



KANBAN DANS LE MONDE DE L'IT

Commençons par nous présenter !

- ▶ Connaissez-vous l'Agile et Kanban ?
- ▶ Avez-vous déjà travaillés grâce à ces outils ?

Sommaire

- 1) Introduction à Kanban
- 2) Visualiser le travail
- 3) Limiter le travail en cours
- 4) Gérer les flux
- 5) Classes de services, Planifier et estimer
- 6) Amélioration continue
- 7) Cadences et Scrumban

1.

Introduction à kanban

“

“Stop Starting, Start Finishing”
David J. Anderson

”

Qu'est-ce que kanban ?

Kanban est un **meta-process** s'appliquant pour **n'importe quel process** avec lequel vous travaillez.

Vous pouvez commencer dès maintenant à l'appliquer sans modifier votre façon de travailler :
vous pourrez visualiser votre process et commencer à l'améliorer rapidement !

Point forts de Kanban

- ▶ Les bouchons de production deviennent visibles
- ▶ C'est un premier pas vers le travail en mode "Agile" !
- ▶ Donne une possibilité de travailler en Agile sans être coincé par des itérations (Sprints), en commençant où vous en êtes
- ▶ Facile à mettre en place
- ▶ Se propage naturellement dans l'organisation (management visuel)
- ▶ Très efficace en maintenance (TMA) ou bugs, lorsque les changements sont extrêmement rapides, quotidiens !
- ▶ On commence enfin à clôturer nos tâches !

Qu'est-ce que kanban ?

Kanban veut littéralement dire :
Carte Visuelle

Toyota a utilisé des
"Kanban" afin de
limiter le "travail en cours"
(Work in Process) pour
sa production de voitures.



Qu'est-ce que kanban ?

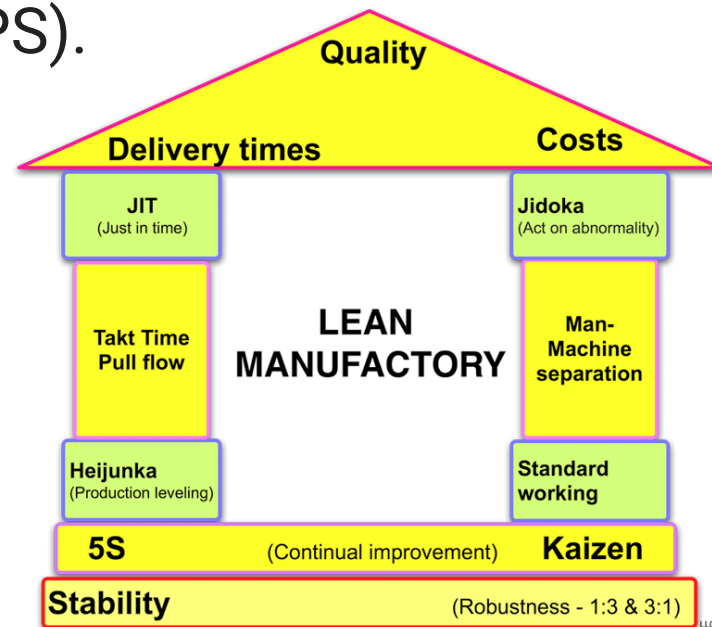
Kanban, en tant que concept, est né avec Toyota, par le Toyota production System (TPS).

Les Européens et Américains ont importé ce concept sous la forme du

“Lean Production System”.

Mais **kanban \neq Lean !**

Le concept de Kanban est le créateur du Lean !



Qu'est-ce que kanban ?

3 types de "kanban" :

- ▶ La méthode Kanban :
 - ▶ *Une méthode pour créer des changements évolutionnaires dans une entreprise (David J Anderson).*
- ▶ kanban :
 - ▶ *Un système de management visuel des processus qui dit quoi produire, quand produire, combien produire.*
- ▶ Le système Kanban :
 - ▶ *Un système créé pour vérifier le travail en cours (Work in process).*

Les 6 principes de Kanban

Visualiser

Visualiser le travail en cours de manière simple, votre process, étape par étape, afin de mieux le comprendre.

Limiter le travail en cours

Etablir délibérément une limite du nombre de tâche à faire en même temps.

Gérer les flux

Comprendre comment le flux de travail fonctionne, et repérer les imperfections, les bouchons et les exceptions.

Expliciter les normes du process

Ecrire et discuter constamment des étapes des process utilisés.

Créer des boucles d'amélioration

Pour aider à améliorer le flux de travail, il faut créer des boucles d'amélioration, comme des rétrospectives.

S'améliorer collaborativement, évoluer expérimentalement

Utiliser des modèles et méthodes scientifiques (Lean, théories des contraintes) pour améliorer ses process.

Les 4 principes selon David J Anderson

Commencer par ce que vous faites actuellement

Respecter le processus actuel, les rôles, les responsabilités et les titres

Accepter d'appliquer les changements évolutifs et augmentés

Leadership à tous les niveaux


Petit point avant de continuer... L'Agile Manifesto.

Nous découvrons de meilleures approches pour faire du développement logiciel, en en faisant nous-même et en aidant les autres à en faire. Grâce à ce travail nous en sommes arrivés à préférer et favoriser :

- **Les individus et leurs interactions** plus que les processus et les outils.
- **Un logiciel qui fonctionne** plus qu'une documentation exhaustive.
- **La collaboration avec les clients** plus que la négociation contractuelle.
- **L'adaptation au changement** plus que le suivi d'un plan.

Cela signifie que bien qu'il y ait de la valeur dans les items situés à droite, notre préférence se porte sur les items qui se trouvent sur la gauche.

Agile Manifesto



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

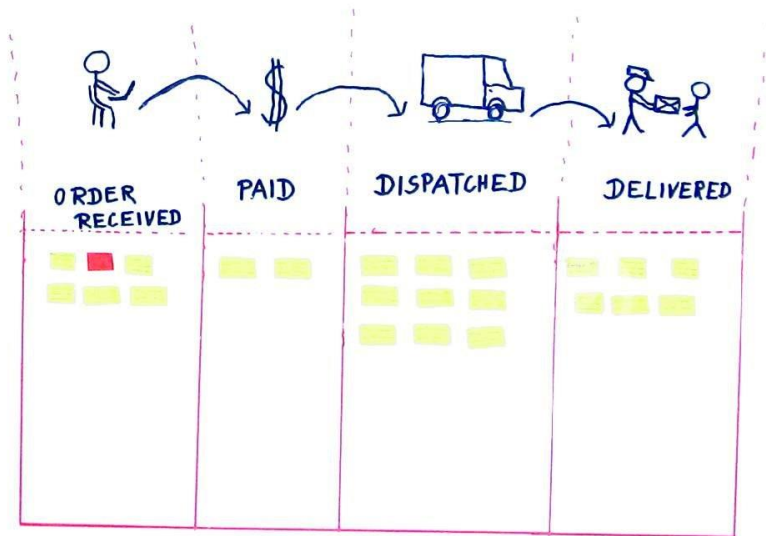
That is, while there is value in the items on the right, we value the items on the left more.

Les 12 principes du Manifeste agile

- Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
- Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.
- Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
- Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
- Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
- La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
- Un logiciel opérationnel est la principale mesure d'avancement.
- Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
- Une attention continue à l'excellence technique et à une bonne conception renforce l'agilité.
- La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
- Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
- À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

2. Visualiser le travail

La transparence : élément clé de l'Agile, et donc de kanban !



Les radiateurs d'information

Radiateur d'information :

Un écran et / ou courbe permettant de voir, en un clin d'oeil, comment le travail avance. A utiliser pour vous, et pour les parties prenantes. Il doit être simple à utiliser (ex : colonnes + post-it), et ne mettez que des informations utiles dessus.



