'accord pour exclure les parties prenantes. Nous partagerons avec eux suffisamment tôt, mais pour l'instant, nous avons besoin d'un endroit pour parler en toute liberté.

Apportez de la nourriture. Il y a quelques années, dans mon équipe, cet atelier ne pouvait tout simplement pas commencer si nous n'avions pas de bagels.

Utilisez cet atelier pour *examiner trois choses : le produit, la planification et le processus*.

Le produit

Commencez par examiner le logiciel développé en tant que résultat des stories. Assurez-vous de le tester en situation réelle sur un écran. Et prenez la peine de *tout* tester. Dans les grandes équipes, ce sera peut-être la seule occasion que tout le monde puisse voir le travail de chacun.

Évaluez la qualité de matière subjective en équipe. Cette évaluation engendrera beaucoup de bonnes discussions.

– *Discutez de la qualité de l'expérience utilisateur*. Ne vous contentez pas de l'aspect de l'interface utilisateur, mais discutez de ce que l'on ressent quand on utilise le produit. Est-il aussi réussi que prévu ? Évaluez-le sur une échelle de 1 à 5, 5 étant la meilleure note.

– *Discutez de la qualité fonctionnelle*. Est-ce que les tests se sont bien passés, ou est-ce qu'il y a beaucoup de bugs ? Est-ce que les testeurs s'attendent à trouver davantage de bugs seulement au moment où l'on ajoutera du code, ou bien s'ils ont plus de temps pour continuer à en chercher ? Évaluez-la sur une échelle de 1 à 5, 5 étant la meilleure note.

– *Discutez de la qualité du code*. Avez-vous écrit du code qui sera facile à maintenir ? Ou vous êtes-vous contenté de coder à l'ancienne ? Évaluez-vous sur une échelle de 1 à 5.

Écrivez des stories pour corriger les problèmes de qualité que vous avez constatés dans le produit.

Si vous êtes engagé à la fois dans un travail de découverte et de livraison, ce qui devrait être le cas, discutez de votre travail de découverte du dernier cycle. Qu'avez-vous fait ? Qu'avez-vous appris ?

La planification

Si vous travaillez dans une itération limitée dans le temps ou dans le cadre d'un sprint, vous avez commencé en faisant une planification et en prévoyant ce que vous pourriez faire dans le temps imparti. La prévision était-elle exacte ?

– *Décidez les stories qui sont finies et celles qui ne le sont pas*. Cela peut être plus difficile que vous ne le pensez. Le fait d'avoir cette discussion va aider votre équipe à élaborer une définition commune de ce qu'elle considère comme étant terminé. Est-ce que « terminé » signifie qu'il y a des tests automatisés ? Est-ce que cela signifie que tous les tests manuels ont été effectués ? Ou bien est-ce que cela veut dire que les PO ou les concepteurs d'interface utilisateur ont passé le produit en revue ?

– Comptez le nombre de stories que vous considérez comme terminées. Il s'agit de votre *vélocité*.

– Comptez le nombre de stories qui sont commencées et non terminées. S'il y en a beaucoup, cela indique que vous devez travailler votre planification.

© Dunod – Toute reproduction non autorisée est un délit.

– Discutez du temps prévu pour le *travail de découverte*. Avez-vous utilisé tout le temps prévu ? Avez-vous utilisé plus de temps que prévu ? Si vous utilisez trop peu de temps, cela vous causera des soucis plus tard quand vous ne serez pas prêt pour développer des choses en totale confiance. Si vous utilisez trop de temps, cela vous causera des soucis pour livrer à temps ce que vous avez promis.

Le processus

Discutez de la façon dont vous avez travaillé au cours du dernier cycle de développement. Pouvez-vous apporter des modifications à votre façon de faire les choses pour améliorer la qualité ou la planification ? Est-il possible de rendre le travail plus amusant au quotidien ? Parce que si vous vous amusez, je vous promets que vous serez en mesure de travailler plus rapidement[1].

– Commencez par discuter des modifications que vous avez essayées lors du dernier cycle. Est-ce que cela a marché ? Voulez-vous les conserver ou les abandonner ?

