UN EXEMPLE DE CHECKLIST POUR SCRUM MASTERS

Michael James
(mj4scrum@gmail.com)
14 septembre 2007, révisé le 24 juillet 2012
Traduction du 12 août 2015 par Maxime Sinclair

Un Facilitateur à Temps Complet ?

Un Scrum Master correct peut gérer deux ou trois équipes en même temps. Si vous vous contentez d'organiser des réunions, de vérifier le respect des délais impartis (timebox – NdT), et de supprimer les obstacles explicitement rapportés, vous pourrez remplir ce rôle à temps partiel. L'équipe pourra quand même dépasser les attentes habituelles, pre-Scrum de votre organisation, et il n'arrivera probablement rien de catastrophique.

Mais si vous pouvez imaginer une équipe qui prend du plaisir à accomplir des choses que personne ne pensait possible auparavant, au sein d'une organisation transformée, alors considérez-vous comme un *grand* Scrum Master.

Un grand Scrum Master ne peut gérer qu'*une* équipe à la fois.

Nous recommandons un Scrum Master dédié par équipe d'environ sept personnes, particulièrement au démarrage.

Si vous n'avez pas découvert tout le travail qu'il y a à faire, mettez-vous à l'écoute de votre Product Owner, de votre équipe, des pratiques de développement de votre équipe, et de l'organisation externe à votre équipe. Bien qu'il n'y ait pas une recette unique pour tout le monde, j'ai relevé des points typiques que les Scrum Masters négligent. Remplissez chaque case avec √, Δ, ?, ou N/A, comme décrit en dernière page.

Partie I - Quelle est l'efficacité de mon PO ?

Les Scrum Masters améliorent l'efficacité des Product Owners en les aidants à trouver les moyens de gérer le Backlog et le plan de livraison. (A noter : seul le Product Owner peut prioriser le backlog.)

- ☐ Est-ce que le Backlog est priorisé selon ses dernières réflexions ?
- ☐ Est-ce que toutes les exigences et les désirs de toutes les parties prenantes sont présents dans le Backlog ? Rappel : le backlog est *émergent*.
- ☐ La taille du backlog est-elle raisonnable ? Pour garder un nombre d'items raisonnable, conserver les items détaillés vers le haut, et les idées générales (epics – NdT) vers le bas. Il est contre-productif d'analyser trop en profondeur ce qui n'est pas au sommet du Backlog. Vos exigences vont évoluer tout le long des discussions régulières que vous aurez avec les parties prenantes / clients à propos du produit en cours de développement.
- ☐ Les fonctionnalités (spécialement celles proche du sommet du Backlog) pourraient-elles être mieux exprimées sous la forme de user stories indépendantes, négociables, ayant de la valeur, estimables, suffisamment petites, et testables ?
- ☐ Avez-vous expliqué à votre PO ce qu'est la *dette technique* et comment l'éviter ? Une pièce du puzzle peut consister à ajouter le test automatisé et le refactoring à la définition du "fini" pour chaque item du backlog.
- ☐ Le backlog est-il un *radiateur d'information*, clairement visible pour l'ensemble des parties prenantes ?

☐ Si vous utilisez un outil électronique de gestion du backlog, tout le monde sait-il l'utiliser facilement ? Les outils de gestion électroniques peuvent se transformer en *réfrigérateurs d'information* privés du rayonnement actif du Scrum Master.
☐ Aidez-vous à diffuser l'information en affichant à la vue de tous des rapports imprimés ?
☐ Aidez-vous à diffuser l'information en créant de grands graphiques bien visibles ?
☐ Avez-vous aidé votre Product Owner à grouper les items du backlog par livraisons ou par priorités ?
☐ Est-ce que tout le monde sait si le plan de livraison est toujours réaliste ? Pendant les réunions de revue de Sprint, vous pourriez montrer à tous le graphique de Product/Release Burndown indiquant les items qui ont été certifiés "finis". Les graphiques montrant le taux de réalisation des items du Backlog et celui d'ajout des nouveaux items permettent de révéler au plus tôt les dérives de périmètre/planning.
☐ Est-ce que votre Product Owner ajuste le plan de livraison après la dernière revue de sprint ? La minorité des Product Owners qui livrent des produits correctement testés à temps *re-planifient* la version à chaque Sprint. Ceci implique probablement de reporter certains items à des versions futures au profit d'items plus importants découverts en cours de route.

Partie II - Quelle est l'efficacité de mon Equipe ?