Attention aux outils numériques : ce sont des **frigo d'information** car l'information est cachée à l'intérieur. Il faut aller la chercher manuellement (ex : Jira).

Point quotidien

Un tableau kanban est un excellent prétexte à la discussion lors d'un point quotidien.

Recette d'un bon point quotidien :

- ▶ Trouver un moment opportun pour toute l'équipe et stable,
- ▶ Réaliser des points de 15 minutes maximum, debout,
- ▶ Les discussions qui n'impliquent pas la majorité des membres de l'équipes sont réalisées après le point quotidien,
- ▶ Rester factuel avant tout !



Systèmes de queues

Prenons un exemple simple de process IT :

- ▶ A faire,
- ▶ Développement,
- ▶ Développement réalisé / Prêt à être testé,
- ▶ Test,
- ▶ Fini.

Exemple de queue : “Prêt à être testé”. Une tâche dans cette colonne prévient les testeurs que cette tâche a été finie par les développeurs. C'est à vous de définir les queues qui vous semblent logiques !

Critères d'entrées et de sorties

Ecrire sur le tableau les critères d'entrées et / ou de sorties des étapes.

Exemple :

Pour une colonne "Analyser", les critères de sorties sont :

- ▶ Critères d'acceptation définis,
- ▶ Vérifié par le Product Owner.

Pour une colonne "Développement", les critères de sorties sont :

- ▶ Installé dans un environnement de test,
- ▶ Couverture de test > 85%,
- ▶ Vérifié par un autre développeur.

Qu'est-ce qu'une "carte de tâche" ?

Une carte devrait :

- ▶ Faciliter les décisions à prendre,
 - ▶ Aider à optimiser les résultats finaux, en mettant par exemple le type de tâche,
 - ▶ Être simple à lire pour tous.
- ▶ Exemples de types : story utilisateur, story technique, bug...


Les User Stories

Une “User Story” ou histoire utilisateur est :

- ▶ Une **C**arte
- ▶ Permettant de commencer des **C**onversations
- ▶ Avec une **C**onfirmation par des critères d'acceptation.

En tant qu'[utilisateur], je veux [un besoin] afin d'avoir [un b

Les User Stories ont été créées par l'eXtreme Programming (Mike Cohn, Kent Beck, etc...)



As a <role>
I want <goal>
So that <benefit>

Acceptance criteria:
...

Les User Stories

Bonne User Story :

En tant qu'admin, je veux des rapports et statistiques du site afin de faire des prédictions pour prioriser de futures fonctionnalités.

Mauvaises User Story :

Fixer SITE_USAGE_CUBE_INDEX74

Page 6 - A FAIRE

Point fort des User Stories : permettre de lancer des discussions sur le besoin, en comprenant rapidement quel acteur est intéressé !

Qui fait la tâche ? Utilisez des avatars !

Comment savoir qui fait quelle tâche ?

Simple : utilisez des avatars ! Des images, cartoons, etc...

Ceci est plus efficace que simplement mettre ses initiales ou son nom... La connection entre l'image et la tâche devient instantanée avec un avatar.



Autre possibilité : utiliser des étiquettes de couleur. (Trello)

Mettre une date butoire

N'en abusez pas !

Description

Dovilė Sinkevičiūtė Monday, January 09, 2012 11:12 AM

Segoe UI 12 [List Icons] [Text Icons]

Create several drafts, keep brand colors

Assignment

Jacob Robinson

Total time 4h 23m
[Pause Icon] [Stop Icon]

[Color Swatches]

Additional fields

Additional fields can be added through integration configuration. See custom integration help for additional info

Tasks

Count: 3
Completed 2/3

ChangeLog

Count: 17

Date: 11/13/2012 9:38:07 AM User: Dovilė Sinkevičiūtė
Action: Item moved from column 'Development' to 'Test'

Attachments

Count: 1

eylean_scr...

Comments

Alert

Deadline: November 23

Complexity left:

-

+

Estimate left:

h

-

+

Categories

Tâches problématiques

En tant que...
Je veux...
Afin de...

La machine est
encore cassée !
XXX

Parfois appelées “tâches puantes” : ce sont les tâches à problèmes, qui ne peuvent pas être résolues rapidement.

- Elles doivent être bien visibles (ex : mini post-it rouge),
- Elles doivent exprimer le problème factuellement,
- Une barre de progression doit être mise. Ici, une croix X est égal à un jour. La tâche est donc bloquante depuis 3 jours.


En 2013, une équipe américaine est allée encore plus loin : elle utilisait une peau de banane pour indiquer les tâches bloquantes ! Après quelques jours, l'équipe savait qu'elle devait régler absolument le problème...



Types de tâches



**User Story
classique**



**Tâche de
maintenance
ou technique**



**Bug
Défauts**

Il en existe d'autres !
(Ex : tâches
organisationnelles, tâches
action rétrospective...)

Aller plus loin dans l'expression des tâches

- ▶ Exemples :
 - ▶ Mettre des smileys,
 - ▶ Marquer les dates de début et de fin,
 - ▶ Marquer les estimations des tâches,
 - ▶ Mettre les IDs des outils numériques utilisées (Jira),
 - ▶ Créer des checklists...

Il existe beaucoup de façon de créer des stories !

3.

limiter le travail en cours (WIP)

Le WIP (Work in Process)

Le travail en cours signifie :

Tout le travail qui existe actuellement.

Soit : les tâches sur lesquelles vous travaillez, les tâches en train d'être analysées, les tâches non-commencées mais existantes...

Tout ce qui n'est pas fini pour délivrer de la valeur à l'utilisateur final !

Le WIP (Work in Process)

Non pas limiter le travail tout court,
Mais **limiter le travail en cours** !

Est-ce que cela sert **vraiment** à
accélérer notre flux de travail ?
Voyons cela grâce à **la loi de Little** !



From the Book: Stop Starting, Start Finishing, by Arne Rook

Little's Law

Cycle time = Work in Process / Throughput

Cycle time : *le temps mis par une tâche pour faire tout le process de travail*

Work in Process : *le nombre de tâches sur lesquelles vous travaillez en même temps*

Throughput : *le temps moyen mis pour compléter chaque tâche*

Little's law en pratique

Exemples :

1 mois = 12 tâches en même temps / (12 tâches par mois).

0,5 mois = 6 tâches en même temps / (12 tâches par mois).

2 mois = 24 tâches en même temps / (12 tâches par mois).

Pourquoi diminuer le nombre de tâches effectuées en même temps ?

Afin d'avoir un feedback plus rapide des tâches réalisées, et d'apprendre plus vite comment améliorer son process !

WIP dans le développement informatique

- ▶ Des spécifications pas encore implémentées
- ▶ Du code pas encore intégré ("ça marche sur mon PC, pourtant !")
- ▶ Du code pas encore testé,
- ▶ Du code pas encore mis en production,
- ▶ etc...

Les effets de trop de WIP

- ▶ Du multitâche par **changement de contexte**, soit un manque de concentration sur les tâches,
- ▶ Un retour d'expérience (*feedback*) rallongé,
- ▶ Plus de difficulté à s'adapter rapidement au changement, le risque technique est donc plus grand,
- ▶ Plus de difficulté à s'organiser en interne sur qui fait quoi,
- ▶ Une perte en qualité dûe au manque d'adaptation,
- ▶ Une perte de motivation ("Je m'occuperai de tes tâches dans 3 semaines, car j'ai déjà 15 tâches à faire d'abord").