– Discutez des changements que vous voulez essayer lors du cycle suivant. Ne soyez pas trop gourmand car les petits changements sont les meilleurs. Si vous essayez de changer trop de choses à la fois, c'est comme si vous tentiez de prendre trop de travail à la fois. Vous courez à la déception.

Voilà, c'est tout ! Vous avez beaucoup appris des stories que vous avez développées et de *tout* le travail que vous avez réalisé.

18.2 DÉBRIEFEZ AVEC D'AUTRES PERSONNES DE VOTRE ORGANISATION

Lorsque l'équipe a fourni une évaluation équitable du produit, il faut élargir le public afin d'inclure toutes les personnes de l'organisation qui s'y intéressent. Ce groupe devra disposer d'un aperçu des discussions que vous avez eues en équipe et des compromis que vous avez faits. N'oubliez pas que les stories que vous avez transformées en logiciel fonctionnel sont comme des roches grossières issues d'une vision plus large du produit fini, alors que les personnes extérieures à l'équipe s'attendent peut-être à voir le produit fini. Elles vont donc probablement souligner ce qui manque, parce qu'elles ne participaient pas aux séances de planification où vous avez décidé de ce qu'il fallait reporter. Attendez-vous à cela et aidez-les à comprendre comment les parties que vous venez de développer s'intègrent dans un projet plus important. Effectuez cela dans une *revue de produit avec les parties prenantes*.

Pour une revue réussie de produit avec les parties prenantes

Il y a probablement beaucoup d'autres personnes dans l'organisation qui sont intéressées par ce que vous avez réalisé et vous allez devoir donner de la visibilité à votre travail. Contrairement à votre équipe, ces autres personnes ne connaissent probablement pas les détails de ce que vous avez choisi de développer, ni le cadre dans lequel ils s'inscrivent au sein de la vision d'ensemble. Vous aurez donc besoin de relier ce que vous avez accompli avec ce que vous avez appris du produit. C'est aussi une excellente occasion d'apprendre de ces personnes et d'obtenir leur soutien.

Invitez tous ceux qui sont intéressés. Il s'agit d'une grande revue publique et toute personne intéressée est la bienvenue. Assurez-vous que la totalité de votre équipe soit présente. Le fait de voir d'autres réactions par rapport à ce qu'ils ont réalisé, qu'elles soient positives ou négatives, leur rappelle que ce qu'ils font est important.

Apportez de la nourriture et je vous promets que tout le monde aimera ce que vous avez à dire quand ils seront repus d'hydrates de carbone. Même les mauvaises nouvelles passent mieux avec des cookies ou des chouquettes.

Vous allez passer en revue deux catégories d'information : le travail de découverte et les stories que vous avez livrées.

Revue du travail de découverte

La revue de la découverte est essentielle. Le meilleur moment pour obtenir un feedback de la part des parties prenantes se situe avant d'avoir investi beaucoup de temps à développer quelque chose. Si vous leur présentez les véritables leçons tirées des réactions des clients et des utilisateurs face au logiciel, ils vont apprécier d'apprendre ce que pensent réellement leurs clients. La seule chose qui a prise sur l'opinion d'un cadre est un fait précis.

– Discutez brièvement de chaque opportunité que vous avez prise en compte : pour qui est-elle ? Pourquoi la développer ? Quels sont les résultats attendus en cas de succès ?

– Discutez et montrez le travail que vous avez fait pour comprendre le problème et la solution.

– Discutez et montrez les prototypes et les expériences que vous avez réalisées. Discutez des réactions de vos clients et vos utilisateurs face à votre solution.

Revue de la livraison

D'après mon expérience, les parties prenantes se concentrent sur ce que vous allez livrer aux clients et aux utilisateurs et à quel moment. C'est normal car ce n'est qu'après avoir livré une solution viable qu'on est en mesure d'observer des résultats réels. Ils seront intéressés par les progrès réalisés pour atteindre cet objectif.

Examinez le travail de livraison que vous avez terminé solution par solution. Pensez à la solution viable minimum comme un gros rocher qui est plus pertinent pour les parties prenantes.

