ÉQUIPE FEATURE

par Craig Larman et Bas Vodde

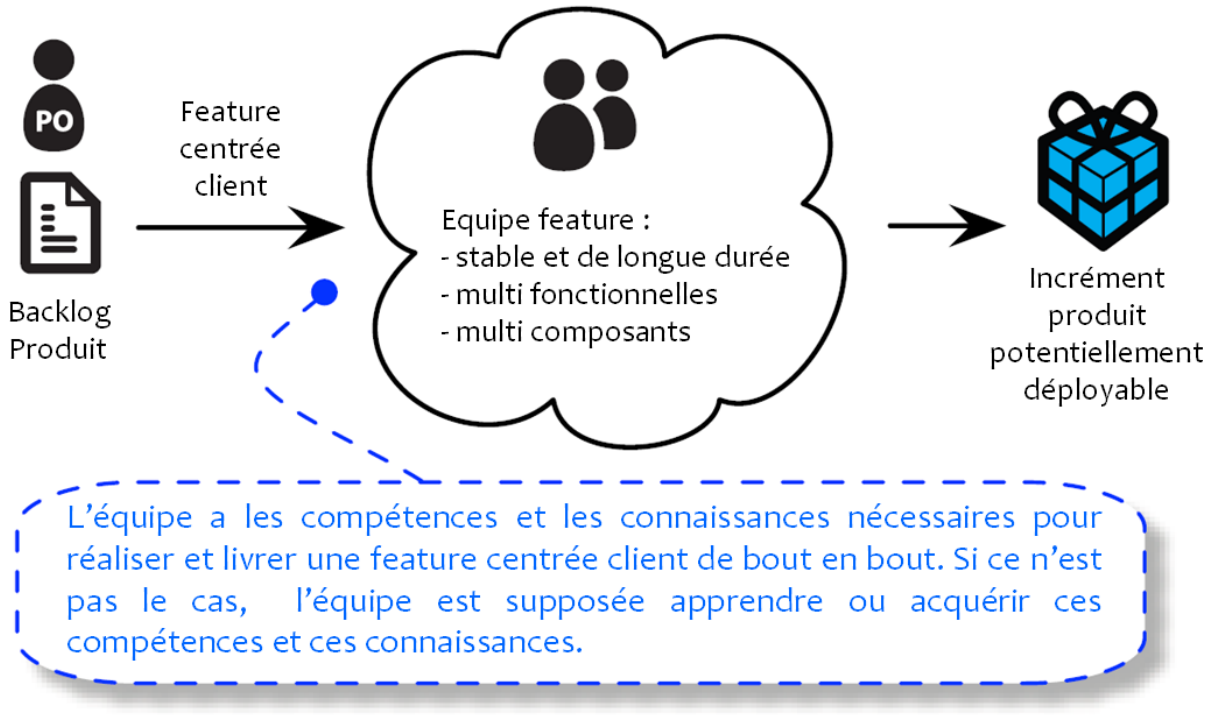
Version 1.2

Les Équipes Feature[1] *et les Domaines Fonctionnels*[2] *sont des éléments essentiels pour dimensionner le développement en mode agile et lean. Ces deux sujets sont analysés en profondeur dans les chapitres « Feature Team » et « Requirement Area » du livre « Scaling Lean & Agile Development : Thinking and Organizational Tools for Large Scale Scrum ». Cet article présente quelques idées clés que l'on peut également retrouver dans le livre « Practices for Scaling Lean & Agile Development : Large, Multisite, and Offshore Product Development with Large-Scale Scrum ».*

INTRODUCTION AUX ÉQUIPES FEATURES

Une **équipe feature**, comme illustrée à la Figure 1, est une équipe multi-composants, multi-fonctionnelles et pérenne[3] qui livre de nombreuses fonctionnalités bout-en-bout à ses clients, mais une à la fois.

Figure 1 : équipe Feature



[1] NdT : Feature Teams
[2] NdT : Requirement Areas
[3] Les membres d'une équipe features restent ensemble pendant des années, réalisant plusieurs features.

