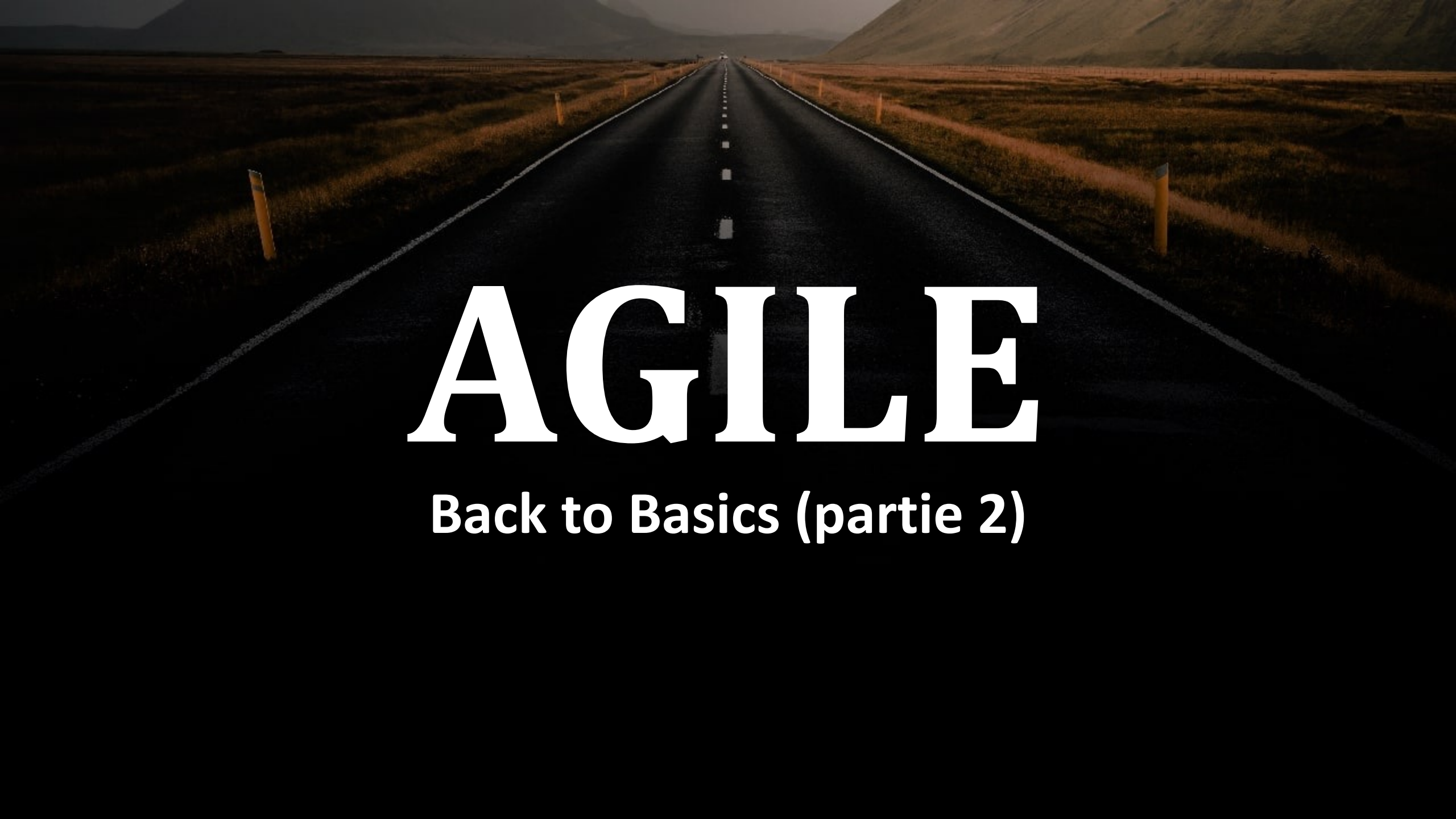


AGILE

Back to Basics (partie 2)

Rappel de la partie 1

- Waterfall et Spiral Model ont été mal interprété, les principaux échecs :
  - Planification en amont ne peut pas être juste
  - Le client ne sait pas lui-même son besoin -> découverte
    - Mais, impossible de s'adapter rapidement
  - On définit plusieurs fois l'objectif du logiciel (durant la phase Analyse, puis Dev puis Test)
- Des personnes remettent en question cette approche
  - (Triangle de fer, avec Agile seul le périmètre peut changer)
- *Nous nous sommes arrêté à la question Pourquoi Agile*