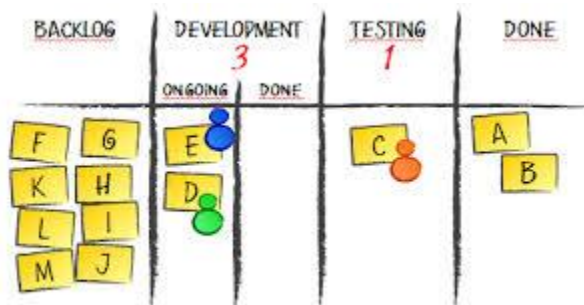


Limiter le travail en cours

Le secret du WIP : “Le but n’est pas de trouver les limites du travail en cours, mais de s’améliorer dans la poursuite d’un meilleur flux !”

Quelques règles :

- ▶ Une limite plus petite est meilleur qu’une limite trop grande
- ▶ Des personnes inactives ou un travail inactif
- ▶ Ne pas mettre de limite est une mauvaise solution !



Limiter le travail en cours

Quelques principes pour décider des limites :

- ▶ Le vrai but de la limite est de “commencer à finir” plutôt que de commencer de nouvelles tâches,
- ▶ Une limite de 1 est une très mauvaise idée : en cas de perturbations dans le flux, comme des blocages, des collaborateurs malades, tout le flux s'en trouve arrêté !
- ▶ Essayer toujours d'avoir une limite relativement petite, en évitant qu'une personne ait plus de 2 tâches à faire en même temps.

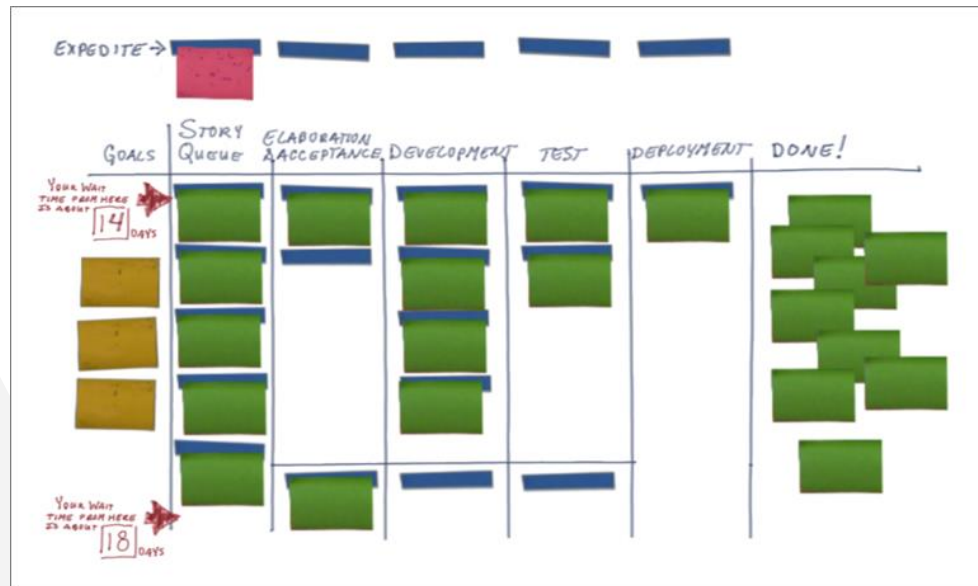
Limiter le travail en cours

- ▶ Limiter par équipe (exemple : 10 tâches en même temps pour une équipe de 5 personnes)
- ▶ Limiter par colonne (exemple : 3 tâches pour la colonne “Développement”, 2 tâches pour la colonne “Test”)
- ▶ Limiter par personne (exemple : 2 tâches par personne)
- ▶ Limiter par l'estimation des tâches (Story points par exemple)

Comment choisir son nombre ? En essayant, en testant, et en vérifiant ce qui fonctionne et ce qui ne fonctionne pas !

Faisons un Kanban !

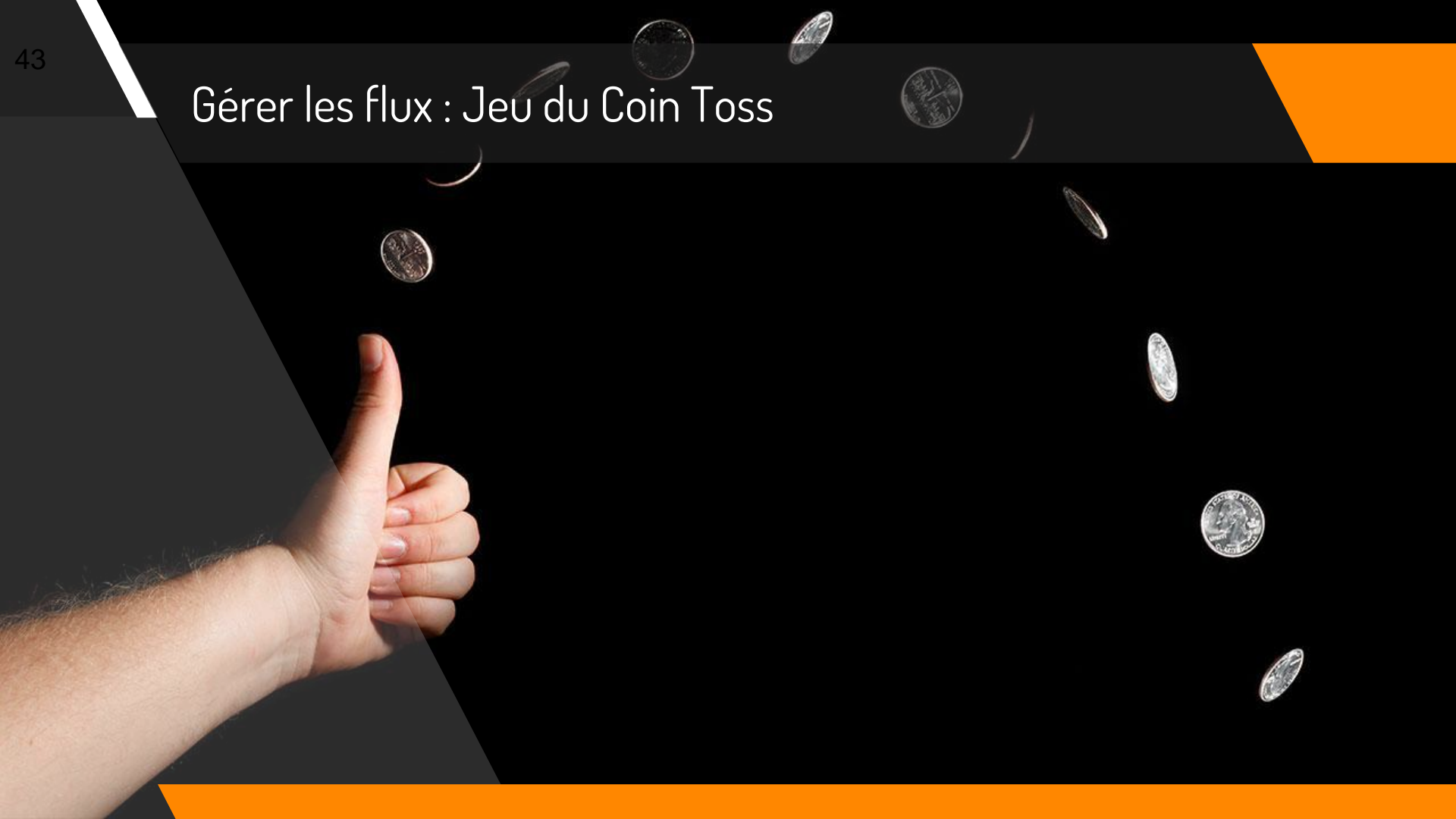
Créons ensemble un premier kanban !