Pour chaque solution :

– Faites une revue des clients cibles de la solution, des utilisateurs et des résultats. Il est bon de se rappeler pourquoi nous développons cela et ce que le succès signifie.

– Discutez et présentez les résultats des stories développées pour chaque solution. Les parties prenantes vont alors réagir et fourniront des commentaires. Si par chance ils ont formulé leurs remarques au cours du travail de découverte, leurs réactions à ce stade devraient être que tout semble bien aller.

– Discutez les stories de façon holistique. Si vous utilisez une stratégie comme la stratégie de *Mona Lisa*, vous devrez leur expliquer pourquoi le logiciel semble incomplet à ce stade. N'oubliez pas qu'ils voudront voir un centimètre carré d'un portrait complet et non pas l'équivalent logiciel d'une esquisse de la toile entière.

1. Cette discussion d'amélioration du processus est communément appelée *rétrospective*, et il existe de nombreuses approches pour bien les mener. Si vous souhaitez étudier une recette plus complète de ce type d'approches, lisez l'ouvrage d'Esther Derby et Diana Larsen, *Agile Retrospectives* (Pragmatic).

© Dunod – Toute reproduction non autorisée est un délit.

– Partagez avec eux vos progrès en vue d'obtenir une solution livrable. Quelle quantité de travail reste-t-il ? Qu'avez-vous appris au cours du développement de la solution qui aura une incidence sur le succès de sa mise en œuvre ?

Soyez prêt à écrire des stories pour de nouvelles opportunités, ou pour les modifications que vous devrez faire.

Il est possible que des personnes présentes dans la salle ne soient pas familières avec ce que vous développez et proposent des choses qui ne sont pas de bonnes idées. Avec patience et respect, rappelez-leur le public cible et l'impact attendu de la solution, et exprimez les raisons pour lesquelles ce qu'ils suggèrent pourrait être une bonne idée, mais ne cadre pas avec l'impact sur lequel vous êtes actuellement concentré.

Donnez de la visibilité à votre travail pour tous les membres de votre entreprise car cela va les aider à rester motivés à propos de ce vous faites et apprenez.

18.3 SUFFISAMMENT DE LOGICIEL

Je suis confiant quand j'utilise un produit que j'aime, mais dont je n'apprécie pas tous les petits détails et les décisions de conception. En fait, si ça fonctionne vraiment bien, je ne fais guère attention au produit. Ainsi, je ne remarque pas comment mon téléphone mobile perd et rétablit la connexion Internet. Je ne remarque pas non plus quand je change la position de quelque chose dans l'application mobile de mon logiciel de gestion des tâches que la version web semble immédiatement se synchroniser. Mais ce sont des qualités importantes et je le remarquerais si elles n'étaient pas là. Votre équipe a travaillé sur de nombreux détails, mais curieusement vous ne souhaitez pas que les utilisateurs les remarquent.

Vous en apprendrez le maximum de la part des parties prenantes de votre organisation, des clients qui achètent vos produits et des utilisateurs individuels quand vous leur présenterez suffisamment de logiciel pour qu'ils puissent voir clairement comment cela va les aider à atteindre un de leurs objectifs.

Pour les parties prenantes

- Suffisamment de logiciel peut être l'ajout d'une fonctionnalité essentielle pour acquérir de nouveaux clients. Ou bien cela peut être des informations sur ce que vous avez appris à propos des détails qui doivent figurer dans cette fonctionnalité pour qu'elle soit attractive.

Pour les clients

- Suffisamment de logiciel peut être l'ajout d'une fonctionnalité qui représentera une réelle plus-value quand leur organisation commencera à utiliser le nouveau logiciel.

Pour les utilisateurs

- Suffisamment de logiciel peut être l'ajout d'un logiciel qui leur permet d'atteindre un de leurs objectifs à l'aide de votre produit.

Si vous avez bien réalisé votre processus de décomposition de blocs, vous vous êtes retrouvé avec de nombreuses petites parties qui peuvent être développées, chacune de ces parties permettant à votre équipe d'apprendre quelque chose. Mais ces petites parties ne sont probablement pas assez pertinentes pour les autres groupes.