Traduit par Fabrice le 18 juin 2011

Les caractéristiques d'une équipe feature sont énumérées ci-dessous :

Équipe Feature
❑ pérenne, les membres de l'équipe restent ensemble et forment un « noyau dur » ❑ très performante, l'équipe réalise de nouvelles fonctionnalités au fil de l'eau ❑ multi-fonctionnelles et multi-composantes ❑ idéalement, colocalisée ❑ travaille sur une fonctionnalité centrée sur le client et de bout en bout, en passant à travers tous les composants et toutes les disciplines (analyse, programmation, tests, ...) ❑ composée de spécialistes se généralisant ❑ dans Scrum, habituellement 7 ± 2 personnes

Appliquer des pratiques d'ingénierie modernes, en particulier l'intégration continue, est essentielle lorsque vous choisissez d'être une équipe feature. L'intégration continue facilite la propriété partagée du code, ce qui est nécessaire lorsque plusieurs équipes travaillent en même temps sur les mêmes composants.

Un malentendu fréquent : chaque membre d'une équipe feature doit connaître l'ensemble du système. Pas forcément, parce que :

- ❑ L'équipe entière, et non chaque membre, doit avoir les compétences pour mettre en œuvre l'ensemble de la feature centrée sur le client. Il s'agit notamment de la connaissance des composants et de compétences fonctionnelles tels que les tests, la conception centrée utilisateur ou la programmation. Mais au sein de l'équipe, les gens continuent à être spécialisés ... de préférence dans de multiples domaines.
- ❑ Les features ne sont pas réparties au hasard sur des équipes features. Les connaissances et compétences actuelles d'une équipe sont prises en compte pour décider quelle équipe va travailler sur quelles features.

Dans une organisation basée sur des équipes feature, lorsque la spécialisation devient une contrainte... l'apprentissage commence.

Une organisation basée sur des équipes feature exploite les avantages de la vitesse générée par la spécialisation, ceci tant que la cartographie des exigences correspond aux compétences des équipes.

Mais lorsque les exigences ne correspondent pas aux compétences des équipes, l'apprentissage est « forcé », rompant ainsi la contrainte de sur-spécialisation.

Les compétences d'une équipe feature s'équilibrent entre spécialisation et flexibilité.