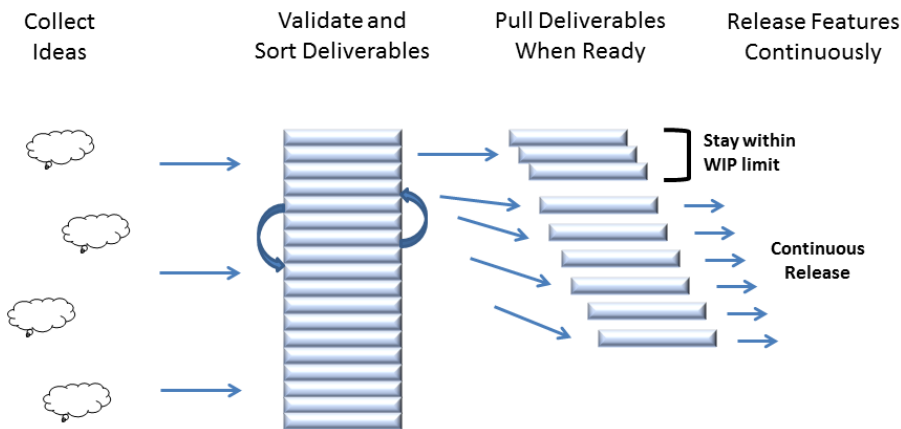
4.

Gérer les flux

Gérer les flux : Jeu du Coin Toss



Kanban / Continuous



Flux continu : un état idéal de la valeur, sans interruption ou temps d'attente. C'est une quête sans fin, pour améliorer son flux encore et encore...

Éliminer le gaspillage

Mary et Tom Poppendieck ont identifié 7 gaspillages dans l'IT :

- ▶ Le travail partiellement terminé,
- ▶ Des features bonus (CHAOS Report 2000 : 45% des features sont inutilisés par les utilisateurs finaux)
- ▶ Réapprendre un travail déjà effectué,
- ▶ Passer son travail à un collègue, ce qui entraîne une perte de données par la communication,
- ▶ De longs délais,
- ▶ Le multitâche,
- ▶ Les défauts techniques.

Les défauts sont ce qui empêchent votre flux d'avancer

🔧 **Cherchez en priorité à améliorer votre R.O.T.I ! (Return on time Invested)**

Gérer le flux

- ▶ Limiter le travail en cours
- ▶ Réduire les temps d'attente, par exemple en vérifiant que le travail est toujours prêt ou en réduisant au maximum la taille des tâches
- ▶ Réduire les points bloquants rapidement
- ▶ Eviter de retravailler ses produits avec une qualité du premier coup, par une "définition de fini"
- ▶ Créer des équipes pluri-disciplinaires

**TOUT LE
CODE EST
ÉCRIT** + **TOUS LES
TESTS
PASSENT** + **LE CLIENT (P.O.)
A ACCEPTÉ LA
FONCTIONNALITÉ**
= DONE. FINI. DONE.

Le Point Quotidien

Pour aider à améliorer le flux, le point quotidien permet d'appréhender la situation du moment et les problèmes.

- ▶ Court (15 minutes max)
- ▶ Timeboxé (commence et fini à l'heure)
- ▶ Rester concentré
- ▶ Régulier



Popularisé par Scrum et XP : le Daily Scrum.

Le Point Quotidien

Le Point Quotidien Kanban classique :

- ▶ Qu'ai-je fait hier ?
- ▶ Que vais-je faire aujourd'hui ?
- ▶ Quels sont les blocages qui m'empêchent d'avancer ?

Intérêt : permettre à l'équipe de se synchroniser sur le travail à faire.

Important : être debout !



Le Point Quotidien

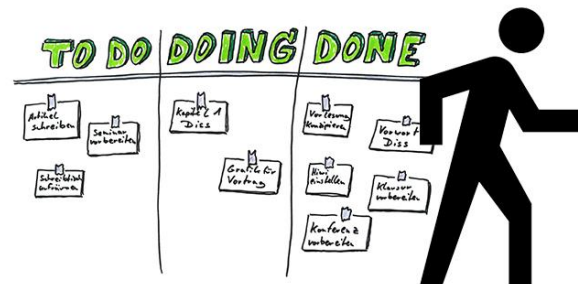
Walk the board

Enumérer le travail effectué de droite à gauche, depuis la colonne "Done".

- ▶ Que devons-nous faire pour faire en sorte de l'état "Done" ?
- ▶ Qui va s'en occuper ?

Focus sur les blocages

- ▶ Parler uniquement des blocages : efficace en cas de gros problèmes récurrents...



Réunions Kaizen “spontanées”

- ▶ Effectuées après le point quotidien.
- ▶ Spontanées.

改善

Kai = Change Zen = Good

Comment améliorer notre process ?

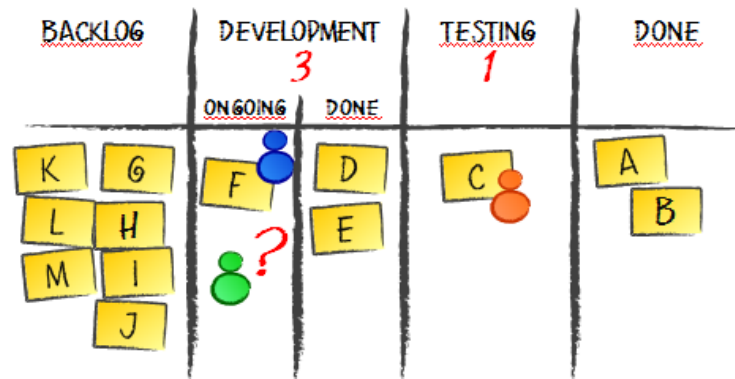
“Trouvons une solution technique pour ce blocage !”

“Peut-on se voir après le point quotidien pour discuter de ta tâche ? J’ai une bonne idée !”

A encourager !

Gérer les bouchons

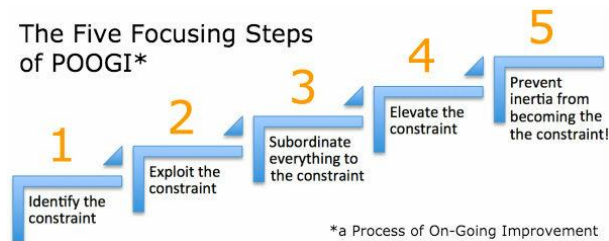
- ▶ Grâce au WIP, vous découvrirez vos propres bouchons.
- ▶ Sur l'image de droite, vous découvrez un bouchon. Que faire ?



- ▶ Solution court terme ? Solution long terme ?

Gérer les bouchons : la théorie des contraintes

1. Identifier la contrainte (une machine où la production passe forcément par elle)
2. Exploiter la contrainte (vérifier que la machine fonctionne tout le temps)
3. Subordonner le travail en support (ne pas envoyer de mauvaises pièces vers la machine, ce qui augmenterait son temps de prod)
4. Élever la contrainte (Acheter une autre machine)
5. Répétez les étapes pour que l'inertie ne devienne pas la contrainte !