Dans mon esprit, je me représente des petits morceaux de logiciels que nous empilons comme des briques de LEGO. Je charge toutes ces briques sur une vieille balance, du genre de celles qui ont deux plateaux'. Je pèse cette pile de briques de logiciel en l'étalonnant avec une brique LEGO plus grosse qui représente suffisamment de logiciel pour permettre à un utilisateur d'accomplir une tâche ou d'atteindre un but ; suffisamment de logiciel pour les clients pour qu'ils considèrent cela comme une plus-value ; suffisamment de logiciel pour les parties prenantes pour qu'ils voient comment cela peut aider leur organisation à atteindre un objectif commercial. Lorsqu'il y a suffisamment de logiciel, il est temps de tester ce logiciel avec les utilisateurs, de le critiquer avec les clients ou les parties prenantes.

Votre équipe de proches collaborateurs a besoin d'examiner les résultats de chaque story pour apprendre et améliorer non seulement votre produit, mais aussi la façon dont vous planifiez et la façon dont vous travaillez ensemble. Quand vous recevez des commentaires de la part des autres groupes, soyez sensible à ce que représente « suffisamment de logiciel » pour ces groupes.

18.4 APPRENEZ DES UTILISATEURS

On peut être assez confiant quand on commence à développer et avoir le sentiment que l'on fait les choses proprement, mais pour rester confiant, il est important de tester le logiciel avec les utilisateurs.

Vous noterez que j'ai employé ici le mot *tester*. On n'apprend pas beaucoup des utilisateurs en leur faisant une démonstration et en leur demandant d'imaginer d'utiliser un logiciel et de décider si ça leur plaît. C'est un peu comme si on regarde une voiture neuve derrière la vitrine d'une salle d'exposition et que l'on tente de savoir si on aimerait la conduire. En testant votre logiciel, les utilisateurs pourront vraiment évaluer s'il est capable de résoudre leur problème. Votre équipe en apprendra davantage en les regardant utiliser votre logiciel. Si votre équipe a eu de bonnes conversations de stories, vous avez probablement parlé des utilisateurs, de leurs motivations et de la façon dont ils allaient utiliser votre produit. C'est le fait de les regarder utiliser le logiciel qui va vraiment valider vos hypothèses.

Quand vous avez suffisamment de logiciel pour permettre aux utilisateurs d'accomplir quelque chose de significatif, il est temps de faire des tests. Il n'est pas souhaitable de tester quelque chose de complètement nouveau, et il est préférable d'apporter des changements ou des améliorations à une fonctionnalité que votre produit possède déjà. Passez du temps avec les utilisateurs à les observer se servir de votre logiciel pour réaliser une véritable tâche.

© Dunod – Toute reproduction non autorisée est un délit.

18.5 APPRENEZ DES UTILISATEURS AU MOMENT DE LA RELEASE

Vous avez développé de petites quantités de logiciel et révisé en équipe chaque petit morceau. Vous avez régulièrement organisé des revues avec les parties prenantes, les clients qui vont acheter ou adopter votre produit et les utilisateurs qui vont s'en servir. Mais si vous vous souvenez du début de ce livre, ce n'est pas vraiment le logiciel notre objectif principal, mais l'impact, c'est-à-dire les services qu'il rendra après sa livraison et sa mise en service.

Lorsque vous sentez que vous avez développé suffisamment de logiciel et que vous êtes convaincu que vous obtiendrez l'impact attendu, alors il est temps de sortir le logiciel.

Je me représente une autre balance où s'empilent des briques du logiciel que j'ai testé avec les utilisateurs, en l'améliorant par itérations successives, et dont je suis maintenant convaincu qu'il peut être livré. Je vais équilibrer cette pile avec une autre brique encore plus grande qui représente également suffisamment de logiciel pour être livré et avoir du succès avec son public cible. Quand j'ai suffisamment de logiciel, il est temps de le livrer.

