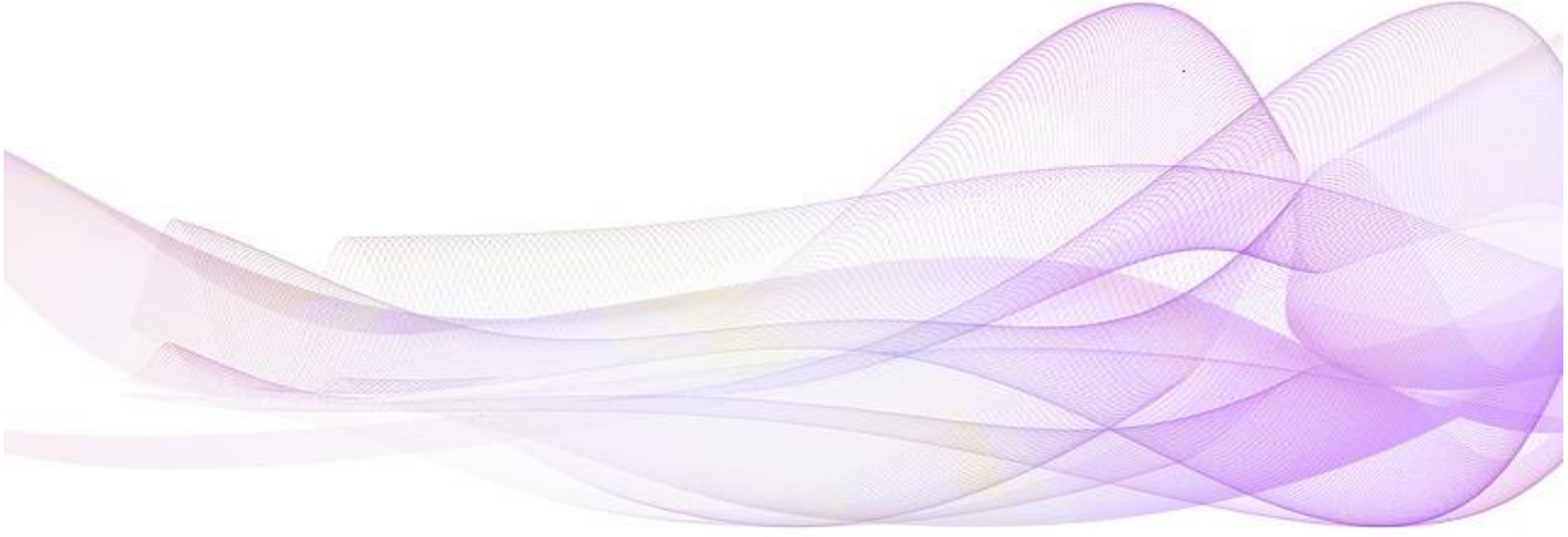
AFFINAGE OU REFINEMENT (GROOMING)