Slide suivante

Cours 2,3 : Framework XP et Scrum

- De nombreux Frameworks sont créés et ils précèdent Agile
  - 1995 : Scrum
  - 1996 : XP
  - Agile naît à partir des auteurs de ces mouvements en 2001

I
Agile le *pourquoi*

1	Orienté valeur
2	Ne plus espérer
3	Réduire l’incertitude
4	Promouvoir l’artisanat logiciel
5	Centré sur l’humain

#1 Orienté valeur

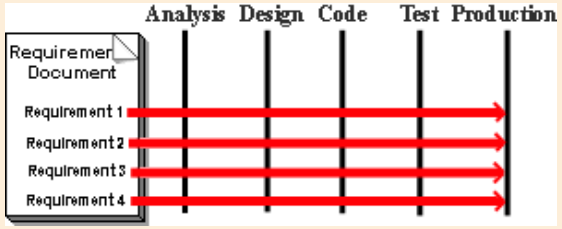
Affirmation
Notre objectif est de créer de la valeur pour le client

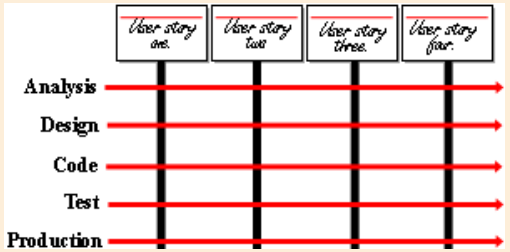
#1 Orienté valeur : comment ?

Question

Comment créer de la valeur ?

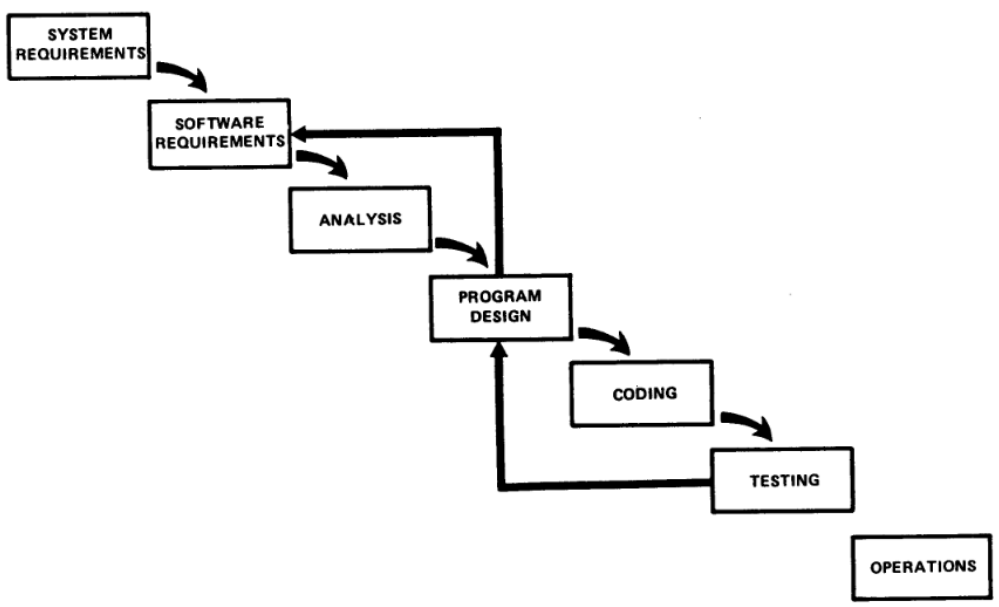
1. Feature > Activité
   - Car la valeur existe uniquement lorsque la feature est en production





2. Prioriser
   - Nous ne pourrons pas implémenter toutes les features, il faut donc les prioriser (cf Chapitre Estimation et Planification)
3. Implication du client

#2 Ne plus espérer



Affirmation

The loss of hope is a major goal of Agile.

Agile cherche à savoir le plus tôt possible à quel point on part dans la mauvaise direction, afin de garder le contrôle de la situation.

Question

Comment ?