Vous aurez besoin de planifier si vous voulez apprendre à chaque release. S'il vous plaît, ne publiez pas votre release en attendant patiemment que vos clients et vos utilisateurs se plaignent. Ces plaintes font partie de l'impact du produit et sont souvent un indicateur tardif de ce que ressentent vraiment les utilisateurs et de la manière dont votre produit va réellement se comporter. Pour chaque release, discutez en équipe, comment vous allez observer les utilisateurs de votre produit pour voir s'ils obtiennent vraiment l'impact attendu. Discutez et décidez de la manière dont vous allez :

- Créer des indicateurs dans votre produit qui vont vous permettre de suivre l'utilisation des nouvelles fonctionnalités
- Planifier des moments pour observer les utilisateurs quand ils se servent de la nouvelle release

En équipe, discutez régulièrement de ce que vous avez appris et inspirez-vous de vos idées d'amélioration pour écrire d'autres stories. Certaines vous paraîtront suffisamment importantes pour être mises en œuvre tout de suite, alors que d'autres seront considérées comme des opportunités à ajouter à votre backlog d'opportunités.

18.6 L'OUTCOME DANS LES DÉLAIS PRÉVUS

Il y a certaines entreprises et certains logiciels qui permettent de sortir une release dès l'instant que nous avons suffisamment de logiciel. Mais pour un grand nombre de sociétés et de produits, voire la plupart, la release doit se faire selon un calendrier prévu. Si nous avons utilisé notre stratégie de développement de manière efficace, nous avons établi une fondation dans nos premières stories d'ouverture de partie, développé le produit à l'aide des stories de milieu de partie, et quand vient le temps de sortir la release, nous jouons nos stories de fin de partie.

Maintenant, je dois vous rappeler quelques vérités supplémentaires sur le développement de logiciels.

> Le logiciel n'est jamais vraiment fini.

Vous aurez terminé d'implémenter le logiciel pour chaque story prise en charge par votre équipe lors d'un cycle de développement court. Mais vous n'aurez probablement pas terminé chaque story que vous aviez imaginée au début du développement, ou que vous aviez identifiée au moment de l'apprentissage de chaque cycle. Cependant, si vous avez utilisé une stratégie de développement efficace, la qualité du logiciel sera optimale au moment de sa livraison.

> L'impact n'est jamais assuré.

Malgré tout le travail que vous avez fait pour valider que vous développez les bonnes choses, souvent les personnes qui utilisent votre produit ne se comportent pas comme prévu. Prévoyez d'apprendre à chaque nouvelle release. Envisagez d'effectuer des modifications en fonction de ce que vous avez appris.

> Les améliorations apportées après la release sont les plus précieuses.

Ce sont ces choses imprévisibles, que vous allez observer lorsque les utilisateurs commencent à adopter et à utiliser régulièrement votre logiciel, qui génèrent la plupart des idées. Si vous planifiez du temps pour vraiment mesurer et observer l'impact, vous serez récompensé par les gens qui aiment vraiment votre produit.

18.7 UTILISEZ UNE CARTE POUR ÉVALUER LA PRÉPARATION DE LA RELEASE

Vous allez achever votre release du produit story par story. Lorsque vous vous approcherez de la date à laquelle vous avez promis de livrer, et il y a *toujours* une date de livraison qui a été promise, posez-vous la question suivante pour chaque activité principale de l'utilisateur : « Si nous devions livrer maintenant, quelle note nous nous donnerions ? » Si vous utilisez un système de notes avec des lettres comme mes enfants à l'école, vous vous retrouverez avec un bulletin de notes pour votre produit.

Par exemple, si vous examinez un produit ou une feature avec cinq activités principales quelques semaines avant sa date de livraison promise et que vous avez un bulletin de notes comportant *A*, *A-*, *B +*, *D*, *B +*, il est souhaitable de prendre les semaines restantes pour mettre l'accent sur la partie du flux de travail de l'utilisateur qui est actuellement évaluée avec un *D*. Si en fin de compte votre release est évaluée

© Dunod – Toute reproduction non autorisée est un délit.

avec des A et des B, c'est plutôt bon. Bien sûr, n'obtenir que des A serait mieux, mais livrer à l'heure peut être encore plus important.