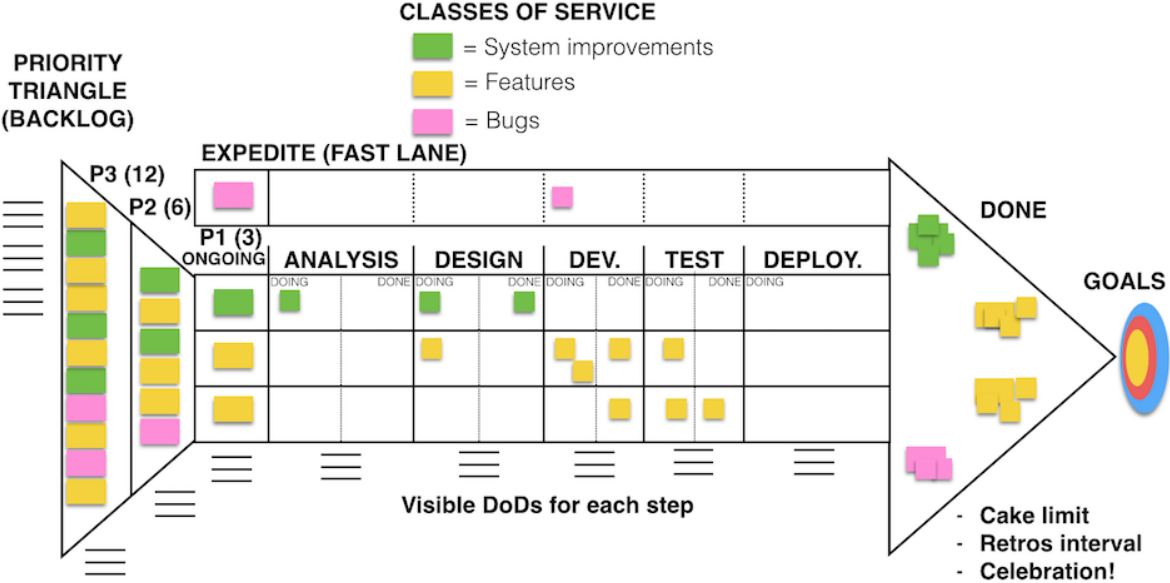


5.

Classes de service,
Planifier et estimer

Exemple : la ligne urgente

Ligne urgente



Exemple : la ligne urgente

- ▶ Tâches exceptionnelles pour votre flux normal
- ▶ Priorité plus importante
- ▶ Une ligne de flux séparée des autres
- ▶ Ne pas la compter comme WIP classique




Classes de service

Comment gérer les tâches dont la flux de production doit différer ?

- ▶ Permet l'auto-organisation
- ▶ Explications sur comment fonctionne chaque Classe de service
- ▶ Visualisation ? (couleur de post-it ?)
- ▶ Impact sur le WIP ?
- ▶ Priorisation ?
- ▶ Différent flux de travail ?

Classes of Services



Class	Characteristics	Contribution to flow
Expedite class	Highest priority. Don't interrupt the work on this task.	Max. 1 task in the process.
Fixed delivery date	Task has to be completed at a defined date.	> 20% in a release
Bug	Is a faulty implementation, has to be corrected as soon as possible.	(has no specific release assignment)
Standard class	Normal task. Will be treated by the FIFO principle. Has no timeline.	> 50% in a release
Intangible class or chore	Assisting task and has no prioritization and release assignment.	> 30% in a release

Classes de service

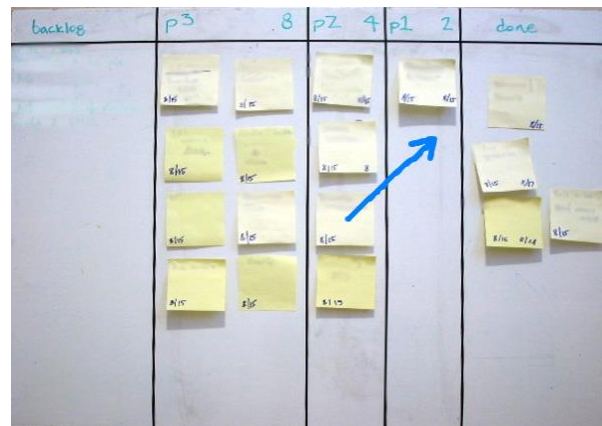
Exemples classiques :

- ▶ Tâches avec deadline
- ▶ Features ou User Stories
- ▶ Tâches intangibles (ex : dette technique)
- ▶ Tâches “défectueuses” (ex : bug)

Prioritisation par la planification

Planification pour priorisation :

- ▶ Juste-à-temps : planifier quand on en a besoin.
- ▶ Point de commande : lorsqu'un nombre de tâches devient inférieur à un certain point (exemple : 3 tâches et moins dans le backlog), on lance une planification.
- ▶ Filtre de priorité : diviser son backlog en colonnes de priorités.



6.

Amélioration continue

Cycle PDSA

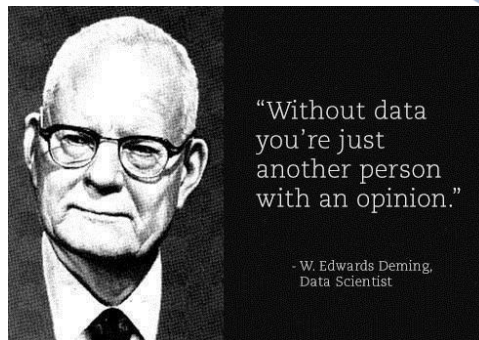
L'Agile est basé sur les travaux de Deming.

Plan : Quel est notre angle d'attaque ?

Do : Comment s'organise-t-on quotidiennement ?

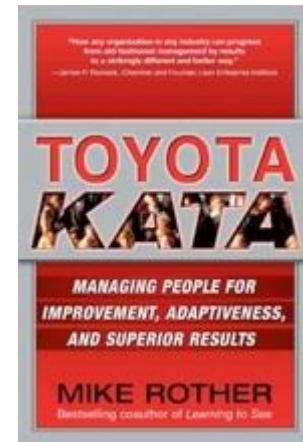
Study : Que faut-il étudier comme mesures ?

Act : Comment peut-on s'améliorer ?



L'amélioration continue : le véritable intérêt de kanban

Toyota Kata de Mike Rother :
Plus un process vieillit avec le temps sans être amélioré, plus il perd en valeur.

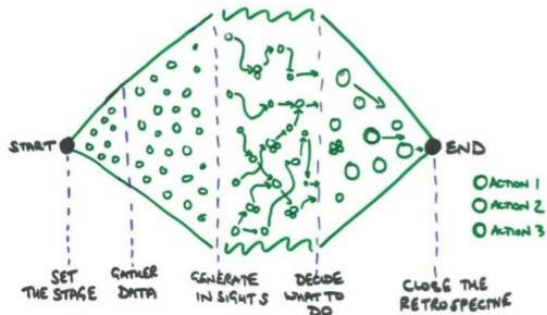


Votre process doit donc continuellement s'adapter et être amélioré pour ne pas devenir sans valeur !