Traduit par Fabrice le 18 juin 2011

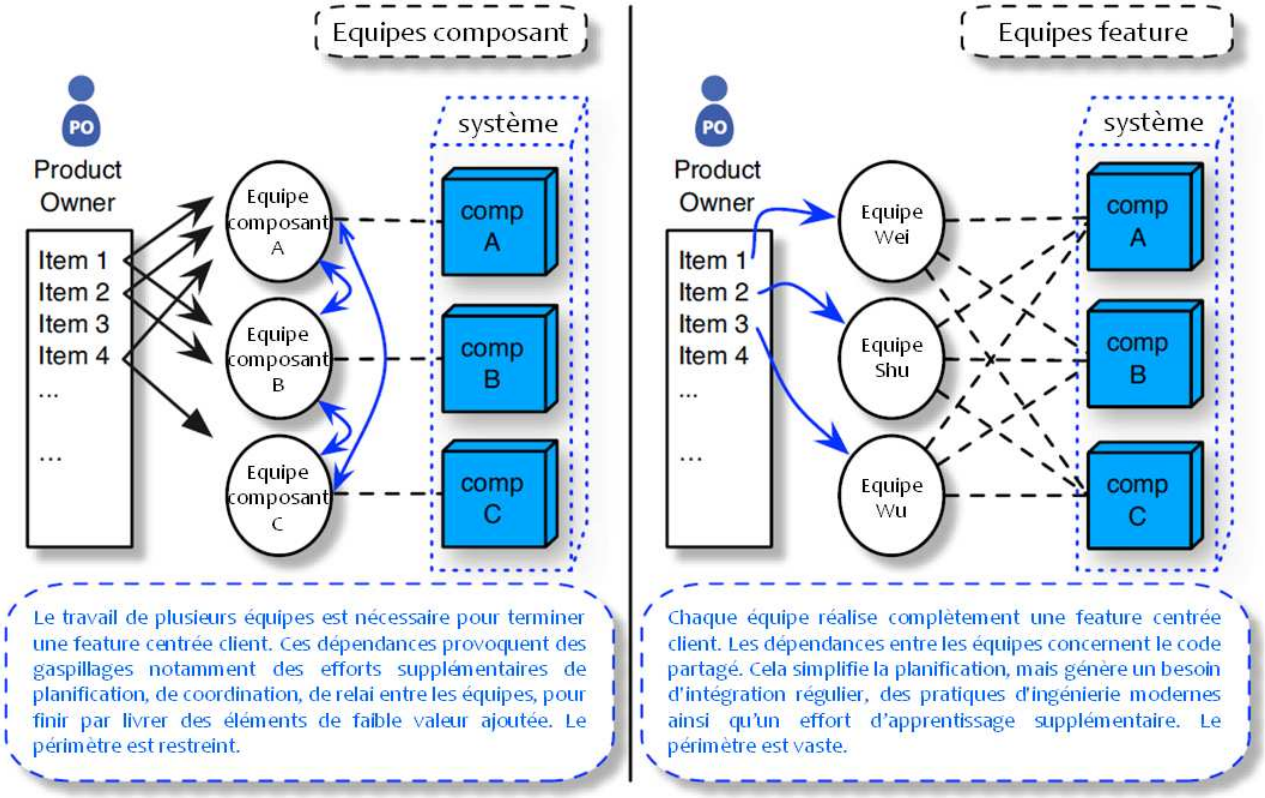
Le Tableau 1 et la Figure 2 suivantes mettent en évidence les différences entre les équipes feature et les équipes composants plus traditionnelles.

Tableau 1 : équipes feature vs composant

Équipe feature	Équipe composant
optimale pour livrer la valeur métier maximale (a)	optimale pour livrer le nombre maximum de lignes de code
se concentre sur des features à forte valeur ajoutée et la productivité du système	se concentre sur l'augmentation de la productivité individuelle en implémentant des features « faciles » de faible valeur
responsable de la feature de bout en bout et orientée client	responsable seulement partiellement d'une feature orientée client
démarche « moderne » d'organisation des équipes (b) — évite la loi de Conway	démarche traditionnelle d'organisation des équipes — respecte la loi de Conway (c)
conduit à se concentrer sur le client, à de la visibilité et à de plus petites organisations	conduit à « réinventer » le travail et à des organisations sans cesse croissantes
minimise les dépendances entre les équipes et augmente la flexibilité	les dépendances entre les équipes génèrent des efforts de planification supplémentaires (d)
met l'accent sur la multi-spécialisations	met l'accent sur l'hyper-spécialisation
propriété collective du code produit	propriété individuel du code
partage des responsabilités par l'équipe	responsabilités clairement portées par chaque individu
s'appuie sur du développement itératif	aboutit à un développement « cycle en V »
exploite la flexibilité ; apprentissage large et continu	exploite l'expertise existante ; au plus bas concernant l'apprentissage de nouvelles compétences
requiert des pratiques d'ingénierie qualifiées, les effets sont largement visibles	fonctionne avec des pratiques d'ingénierie bâclées, les effets restent localisés
fournit une motivation pour produire un code facile à maintenir et à tester	contrairement à la croyance, génère souvent un code de faible qualité dans le composant
apparemment difficile à mettre en œuvre	*apparemment* facile à mettre en œuvre

a) Une optimisation différente peut donner *l'impression* à l'équipe feature qu'elle est plus lente, d'un point de vue local.
b) Relativement « moderne » puisque les équipes feature ont une longue histoire dans le développement à grande échelle, par exemple chez Microsoft et Ericsson.
c) Mel Conway a observé cette structure indésirable en1968, il ne la *recommandait* pas, en fait c'était même tout le contraire.
d) Cet effort de planification supplémentaire est visible dans la plupart des « réunions de planification de releases » ou « trains de release » et génère un management inutile.