Comment écrire des User Stories efficaces



AGENDA

- ✓ **Objectifs**

- ✓ **Les 3C**

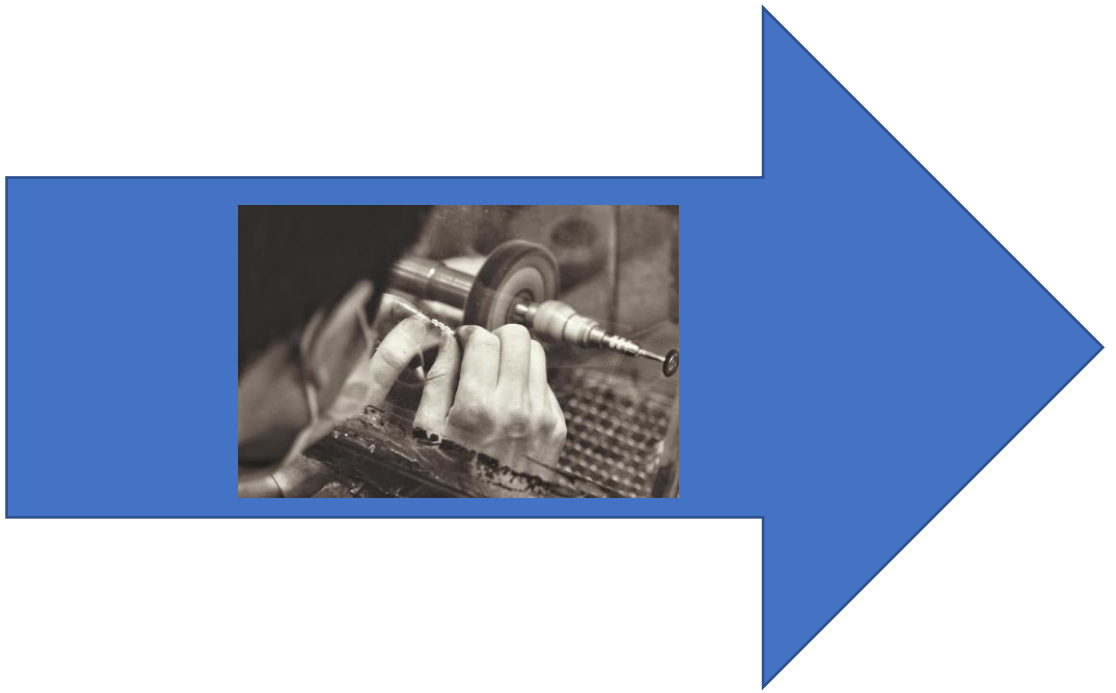
- ✓ **Kiss**

- ✓ **Structure**

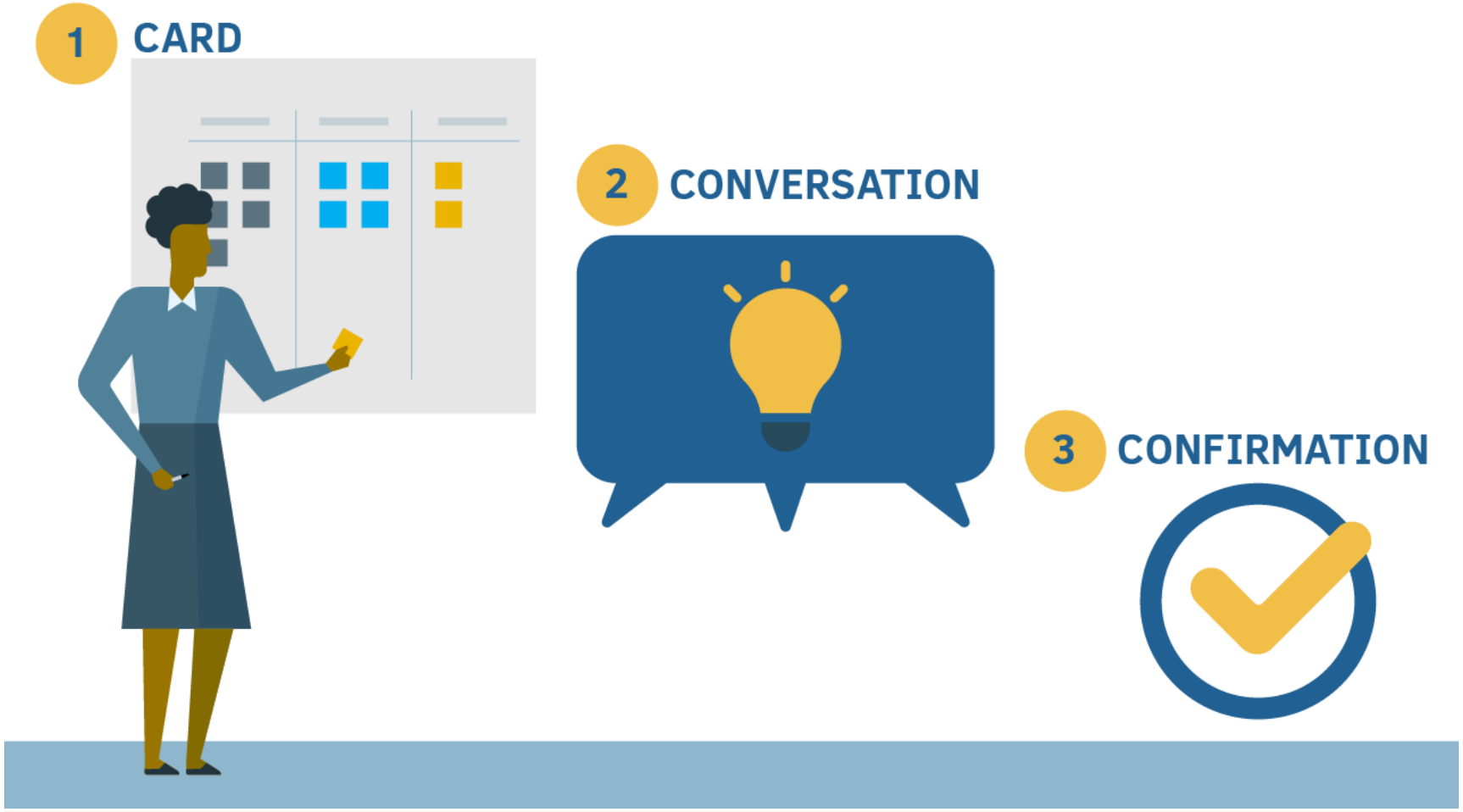
- ✓ **Recommandations**

- ✓ **Invest**

L’objectif de l’affinage



Les 3 C



https://resources.scrumalliance.org/Article/anatomy-user-story

KISS

Keep It Simple & Stupid

Commençons par formuler la carte ainsi :
"En tant que < rôle >,
je souhaite < action >,
afin de < valeur ou justification >"

Structure au choix

Car ce qui est important, c’est :

- Qui est l'utilisateur ? (Partie prenante, persona, rôle)
- Qu'est-ce qu'ils veulent faire? (Action)
- Pourquoi veulent-ils le faire ? (Valeur ou justification)

Puis, lors de la conversation, ajoutons à la simple story de base des réflexions supplémentaires aux notes sur l'élément ; joignons des documents et des dessins au fur et à mesure que nous en apprenons davantage