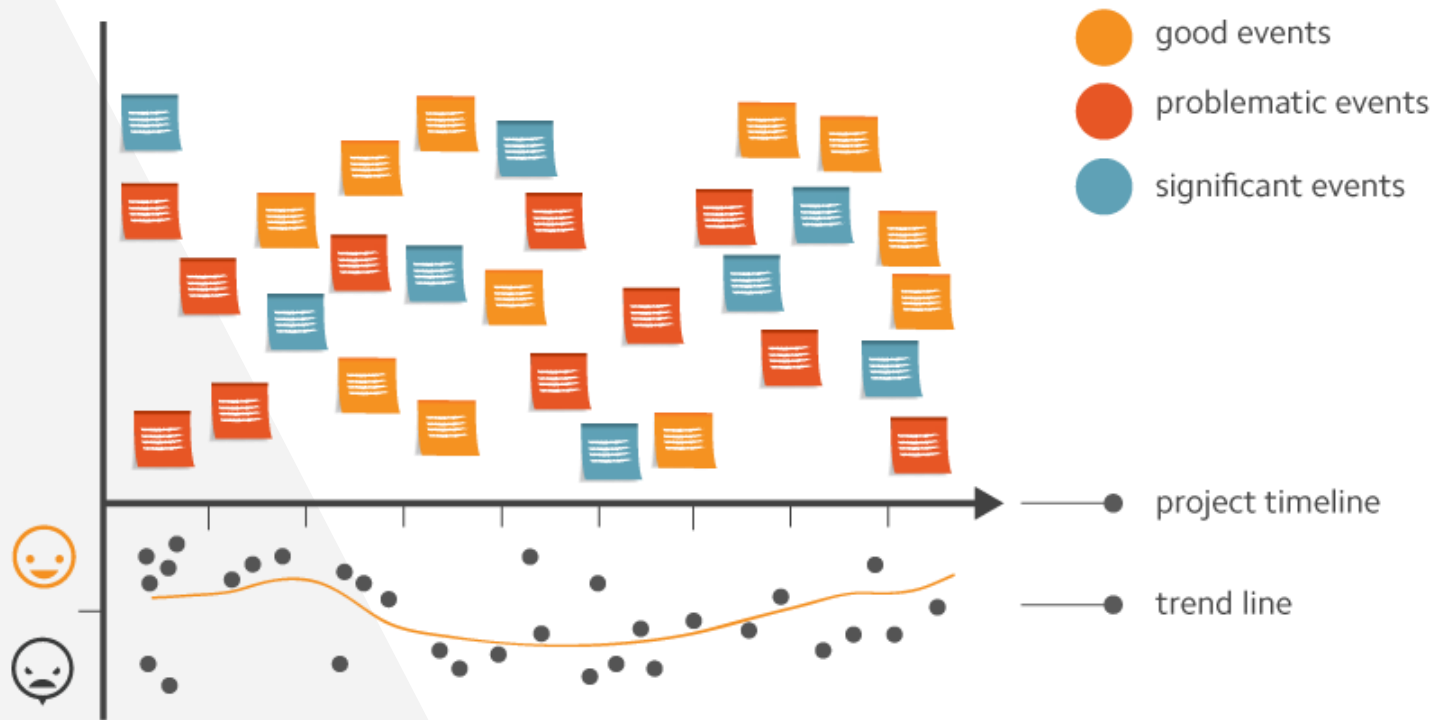
La rétrospective

Le but de la rétrospective, un rituel effectué à des intervals réguliers (1 fois toutes les semaines ou 4 semaines max), est de trouver des actions d'améliorations (1 ou 2) avant la prochaine rétrospective.

Structuring a retrospective



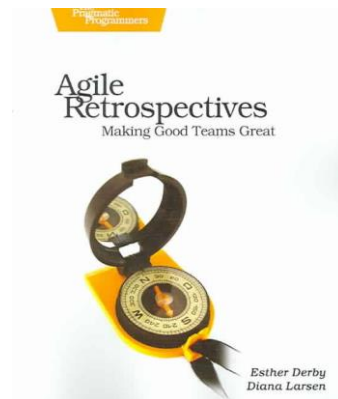
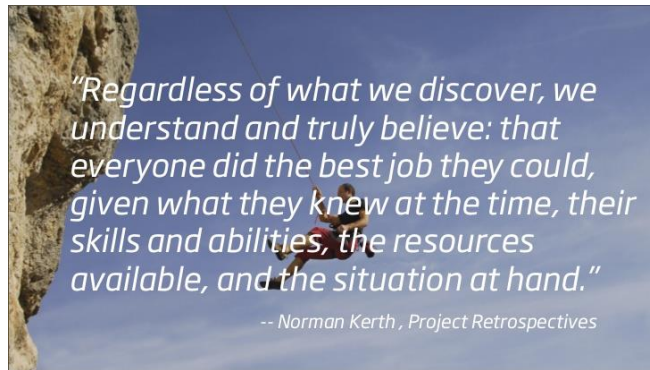
Rétrospective classique : la timeline