Même si vous êtes encouragé à montrer l'exemple en collaborant avec les membres de l'équipe dans leur travail, il y a un risque que vous vous perdiez dans les tâches techniques. Réfléchissez à vos principales responsabilités envers l'équipe:

☐ Est-ce que votre équipe est en état d'*expérience optimale*[1] ? Voici quelques signes qui ne trompent pas :
- Des objectifs clairs (les attentes et les règles sont correctement perçues et les objectifs sont atteignables, alignés avec les compétences et capacités de chacun).
- Concentration et ciblage, un haut niveau de concentration sur un périmètre limité.
- La perte de la conscience de soi, résultat de la fusion entre l'action et la réflexion.
- Des retours directs et immédiats (les succès et les échecs sont immédiatement repérés, ainsi le comportement de l'équipe peut être ajusté en fonction).
- Equilibre entre capacité à faire et défi (l'activité n'est ni trop facile, ni trop difficile).
- Un sentiment de contrôle de soi et de l'environnement.
- L'activité est en soi gratifiante, le travail se fait sans effort.

☐ Est-ce que les membres de l'équipe semblent s'apprécier, plaisantent entre eux, et fêtent les succès de chacun ?
☐ Est-ce que chaque membre de l'équipe se sent responsable d'un haut niveau de qualité, et stimule les autres pour progresser ?
☐ Y a-t-ils des risques/opportunités que l'équipe ne discute pas car ses membres ne se sentent pas à l'aise ?
☐ Avez-vous essayé différentes formes et différents lieux pour réaliser vos Rétrospectives de Sprint ?
☐ L'équipe est-elle encore concentrée sur les objectifs du Sprint ? Vous pourriez organiser un point de mi-Sprint pour re-revoir les critères d'acceptation des items du backlog de l'itération.
☐ Le tableau des tâches reflète-t-il ce que l'équipe fait réellement ? Méfiez-vous de la "matière noire" des tâches cachées et des tâches supérieures à un jour de travail. Les tâches qui ne sont pas liées aux objectifs du Sprint sont des obstacles à la réalisation de ces objectifs.

1Aussi nommée expérience du flow, https://fr.wikipedia.org/wiki/Flow_%28psychologie%29

- ☐ Votre équipe est-elle composée de 3 à 9 personnes avec un éventail suffisant de compétences pour développer un incrément du produit potentiellement livrable ?
- ☐ Le tableau des tâches de votre équipe est-il à jour ?
- ☐ Les outils d'autogestion de l'équipe (tableau des tâches, Sprint Burndown Chart, listes des obstacles, etc) sont-ils bien visibles, pratiques à utiliser pour l'équipe ?
- ☐ Ces outils sont-ils correctement protégés des fouineurs ? Une attention excessive portée sur l'activité quotidienne de l'équipe par des personnes externes peut nuire à la transparence interne et l'autogestion.
- ☐ Les membres de l'équipe se portent-ils volontaires pour réaliser les tâches ?
- ☐ Le remboursement de la dette technique se traduit-il explicitement sous forme d'items du backlog, rendant progressivement le code plus agréable à maintenir ?
- ☐ Les membres de l'équipe abandonnent-t-ils leur intitulé de poste à la porte du bureau, préférant être collectivement responsables de tous les types de travaux engagés (test, documentation utilisateur, etc) ?

Partie III - Que valent vos pratiques d'ingénierie ?

- ☐ Le système que vous développez possède-t-il un bouton "Lancer les tests" permettant à n'importe qui (de l'équipe ou d'une autre équipe) de détecter simplement s'il a causé une régression (cassé une fonctionnalité qui marchait) ? Généralement, ceci est réalisé en utilisant un framework xUnit (JUnit, NUnit, etc).
- ☐ Vos tests automatisés présentent-ils un juste équilibre entre les tests systèmes de bout en bout (ou "tests fonctionnels") et les tests unitaires ?
- ☐ L'équipe écrit-elle les tests systèmes et unitaires dans le même langage de développement que celui utilisé pour développer le système ? Les langages de script propriétaires et les outils de capture de transactions que seule une partie de l'équipe sait maintenir sont un frein à la collaboration.
- ☐ L'équipe a-t-elle découvert la zone grise mais bien utile située entre les tests systèmes et les tests unitaires ?
- ☐ Votre serveur d'intégration continue émet-il une alarme sonore lorsque quelqu'un provoque une régression ? Le délai d'apparition de cette alarme peut-il être réduit à quelques heures ou minutes ? ("Les builds de nuit sont pour les mauviettes." – Kent Beck)
- ☐ *Tous* les tests sont-ils bien inclus dans le résultat du serveur d'intégration continue ?
- ☐ Les membres de l'équipe ont-ils découvert les joies de la conception continue et du refactoring permanent, en remplacement du document de Conception Générale et Détaillée ? Le refactoring a une définition stricte : modifier la structure interne du système sans changer son comportement externe. Le refactoring doit avoir lieu plusieurs fois par jour, à chaque fois qu'il y a du code dupliqué, une logique conditionnelle complexe (détectable par des indentations excessives ou des méthodes trop longues), des identifiants mal nommés, un couplage excessif entre objets, etc. Le refactoring se peut se faire en toute confiance que grâce aux tests automatisés. Négliger le refactoring rend le produit difficile à faire évoluer, notamment parce qu'il est difficile de trouver des bons développeurs qui veulent travailler sur du mauvais code.
- ☐ La définition du "fini" pour les items du backlog intègre-t-elle la couverture complète des tests automatisés et le refactoring du code ? Apprendre le développement piloté par les tests (TDD) vous y aidera.