Agile produit des données via les feedbacks clients

#3 Réduire l'incertitude

Affirmation

Le développement logiciel est très rarement prédictif

- <u>Pourquoi ?</u>
  - Le client ne connait pas ses besoins
  - Le développement logiciel est un process créatif se qui le rend compliqué à planifier et donc prédictible

Lecture « The new methodology » de Martin Fowler
Pourquoi il y a de l'incertitude
Comment pouvons-nous la réduire

#3 Réduire l'incertitude : *comment?*

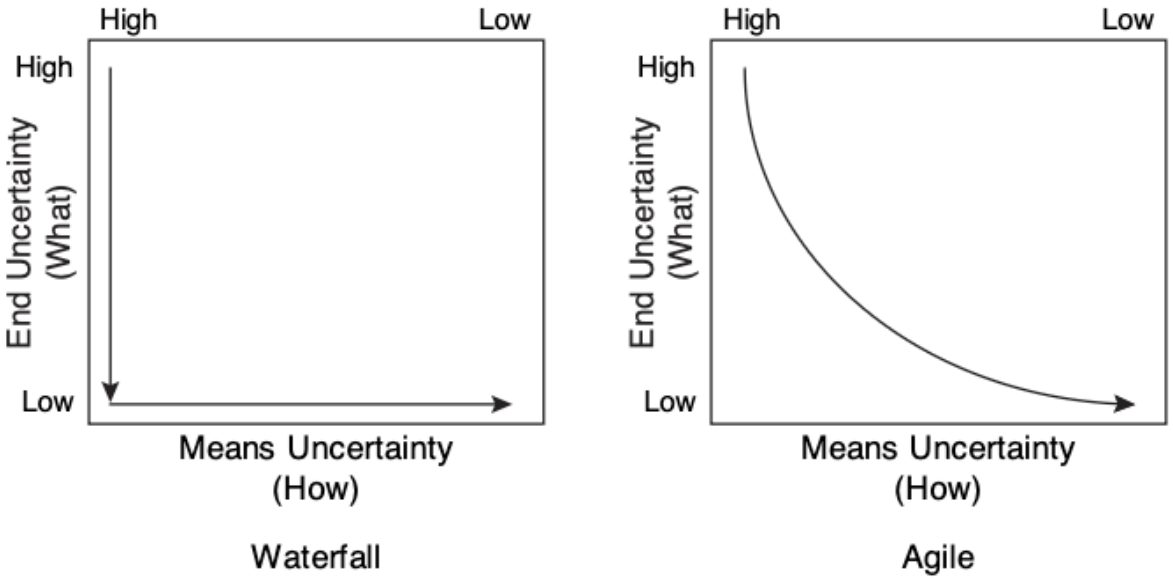


Figure 9.1 Traditional and agile views of reducing uncertainty. Adapted from Laufer (1996).