Figure 2 : équipes feature vs composant



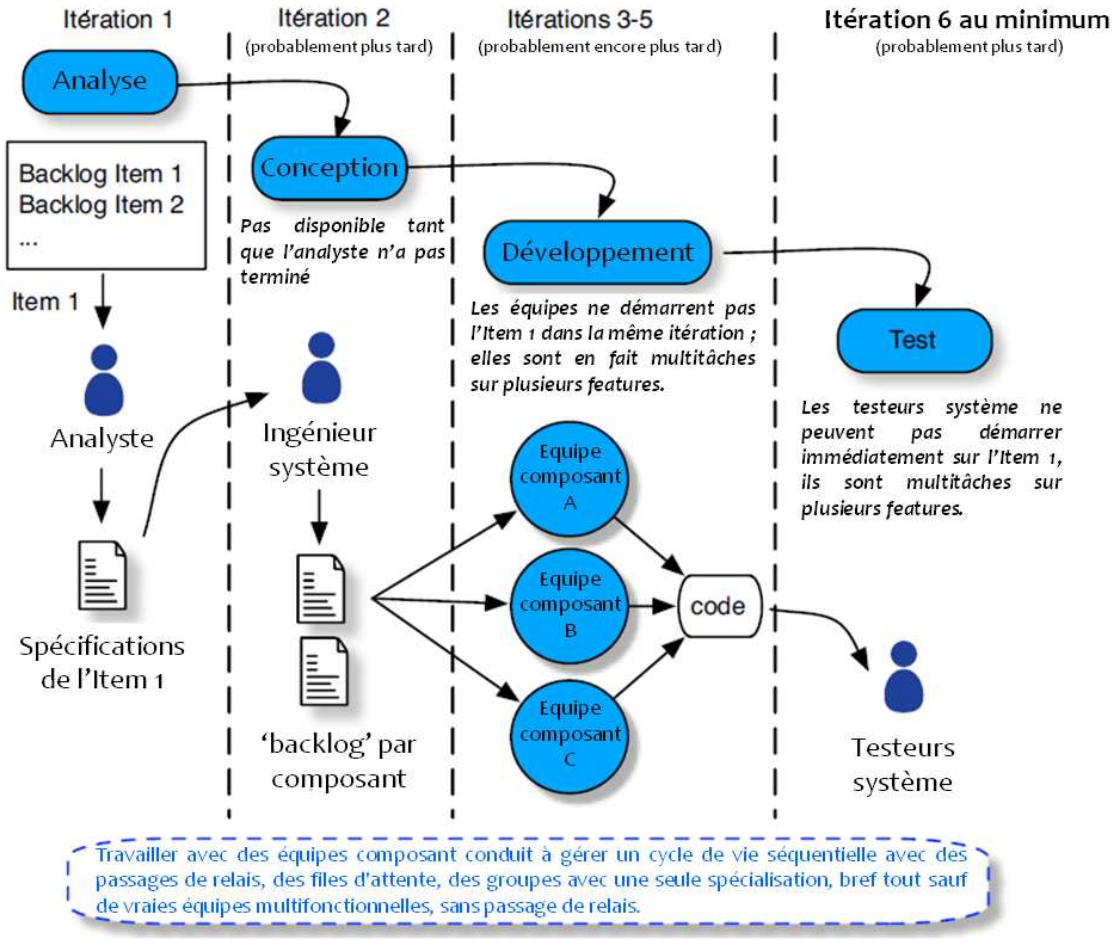
Le tableau ci-dessous résume les différences entre les équipes feature et les *projets* traditionnels ou *groupes feature*.