Stories étiquetées dans un backlog sous forme de carte pour afficher la progression

Lorsque vous vous approchez de la date de sortie de votre produit, travaillez en équipe afin d'évaluer l'état de préparation de la release. Je vous promets que tout le monde a envie de savoir où en sont les choses.

Ce livre est presque fini. Si vous l'avez lu jusqu'ici, vous avez peut-être une opinion sur l'état de préparation de sa release... Vous pouvez revenir à la table des matières et attribuer une note à chaque chapitre. Prenez une photo avec votre smartphone et je serais ravi si vous me l'envoyez.

Fin provisoire

Comme tout bon logiciel, ce livre n'est pas vraiment terminé. Tout au long de cet ouvrage, vous avez lu de nombreux exemples fournis par des gens que j'ai rencontrés et qui réalisent des choses formidables avec des stories et du story mapping. J'ai encore plein d'autres stories sur mon disque dur et cela me navre de ne pas avoir assez de temps pour les inclure dans ce livre.

Il y a aussi beaucoup d'autres détails sur les stories et les story maps dont je pourrais discuter, et je suis certain que vous avez des questions, sur l'utilisation des stories dans votre propre contexte, auxquelles je n'ai pas répondu. Et cela me contrarie au moment de clore ce livre.

En tant qu'ancien développeur, concepteur d'interface utilisateur et chef de produit, je peux confesser que j'ai rarement été heureux lors de la release d'un produit. Cela est dû au fait que je savais que l'on n'avait pas pu inclure de nombreux éléments et qu'il restait encore plein de petites choses qui auraient pu être améliorées si l'on avait disposé d'un peu plus de temps pour les peaufiner. Si vous vous souciez vraiment de ce que vous développez, je sais que vous connaissez ce sentiment.

Je vais ici répéter la citation de Léonard de Vinci que j'ai employée au chapitre 4 :

> Un chef d'œuvre n'est jamais terminé, il est seulement abandonné.

Je n'irai pas jusqu'à dire que ce livre est de l'art, mais je dois avouer que je l'ai arrêté alors qu'il y avait encore beaucoup à faire. Je partage avec vous la responsabilité de ce travail qui reste à faire en espérant avoir de vos nouvelles quand vous aurez découvert vos propres méthodes de travail collaboratif pour créer des produits de qualité.

...roid – Toute reproduction non autorisée est un délit.

Bibliographie

Adlin, Tamara, and John Pruitt. The Essential Persona Lifecycle: Your Guide to Building and Using Personas. Burlington: Morgan Kaufmann, 2010.

Adzic, Gojko. Impact Mapping: Making a Big Impact with Software Products and Projects. Surrey, UK: Provoking Thoughts, 2012.

Specification by Example: How Successful Teams Deliver the Right Software. Shelter Island: Manning Publications, 2011.

Armitage, John. "Are Agile Methods Good for Design," Interactions, Volume 11, Issue 1, January-February, 2004. http://dl.acm.org/citation.cfm?id=962352.

Beck, Kent. Extreme Programming Explained: Embrace Change. New York: Addison-Wesley Professional, 1999.

Beck, Kent, and Michael Fowler. Planning Extreme Programming. New York: Addison-Wesley Professional, 2000.

Cagan, Marty. Inspired: How to Create Products Customers Love. Sunnyvale: SVPG Press, 2008.

Cheng, Kevin. See What I Mean: How to Use Comics to Communicate Ideas. Brooklyn: Rosenfeld Media, LLC, 2012.

Cockburn, Alistair. Agile Software Development. New York: Addison-Wesley Professional, 2001.

Writing Effective Use Cases. New York: Addison-Wesley Professional, 2000.

Cohn, Mike. User Stories Applied: For Agile Software Development. New York: Addison-Wesley Professional, 2004.

Constantine, Larry L., and Lucy A.D. Lockwood. Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. New York: Addison-Wesley Professional, 1999.

Cooper, Alan. The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity. Indianapolis: Sams – Pearson Education, 2004.

© Dunod – Toute reproduction non autorisée est un délit.