Question

Expliquez ces deux graphiques

Question

Avec Agile, quels moyens utiliser pour réduire l'incertitude

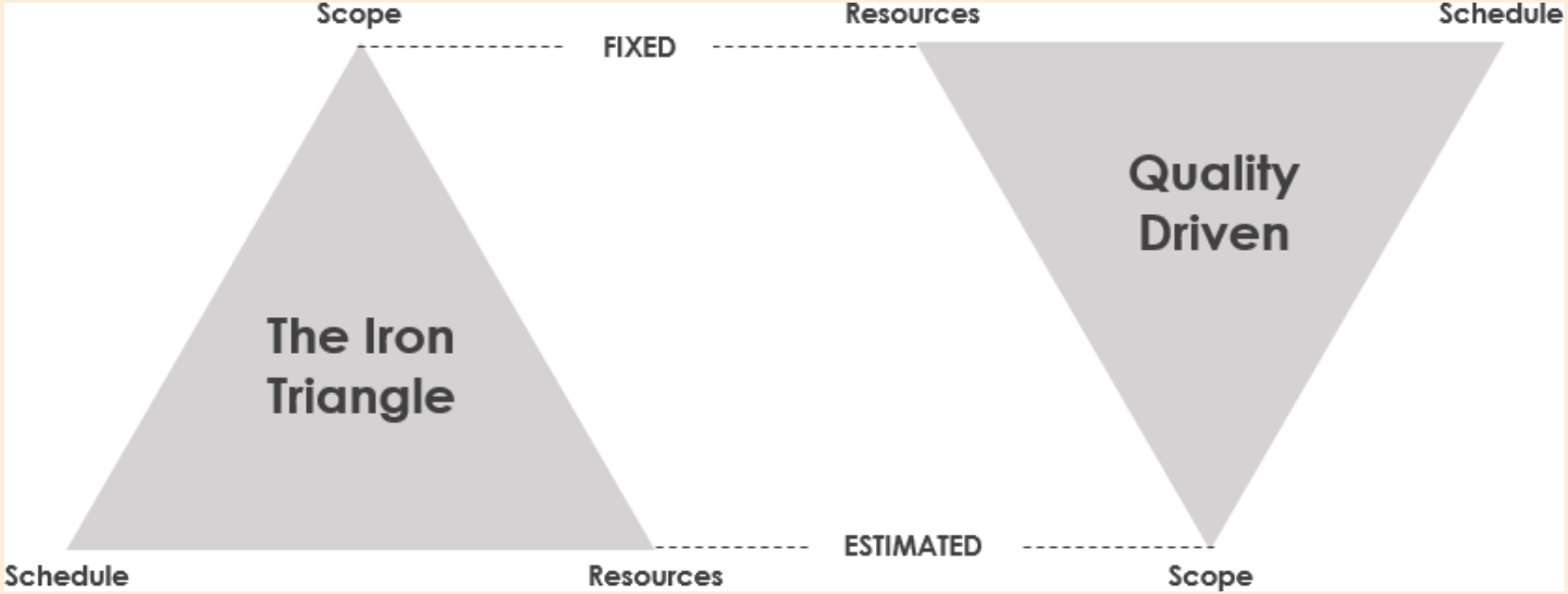
1. Itérations courtes
2. Boucles de feedback

#3 Réduire l'incertitude : *comment?*

- <u>Itérations courtes</u>
  - Permettent d'apporter du contrôle dans une situation peu prédictible
  - MAIS nécessite de changer la relation contractuel DEV/Client par une relation de **collaboration**
  - **=>** DONC à changer notre relation aux variables dynamiques et fixes

Slide suivante

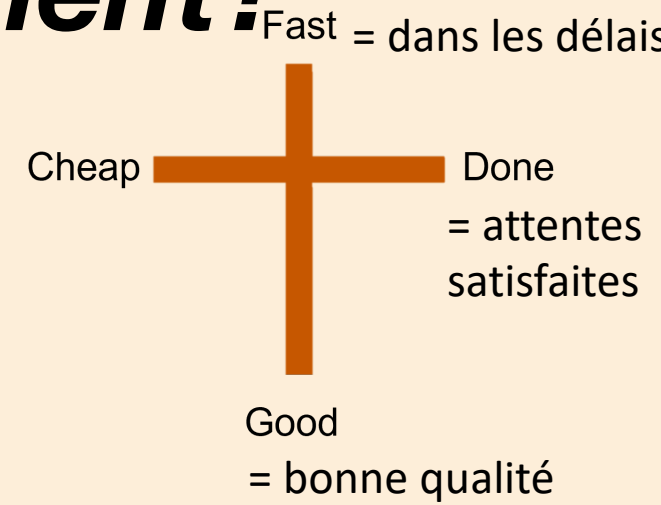
#3 Réduire l'incertitude : *comment?*



- Agile
  - La deadline + les ressources sont fixes
  - Le **périmètre est variable**

#3 Réduire l'incertitude : *comment?*

Vous ne pouvez satisfaire que 3 des 4 caractéristiques



- Changer calendrier : impossible
- Ajouter des ressources : Loi de Brooks
- Réduire la qualité : *Le seul moyen d'aller vite, c'est d'avancer proprement*

- Changer le périmètre : dans l'ordre de la valeur à produire -> implique prioriser

#3 Réduire l'incertitude : *comment?*

> Some folks think that Agile is about going fast. It's not. It's never been about going fast. **Agile is about knowing, as early as possible, just how screwed we are.**

- Feedbacks
  - Car couplés aux itérations ils permettent de **savoir au plus tôt**
  - et la connaissance permet de nous adapter (Inspect & Adapt)
    - Niveau équipe : qualité logicielle, ambiance, etc …
    - Niveau produit : **construit-on la bonne chose**