Tableau 2 : Équipes feature vs Projets feature

Équipe feature	Groupe/Projet feature
équipe stable dont les membres restent ensemble pendant des années et qui travaillent sur de nombreuses features	groupe temporaire de personnes créé pour une feature ou un projet
responsabilité partagée par l'équipe sur toutes les tâches	responsabilité individuelle des membres sur « leur » partie et basée sur la spécialisation
équipe auto-gérée	contrôlée par un chef de projet
génère une organisation horizontale simple (pas de matrice !)	génère une organisation matricielle avec des pools de ressources
les membres sont affectés à temps plein (100%) à l'équipe	les membres sont à temps partiel sur de nombreux projets en raison de leur spécialisation

La majorité des défauts des équipes composant sont explorés dans le chapitre « Feature Teams » du livre « Scaling Lean & Agile Development » ; la Figure 3 en résume quelques-uns.

Figure 3 : quelques défauts des équipes composant



Ce qu'on ne voit pas parfois, c'est que la structure d'une équipe composant renforce le développement séquentiel (modèle de la « cascade » ou cycle en V) en générant beaucoup de files d'attente, des tâches de taille variable, des niveaux élevés de TAF[4], de nombreux passages de relais, une augmentation du multitâches et l'affectation partielle des ressources.

CHOISIR DES ÉQUIPES COMPOSANT OU DES ÉQUIPES FEATURE ?

Une organisation basée uniquement sur des équipes feature est idéale d'un point de vue valeur ajoutée livrée et souplesse. Valeur et souplesse ne sont cependant pas les seuls critères pour construire une organisation, et de nombreuses organisations finissent par conséquent en mode hybride, plus particulièrement pendant la transition des équipes

[4] NdT : WIP = Work in Progress - Travaux à Faire

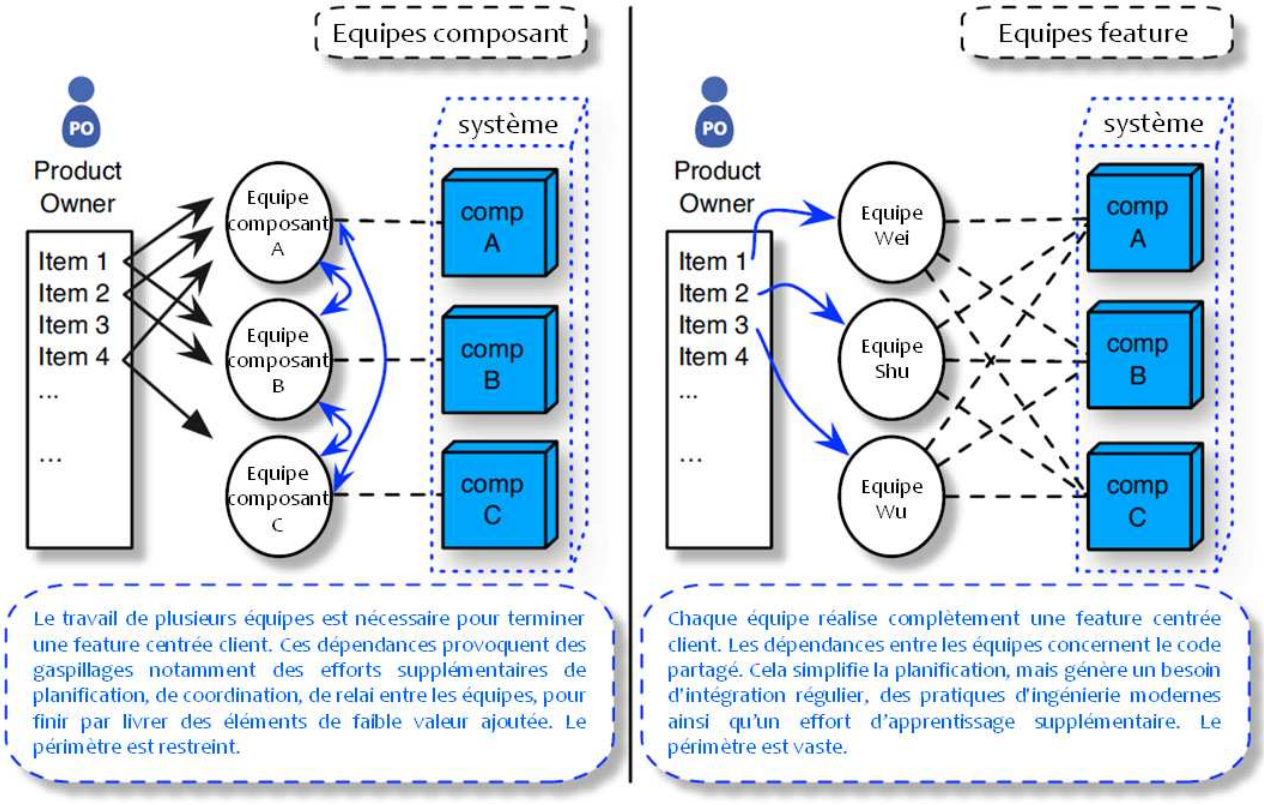
composant vers des équipes feature. Attention : les modèles hybrides ont les inconvénients des deux mondes et peuvent donc être... douloureux.