Gothelf, Jeff. Lean UX: Applying Lean Principles to Improve User Experience. Sebastopol: O'Reilly Media, 2013.

Jeffries, Ron, Ann Anderson, and Chet Hendrickson. Extreme Programming Installed. New York: Addison-Wesley Professional, 2007.

Klein, Laura. UX for Lean Startups: Faster, Smarter User Experience Research and Design. Sebastopol: O'Reilly Media, 2013.

Ries, Eric. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Business, 2011.

Sy, Desiree. "Adapting Usability Investigations for Agile User-Centered Design," Journal of Usability Studies, Vol. 2, Issue 3, May 2007. http://www.upassoc.org/upa_publications/jus/2007may/agile-ucd.html.

Tom Demarco et al. Adrenaline Junkies and Template Zombies: Understanding Patterns of Project Behavior. New York: Dorset House, 2008.

Yates, Jen. Cake Wrecks: When Professional Cakes Go Hilariously Wrong. Kansas City: Andrews McMeel Publishing, 2009.

Index

A

activité 78
affinage de backlog 144
agile
  développement 15
  Manifeste 121
analyste métier 158
anti-pattern 156
apprentissage 49, 104, 126, 229
  à partir de la release 236
  à partir des utilisateurs 235
apprentissage validé 53, 127, 191, 198
atelier de stories 143, 155, 218
  animation 206
  mode d'emploi 207

B

backlog 19, 45, 49
  affinage 102, 144
  de produit 95, 173
  de release, décomposition 68
  nettoyage 225
  opportunités 141, 162
  piège 32
  visualisation du fonctionnement 219
Basecamp 221
Beck, Kent 13, 16, 91, 127
Blank, Steve 198
budget, gestion 62

C

Cardboard 117
carte de voyage narrative 177
chef de produit, responsabilité 152
client, évaluation 146
Cockburn, Alistair 61, 74, 115
collaboration
  documentation 118
  en vase clos 212
  grands groupes 212
colonne vertébrale 79
compréhension mutuelle 4, 23, 32, 46, 117, 173
conception
  en comité 152
  erreurs à ne pas commettre 197
confirmation 95
Confluence 114
Connextra 99
conversation 95, 102
  liste de contrôle 105
  opportunité 161
  pour décomposer les stories 137
copie d'écran 58
critères d'acceptation 95, 143

© Dunod – Toute reproduction non autorisée est un délit.

cycle de développement, planification 216
cycle de vie agile 139

D

débriefing 229
  avec des personnes externes 232
décideur 11
décomposition 24, 39, 213
découverte 142, 173
  activités 188
  apprentissage validé 191
  composition de l'équipe 154
  compréhension mutuelle 189
  définition 54
  étapes essentielles 174
  revue 233
define, Design Thinking 195
délais, respect 57
Design Studio 179
  mode d'emploi 180
Design Thinking 54, 193, 196
  étapes 194
développement
  agile 15
  classique 193
  estimation du temps 59
  limité par le temps 216
  stratégie 68
document, aide à la mémorisation 7
documentation 118

E

empathize, Design Thinking 195
epic 138
équipe
  communication 58
  de découverte 153
  taille idéale 153
estimation 62
  du temps de développement 59
évaluation
  après la release 148
  avec les parties prenantes 147
exigence
  mésinterprétation 2
  non fonctionnelle 3
expérience, limitation 54
expression des besoins 48
Extreme Programming 16, 99

F

Fabricant, Robert 10
fiche 111
  amélioration 112
  contenu 115
flux narratif 75, 77

G

go 141, 162
GreenHopper 228
grooming 102

I

idéation, Design Thinking 195
IDEO 194
impact 11, 37, 192, 237
  cible 38
  hiérarchisation 38
  maximisation 11
ingénierie, validation des problèmes 182
interface utilisateur, visualisation 178
interview 202
itération 51, 67
  planification 209

J

Jeffries, Ron 94
jeu du bon-mieux-meilleur 215
jeu du téléphone arabe 1
JIRA 114, 119
journey map 83, 167
journey mapping 167
JSTOR 201