#3 Réduire l’incertitude : conclusion

- Via des itérations courtes et des boucles de feedback nous pouvons :
  - Accepter le changement (e.g. de besoins) à bras ouverts plutôt que de le craindre et le combattre
  - Avoir une amélioration continue

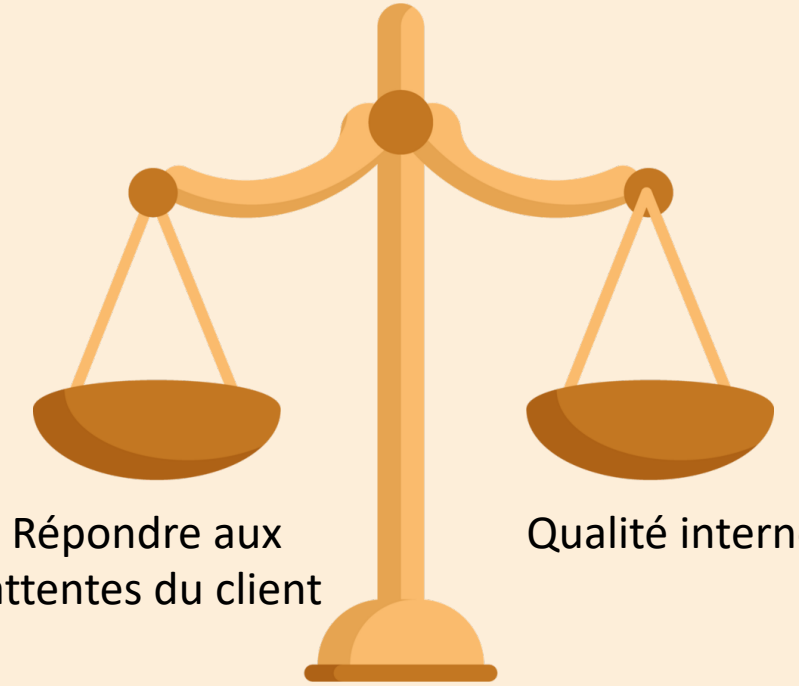
#4 Promouvoir l’artisanat logiciel

- Il y a la gestion de projet mais également l’artisanat logiciel, Agile chercher à lier fortement la gestion d’un projet logiciel avec les préoccupation des programmeurs

#4 Promouvoir l’artisanat logiciel

Notre objectif : Un produit de haute qualité

C’est quoi un produit de haut qualité ?



Répondre aux attentes du client

Qualité interne

Affirmation : Viser l’excellence

Agile est un engagement à tendre vers l’excellence, à être toujours plus professionnel et à favoriser les comportements professionnels dans l’industrie logiciel

#5 Human-centric

Question :

Centré sur l'humain, *qui?* se cache derrière le mot « humain » ?

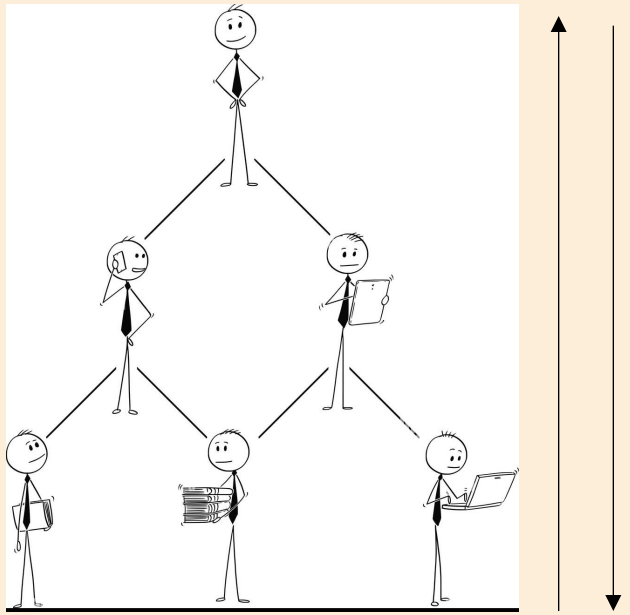
Les parties prenantes (stakeholders)

I would like the world to be safe for developers
- Kent Beck

#5 Human-centric : L'équipe propriétaire du processus

Question :