Un motif fréquemment exprimé en faveur d'une organisation hybride est la nécessité de construire des infrastructures, des composants réutilisables, ou de nettoyer le code écrit au sein des équipes composant. Mais ces activités peuvent aussi être réalisées une organisation basée sur des équipes purement feature, sans établir d'équipes composant permanentes. Comment ? En ajoutant les besoins d'infrastructure, de composants réutilisables ou de travail de nettoyage dans le Backlog Produit et en le donnant tel quel à une équipe feature existante comme s'il s'agissait de feature centrée client. L'équipe feature réalise ces travaux pendant un certain temps - aussi longtemps que le Product Owner le permet - puis retourne au développement de features centrées client.

ÉVOLUER VERS DES ÉQUIPES FEATURE

Différentes organisations exigent différentes stratégies de transition lors du passage d'équipes composant à des équipes feature. Nous avons l'expérience de nombreuses stratégies qui ont marché... et échoué dans un contexte différent. Une stratégie de transition sécurisée - mais lente - consiste à établir une seule équipe feature au sein de l'organisation basée sur des équipes composant. Une fois que cette équipe fonctionne bien, une deuxième équipe feature est formée. Cela continue progressivement selon un rythme adapté à l'organisation. Ceci est illustré à la Figure 4.

Figure 4 : transition graduelle d'équipes composant vers des équipes feature

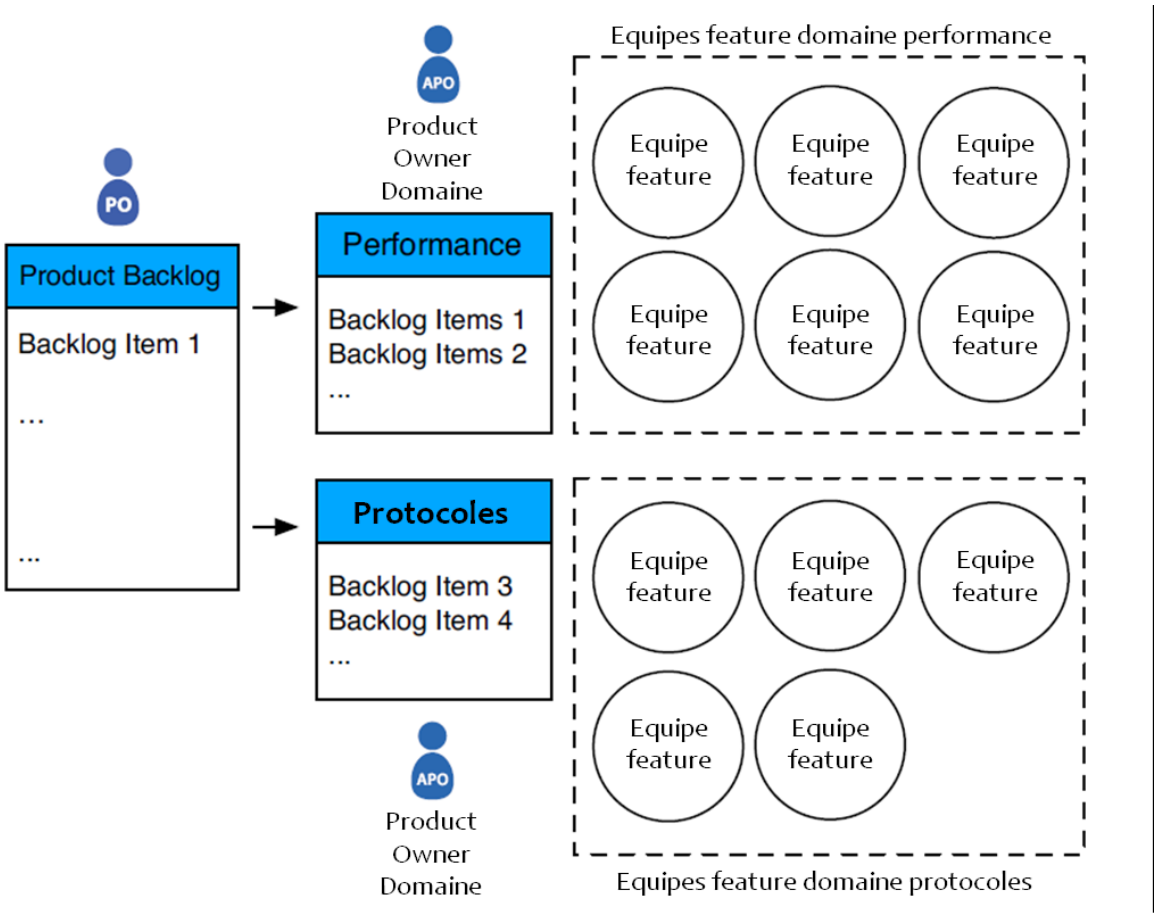


INTRODUCTION AUX DOMAINES FONCTIONNELS