L

Lean Startup 54, 127, 199
Lean Thinking 199
Levitt, Gary 18
littéral 2
livrable 132
livraison
  date 36
  revue 233
logiciel
  échec 192
  évaluation 145

M

Manifeste agile 15, 121, 127
matrice d'opportunité 163
  flux 165
minimisation 185
modèle 17
MVP *Voir* produit viable minimum

N

niveau d'objectif 74
no go 141, 163

O

Obama, Barack 219
Omnigraffle 179
opportunité 46, 141, 161
  conversation 161
  décomposition des tâches 174
  évaluation 162
  formulation 46
  matrice 163
outcome 9, 48, 236
  différences avec output 9
  maximisation 11
  spécifique 79
output 9
  différences avec outcome 9
  minimisation 11

P

partenaire de développement client 47
pattern zombie 103
périmètre
  inflation 35
persona 34, 39, 96, 171, 175
  création 176
  croquis 175
photo de vacance 108
photographie 118
Pivotal Tracker 221
planification 57, 127, 185, 231
PO *Voir* product owner
Post-it® 71
priorisation 186
priorités, hiérarchisation 186
problème, validation 46
product owner 45, 151
  vu comme un producteur 157
produit
  découverte 53
  développement 1
  minimum 51
  qualité 124
  revue en équipe 230
produit viable minimum 37, 41, 42
  expérimentation 53
profil d'organisation 177
prototypage 47
prototype, Design Thinking 196

Q

qualité 124, 231

© Dunod – Toute reproduction non autorisée est un délit.

R

radiateur d'informations 115
release 52
  produit viable minimum 37
  roadmap 37
  viable plus petite 39
rétrospective 232
  sprint 146
revue, avec les parties prenantes 232
revue de produit, mode d'emploi 230
revue de sprint 146, 230
Ries, Eric 198
risque 63, 68, 201
roadmap 38
  décomposition en releases 37

S

scénario utilisateur 47
scope creep 35
Scrum 50, 124, 146
séquence 23
solution
  cartographie 178
  prototypage 48
  valable, utilisable et faisable 152
solution viable minimum 42, 136, 141
spécifications 9, 12, 91
spike 143
sprint 209
  rétrospective 146
  revue 146
squelette fonctionnel 61
steel thread 61
story
  alternative 76
  amélioration des conversations 99
  atelier 143, 206
  comme outil généraliste 127
  comment 106
  conversations 94, 95
  décomposition 132, 135, 206, 224
  décomposition à l'aide de conversations 137
  découpage 131
  détails 143
  fiche 100, 115, 205
  histoire du concept 91
  inspection 124
  liste de contrôle 105
  modèle de fiche 100
  objectif 1
  organisation 75
  origine 93
  pattern 104
  pourquoi 105
  première discussion 46
  processus 94
  qui 105
  quoi 105
  réassemblage des blocs décomposés 225
  recette de cuisine 129
  responsabilité de l'écriture 151
  solution trop coûteuse 132
  tâches de livraison 130
  taille appropriée 135
  tests 95
  tradition orale 123
story map 178, 204
  actuelle 82
  colonne vertébrale 33, 50, 78
  décomposition en tranches 79
  dépendances 33
  évaluation de la préparation de la release 237
  exemple d'exercice 71
  exercice pratique 83
  exposition des risques 63
  pour visualiser les progrès 218
  tâches 80
  tâches utilisateur 73
story mapping
  étapes 85
  exemple chez SAP 86
  limitation 227
  opportunités 167
  pendant la livraison 218
  repérage des trous dans la story 34
storyboard 169, 179, 183
storytelling 123, 139
stratégie incrémentale 65
suivi 119

T

tâche
  décomposition 79
  utilisateur 72
test 201
test, Design Thinking 196
The Lean Startup 43, 198
The Learning Connection 28
thème 139
tracer bullet 61
Trello 102
triade 154
trois amigos, modèle 155

U

user story 99
user story mapping 86, 89
utilisateur 11
  évaluation 146
  expression des besoins 48

Y

Yates, Jen 2

Z

zombie, pattern 103