☐ L'équipe pratique-t-elle la programmation en binôme ? La programmation en binôme peut améliorer la maintenabilité du code et réduire fortement le nombre d'anomalies. Cette collaboration étroite peut générer des réticences et sembler parfois moins productive (si on mesure en nombre de lignes de code plutôt que par fonctionnalités livrées). Montrez l'exemple en initiant des journées de travail en binôme avec les membres de l'équipe. Certains d'entre eux commenceront à préférer cette méthode de travail.

Partie IV – Qu'elle est l'efficacité de mon Organisation ?

☐ Les équipes communiquent-elles suffisamment entre elles ? Le "Scrum de Scrums" n'est qu'une façon d'y parvenir, et pas forcément la meilleure.
☐ Les équipes peuvent-elles produire de manière autonome des fonctionnalités livrables, qui éventuellement traversent différents blocs de l'architecture du système.
☐ Vos Scrum Masters se rencontrent-ils pour travailler sur la liste des obstacles ?
☐ Quand cela est nécessaire, les obstacles organisationnels sont-ils placardés dans le bureau du directeur des développements ? Le coût peut-il être quantifié en dollars, en délai de livraison, en perte de qualité, en perte de clients ? (Mais apprenez des erreurs de Ken Schwaber : "Un Scrum Master mort est un Scrum Master inutile.")
☐ Votre société est-elle une des rares qui encouragent des évolutions de carrières compatibles avec les objectifs de vos équipes ? Répondez "non" s'il y a des incitations à faire de la programmation ou de l'architecture au détriment des tests, de l'automatisation des tests ou de la documentation utilisateur.
☐ Votre société a-t-elle été reconnue par la presse ou une organisation indépendante comme l'une des meilleures où travailler, ou comme un leader dans votre domaine ?
☐ Travaillez-vous à créer une *organisation apprenante* ?

Conclusion

Si vous pouvez cocher la plupart de ces points et qu'il vous reste du temps libre dans la journée, n'hésitez pas à me contacter.

Il n'y a pas de recette toute faite pour susciter le génie humain. Cette liste peut, ou pas, vous aider dans votre contexte.

Quand vous commencez à réaliser que vous pouvez faire la différence, vous pouvez avoir peur de le faire. C'est le signe que vous êtes sur la bonne voie.

FICHE D'OBSTACLE ORGANISATIONNEL

PROBLÈME APPARENT :

CAUSE RACINE (UTILISEZ 5 FOIS « POURQUOI ? ») :

CONSÉQUENCE MÉTIER :

CONSÉQUENCE ÉMOTIONNELLE :

DEMANDE CLAIRE ET RÉALISABLE :

FICHE D'OBSTACLE ORGANISATIONNEL

PROBLÈME APPARENT :

CAUSE RACINE (UTILISEZ 5 FOIS « POURQUOI ? ») :

CONSÉQUENCE MÉTIER :

CONSÉQUENCE ÉMOTIONNELLE :

DEMANDE CLAIRE ET RÉALISABLE :

INSTRUCTIONS

Si vous avez récupéré cette checklist en tant qu'outil de formation et que votre organisation tente de mettre en œuvre quelque chose ressemblant à Scrum, appliquez-la à ce que vous observez. Cochez chacun des points avec l'un des symboles suivants :

√ (pour « déjà fait correctement »)

Δ (pour « pourrait être amélioré et je sais par où commencer »)

? (pour « pourrait être amélioré, mais comment ? »)

N/A (pour « non-applicable » ou « n'apporterait pas de bénéfice »)

Ou, si votre organisation n'a encore rien tenté qui ressemble à Scrum, utilisez la notation suivante :

√ (pour « déjà fait correctement » ou « serait facile à faire »)

Δ (pour « serait un défi et je sais par où commencer »)

? (pour « serait un défi et j'ignore par où commencer ? »)

N/A (pour « non-applicable » ou « n'apporterait pas de bénéfice »)

Lorsque que vous avez coché tous les points de la checklist, remplissez 2 à 6 Fiches d'Obstacle Organisationnel, que ces obstacles soient ou non issus de la checklist. Choisissez des obstacles que vous pourrez surmonter avec au moins 1 % de chance.