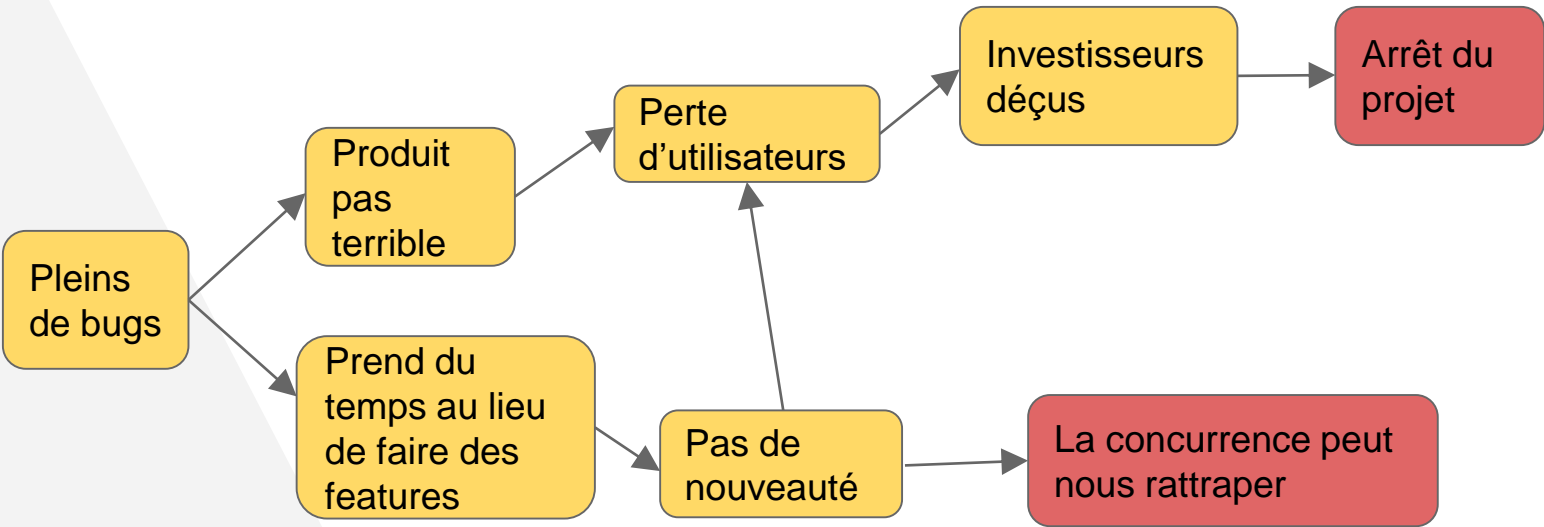
La Rétrospective

1. Mise en place
2. Rassembler les données
3. Générer des avis
4. Décider quoi faire
5. Clôturer la rétrospective

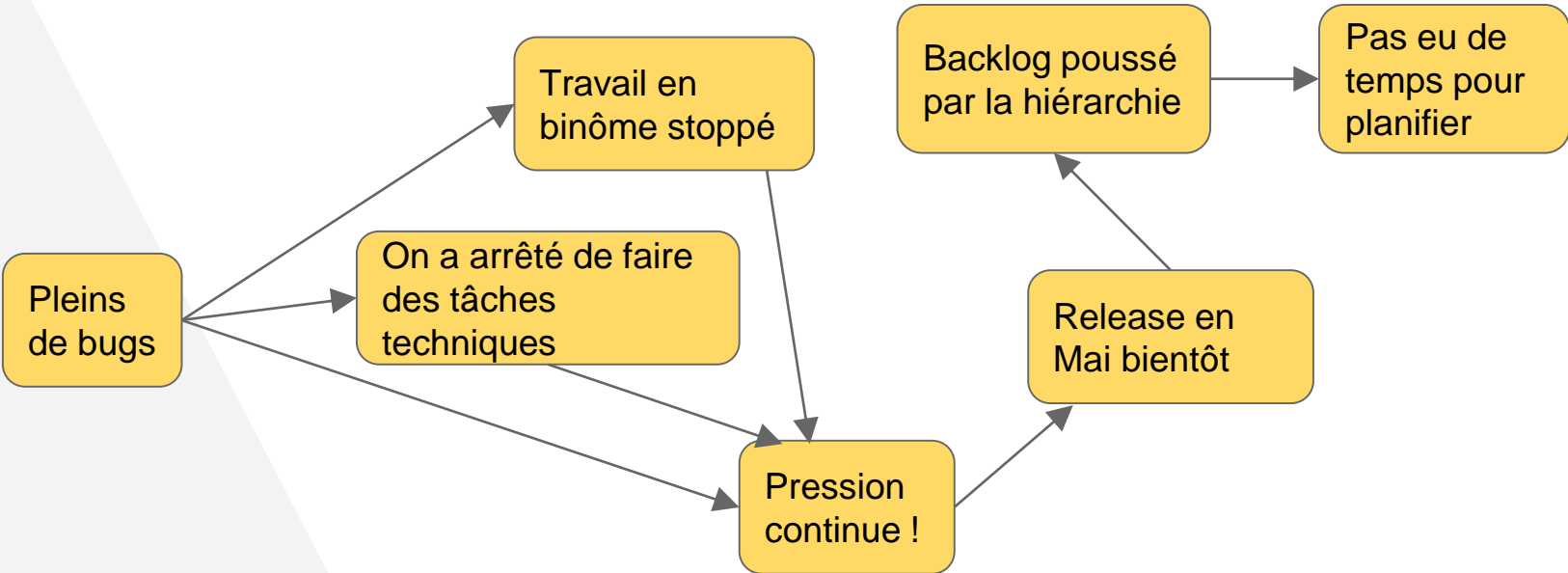
La bible : Agile Retrospective
Par Esther Derby et Diana Larsen.



Analyse des causes profondes (root case analysis)



Analyse des causes profondes (root case analysis)



Coaching Katas : approche scientifique

Les 5 questions :

1. Quelle est la condition cible ?
2. Quelle est la condition actuelle ? (puis vérification PDSA)
3. Quels obstacles vous empêchent d'atteindre la condition cible ?
4. Quelle est votre prochaine étape ? Qu'attendez-vous comme résultats?
5. Quand pourrons-nous observer ce que nous avons appris de cette expérience ?

Amélioration via questions prédéfinies

COACHING KATA

The Five Questions

- 1) What is the **Target Condition**?
- 2) What is the **Actual Condition** now?
-----*(Turn Card Over)*----->
- 3) What **Obstacles** do you think are preventing you from reaching the target condition?
Which ***one*** are you addressing now?
- 4) What is your **Next Step**?
(Next experiment) What do you expect?
- 5) How quickly can we go and see what we **Have Learned** from taking that step?

*You'll often work on the same obstacle with several experiments

Reflect on the Last Step Taken

Because you don't actually know what the result of a step will be!

- 1) What did you plan as your **Last Step**?
- 2) What did you **Expect**?
- 3) What **Actually Happened**?
- 4) What did you **Learn**?

----->
Return to question 3

Sans mesure, pas d'amélioration !

- ▶ Que doit-on mesurer ? **L'avancement opérationnel ! (7eme principe de l'agile manifesto)**

Une mesure est une visualisation de la santé de votre process.

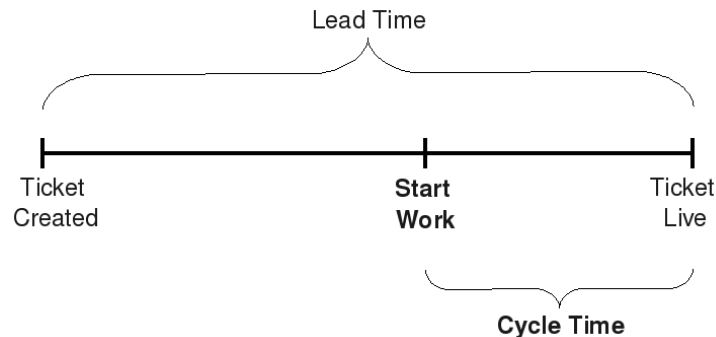
STREETLIGHT EFFECT - AGILE



BY AXOSOFT.COM

Types de mesures

- ▶ Cycle time : temps mis pour compléter une partie du process.
- ▶ Lead time : temps mis pour compléter tout le process.
- ▶ Throughput : le nombre de stories faites par semaine (ou mois, ou autres mesures de temps)
- ▶ Nombre de tâches bloquantes sur le tableau
- ▶ Performance par rapport aux deadlines
- ▶ Demande de valeurs VS demande d'échec : vérifier le nombre d'échecs, soit de refonte d'une feature, par rapport aux features demandées initialement.



Visualiser les mesures : Statistical Process Control (SPC)

Exemple :

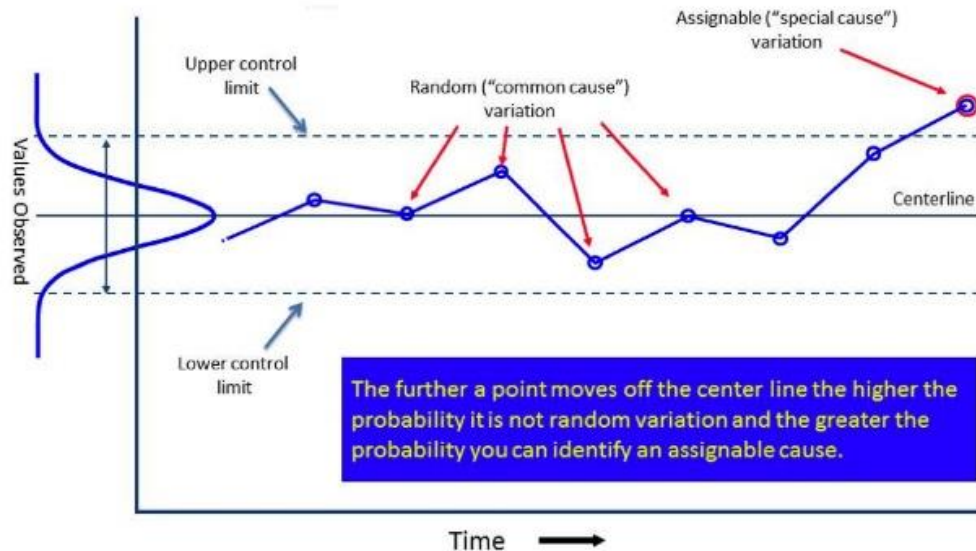
En ordonnée, le "lead time" en moyenne par semaine.

En abscisse, les semaines.

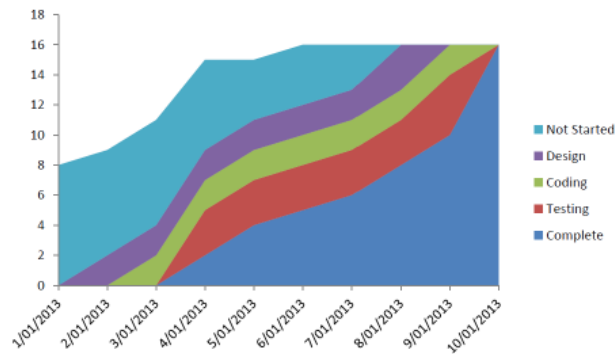
Intérêt :

Trouver les variations spéciales, et comment on en est arrivé là.