On peut créer des équipes features sans problème mais quand leur nombre dépasse dix, soit environ une centaine de personnes, une structure supplémentaire est nécessaire. Les domaines fonctionnels fournissent cette structure et étoffent les concepts derrière les équipes feature. Un **domaine fonctionnel** est une catégorisation des exigences menant à des vues différentes du Backlog Produit.

Le Product Owner (PO) classe chaque item du Backlog Produit sous exactement une catégorie fonctionnelle, son domaine fonctionnel. Ensuite, il génère des vues différentes sur l'ensemble du Backlog Produit que l'on peut appeler un **Backlog Domaine**. Les Backlogs Domaines sont priorisés par **Product Owner Domaine** qui se spécialise dans une partie du produit et selon une perspective client. Chaque domaine fonctionnel comporte plsieurs équipes feature travaillant sur le Backlog Domaine, comme illustré à la Figure 5.

Figure 5 : domaines fonctionnels



Les domaines fonctionnels permettent de dimensionner correctement les équipes feature. Dimensionner en structurant les équipes en fonction de l'architecture du produit est appelé **domaines de développement**. Le Tableau 3 résume les différences.

Tableau 3 : domaines fonctionnels vs domaines de développement

Domaine fonctionnel	Domaine de développement
organisé autour d'exigences centrées client	organisé autour de l'architecture du produit
pas de propriété sur le code sous-système	propriété du code pour chaque sous-système
de nature temporaire ; devrait changer au cours de la durée de vie du produit, mais pas à chaque itération	tend à être plus figé sur la durée de vie du produit
on se concentre sur le client, en utilisant le langage du client	on se concentre sur l'architecture, en utilisant le langage technique