Attention aux adverbes ambigüs : exemples : rapidement, facilement, régulièrement…
Préférons la précision : exemples : en 3s, d’une seule main, au moins tous les 2 jours…

Il s'agit du problème, pas de la solution

- Que veut on résoudre comme problème, pour qui et pour quoi?
  - Exemple : **En tant qu**'étudiant, **je souhaite** rester en contact avec les autres étudiants de ma classe **afin de** collaborer, en sécurité, à la réalisation des TD de groupe

- Et non décrire ce qu'il faut faire et comment il faut le faire
  - Exemple : «YAQA » installer « TIMS;-) » sur nos ordinateurs…

Inclure les attentes importantes pour l'utilisateur

- En précisant les besoins non fonctionnels* :
  - **Contraintes** pesant sur le système : prix maximum de la solution à développer, ressources humaines et matérielles imposées, …
  - **Conformité** du système à un environnement : normes réglementaires, documentaires, conformité aux licences acquises, …
  - **Maintenabilité** du système : traçage des erreurs, possibilité des mises à jour, extensibilité / modifiabilité du système initial, supportabilité en fonction de l'implantation géographique du futur système, testabilité…
  - **Performance** du système: charge utilisateurs / transactions
  - **Portabilité** du système : compatibilité avec diverses plateformes, facilité de remplacement d'autres systèmes en place, facilité d'installation et de désinstallation de l'application …
  - **Fiabilité** du système : capacité à gérer les erreurs du système, densité des défauts de qualité, capacité à être remis en état rapidement, capacité à résister aux attaques…
  - **Sécurité** du système : traçage des mises à jour des données dans le système, gestion de la confidentialité, gestion de l'intégrité des données, protection des données personnelles …
  - **Utilisation** du système : facilité d'utilisation en limitant le nombre de clic à maximum 3 clics pour finaliser la transaction, rendre l'application attractive à une certaine audience (facteurs émotionnels), certification du système à une technologie particulière, capacité à respecter les exigences d'un pays, réutilisabilité de certains composants…

*Les exigences « non fonctionnelles » sont celles qui sont importantes voire critiques aux yeux des utilisateurs et qui assurent le bon fonctionnement du système. (https://bestofbusinessanalyst.fr/comment-decrire-les-exigences-non-fonctionnelles/)

Recommandations

- La **conversation** doit conduire à un contenu de card « péniblement » clair, précis et sans ambiguïté (il ne doit plus rester de question en suspend)
- Le produit de la **conversation** doit permettre à l'équipe de réalisation d'estimer la réalisation
- Il ne faut pas hésiter à « **découper** » une story trop grosse ou trop complexe en d'autres plus « gérables »
- La **confirmation** doit décrire les éléments permettant de connaitre les **critères d'acceptation*** de la story (l'ensemble des propositions qui seront vraies une fois la réalisation mise à disposition de l'utilisateur).
  - Exemples :
    - 100000 utilisateurs simultanés
    - temps de réponse inférieur à 2 secondes par utilisateur
    - L'écran est conforme aux normes graphiques de l'entreprise
    - L'ensemble des informations du client sont accessibles en appuyant sur une touche

*Attention de ne pas confondre avec la « définition of done » qui est valable pour toutes les stories

INVEST

- **Indépendant**

Les histoires indépendantes peuvent être librement réorganisées dans le backlog du produit. Parfois, vous ne pouvez pas vous débarrasser d'une dépendance de commande, mais cela devrait être une exception.

- **Négociable**

Une user story est le rappel d'avoir une conversation. Dans cette conversation, l'équipe négocie la solution concrète, la partie "Je veux". L'histoire peut être améliorée ou réécrite.

- **de Valeur**

Chaque histoire ajoute quelque chose d'utile pour l'utilisateur final / client - la partie "pour que". Cela conduit à des incréments verticaux : par exemple, une tranche de travail du frontal, des scripts et de la BD, au lieu d'une BD finie sans le frontal.

- **Estimable**

Vous avez besoin d'une estimation approximative de l'effort pour estimer le retour sur investissement et commander le backlog. Si vous ne pouvez pas estimer, vous devez a) diviser l'histoire en morceaux ou b) mieux comprendre la valeur qu'elle est censée ajouter ou c) explorer une technologie inconnue dans un pic de recherche limité dans le temps.

- **Petit (Small enough)**

Les petites histoires sont plus faciles à estimer et à tester et cachent moins de malentendus. "Petit" peut être 1 jour dans une boutique en ligne ou 3 semaines-personnes pour un produit médical. À tout le moins, l'équipe doit être capable de terminer une histoire ("done done") en 1 itération.

- **Testable**

Il doit être possible d'écrire un test (au moins en théorie) pour chaque histoire. Sinon, comment allez-vous confirmer que l'histoire est terminée ?
Parfois, des cas de test sont donnés comme critères d'acceptation.
Si vous ne pouvez pas penser à un test, l'histoire est probablement trop floue.

https://wall-skills.com/2015/good-user-stories-are-invest/