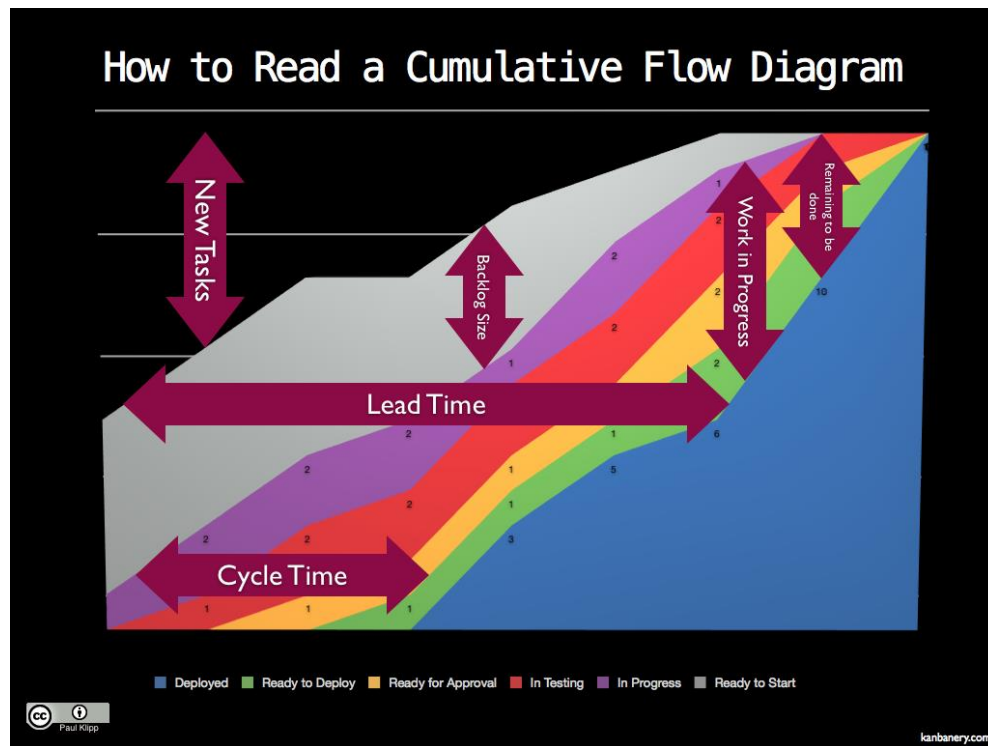
Statistical Process Control Chart



Visualiser les mesures : Cumulative Flow Diagram (CFD)



Complexe à lire, mais très puissant !
L'un des outils de mesure les plus utilisés en kanban.



7.

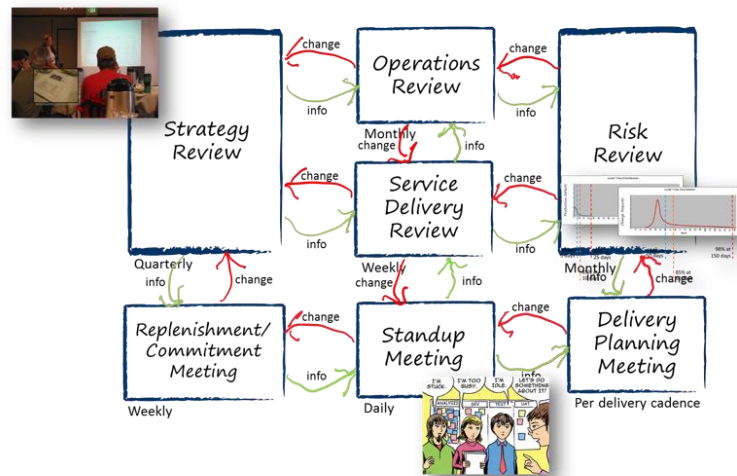
Cadences et Scrumban

Flux et Cadences

Kanban fonctionne de base en “flux continu”, des tâches arrivant au fur et à mesure.

Il existe une autre façon de faire du kanban : par “cadences” ou itérations.

Ce qui est très proche du cadre méthodologique “Scrum” !



Timebox

Une timebox est **un temps maximal** alloué à un évènement.

Cela permet de :

- ▶ Se concentrer sur un objectif.
- ▶ Manager le risque (deadlines itératives).
- ▶ Aide à l'auto-organisation.
- ▶ Le temps est fixé, l'objectif peut donc changer.



Rituels d'équipes

Popularisés par Scrum et XP (eXtreme Programming), les rituels classiques sont :

- ▶ Planification : quel est le besoin actuel du client, et quel est notre plan d'action pour y répondre ?
- ▶ Point quotidien : quel est notre plan d'action au jour le jour ?
- ▶ Revue : quel est l'avis de l'utilisateur ? A-t-on bien identifié le besoin ?
- ▶ Rétrospective : qu'est ce qui a bien marché ? Qu'est-ce qu'on doit changer ? Comment peut-on s'améliorer soi-même et notre démarche actuelle ?



Scrumban

Scrumban : mixe de Kanban et Scrum.

Modifications par rapport à Kanban :

- ▶ Rôles désignés : Scrum Master (coach), Product Owner (représentant du client), Equipe de développement (équipe technique)
- ▶ Point quotidien (Daily Scrum) obligatoire, les autres rituels sont optionnels (Planification, Revue, Rétrospective) mais recommandés.

Cependant, il n'existe pas de guide clair sur Scrumban...

SCRUM

VS

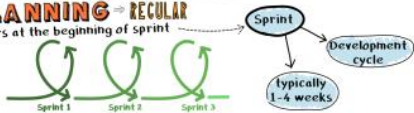
KANBAN

VS

SCRUMBAN

PLANNING → REGULAR

occurs at the beginning of sprint



ESTIMATIONS of TIME

BEFORE start of sprint



items should be small to finish within sprint



CHANGES TO WORK SCOPE

should wait for next sprint



ROLES



MEETINGS

SPRINT PLANNING
1-4 hour collaborative session
SPRINT

DAILY SCRUM
10-15 min everyday
everybody talks about achievements/issues

SPRINT REVIEW
0.5-2 hours review the results

RETROSPECTIVE
0.5-2 hours what went well and what did not

OWNERSHIP

Product Owner

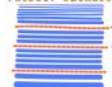
WHEN TO USE

- small items - small value
- adding increments possible
- requirements in a good shape
- roadmap is clear
- more cross-dependent teams



BOARDS / ARTIFACTS

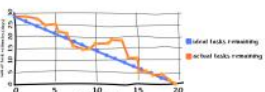
PRODUCT BACKLOG



SPRINT BACKLOG



BURNDOWN CHART



PLANNING

NOT PRECISE planning routine

PLAN WHEN they FINISH items

demand planning



CONTINUOUS FLOW



ESTIMATIONS of TIME

optional when items are completed



LIMIT

how many items can be in working columns at the same time

teams simply PULL

next item from backlog and implement it

CHANGES TO WORK SCOPE

added AS NEEDED



ROLES AS NEEDED



MEETINGS

NONE REQUIRED

OWNERSHIP

DEPENDS on defined roles and necessities



WHEN TO USE

Changes are too fast

support/maintenance work (operational level)



BOARDS / ARTIFACTS

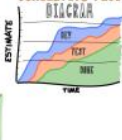
KANBAN BOARD



LEAD and CYCLE TIME DIAGRAM



CUMULATIVE FLOW DIAGRAM



PLANNING

plan when items are completed



ESTIMATIONS of TIME

optional when items are completed



LIMIT

work in progress

teams simply PULL

next item from backlog and implement it

CHANGES TO WORK SCOPE

added AS NEEDED



ROLES



MEETINGS

DAILY SCRUM STANDUP



other scrum related ceremonies IF NEEDED
typically ALL

OWNERSHIP

DEPENDS on defined roles and necessities

typically Product Owner

WHEN TO USE

Evolving requirements

no clear roadmap

OR

too fast changes

Need to include support/ maintenance (event driven) work in the process



BOARDS / ARTIFACTS

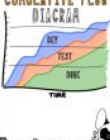
PROCESS BOARD



LEAD and CYCLE TIME DIAGRAM



CUMULATIVE FLOW DIAGRAM



7.

Pizza time !

Atelier final : Kanban pizza game !