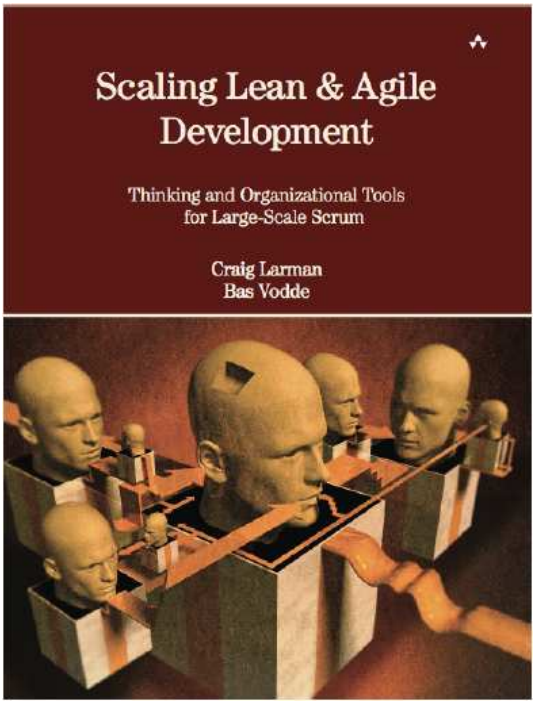
Finalement, un Product Owner Domaine est différent d'un Product Owner *délégué* qui travaillerait avec une ou deux équipes pour aider un Product Owner débordé. Un Product Owner Domaine a différentes responsabilités, se concentre et travaille avec (probablement) au moins *quatre* équipes, pas seulement une. Cela évite des situations d'optimisation localisée aux activités d'une seule équipe.

CONCLUSION

Les équipes feature sont des équipes stables qui travaillent sur des features complètes et centrées client. Ces équipes apportent des solutions vis-à-vis des habituelles optimisations locales et travaux de coordination supplémentaires liés aux organisations basées sur des équipes composant. Cependant, ces équipes features doivent elles aussi relever des défis.

Les différents domaines fonctionnels génèrent des vues centrées client sur la globalité du Backlog Produit et permettent donc de structurer et dimensionner les équipes feature à la taille voulue.

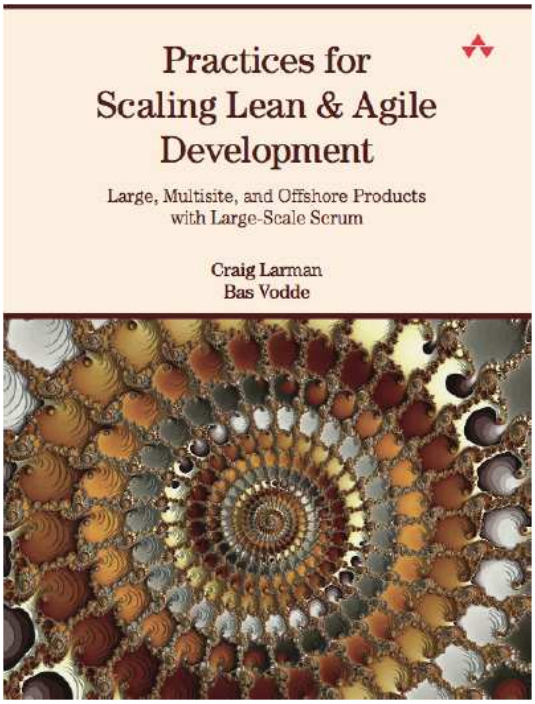
RÉFÉRENCES



Scaling Lean & Agile Development

Chapters :

- ❑ Introduction
- ❑ Systems Thinking
- ❑ Lean
- ❑ Queueing Theory
- ❑ False Dichotomies
- ❑ Be Agile
- ❑ Feature Teams
- ❑ Teams
- ❑ Requirement Areas
- ❑ Organization
- ❑ Large-Scale Scrum



Practices for Scaling Lean & Agile Development

Chapters :

- ❑ Large-Scale Scrum
- ❑ Test
- ❑ Product Management
- ❑ Planning
- ❑ Coordination
- ❑ Requirements
- ❑ Design
- ❑ Legacy Code
- ❑ Continuous Integration
- ❑ Inspect & Adapt
- ❑ Multisite
- ❑ Offshort
- ❑ - Contracts