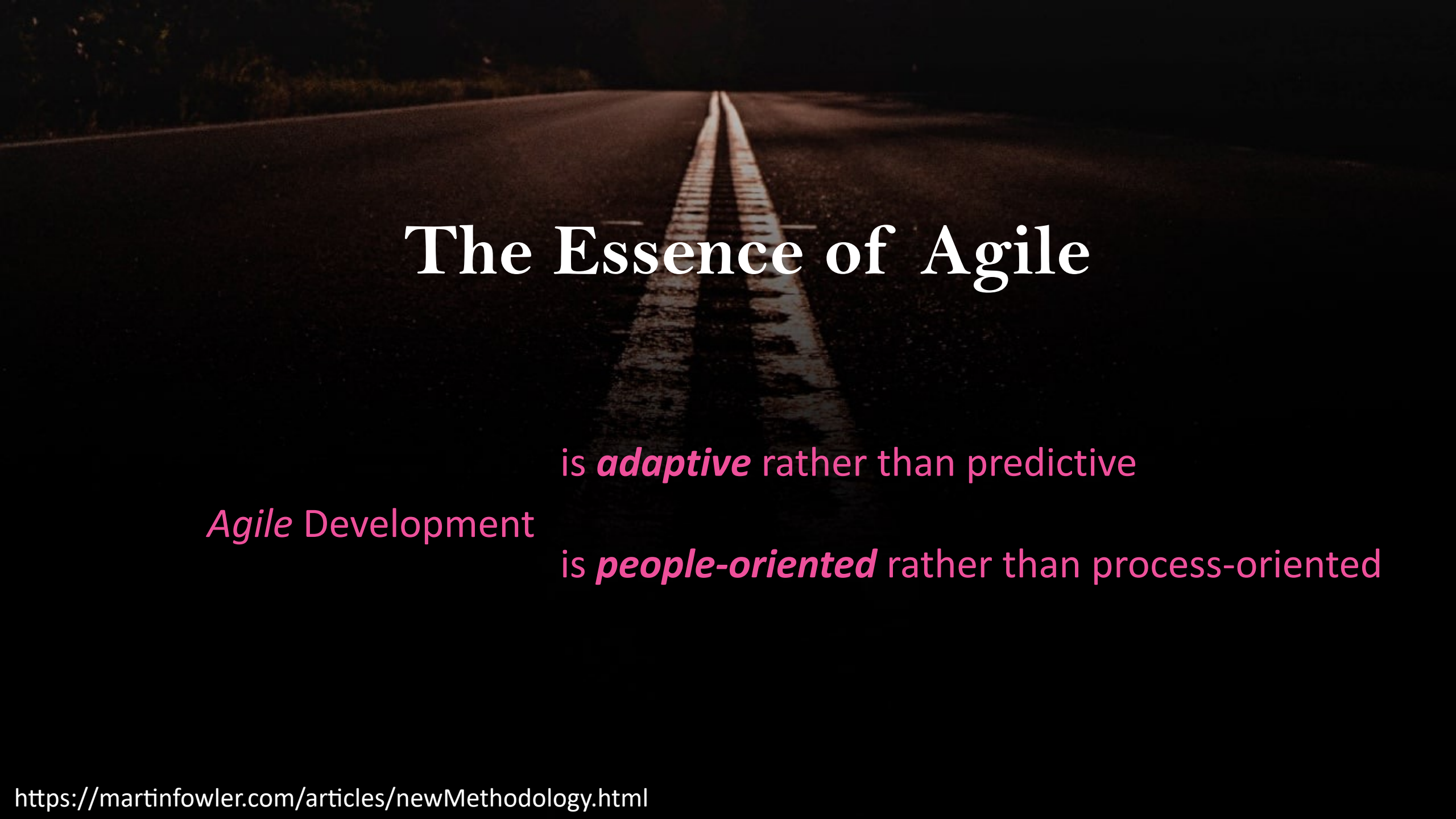
A votre avis, que cela peut-il signifier et impliquer



« Dans une approche d'amélioration continue, l'équipe est propriétaire de SON processus »

Equipe autonome et pluridisciplinaire

1	Orienté valeur
2	Ne plus espérer
3	Réduire l’incertitude
4	Promouvoir l’artisanat logiciel
5	Centré sur l’humain



The Essence of Agile

Agile Development

is ***adaptive*** rather than predictive

is ***people-oriented*** rather than process-oriented

https://martinfowler.com/articles/newMethodology.html

II
Définir Agile

La Manifeste Agile

History: The Agile Manifesto

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground–and of course, to eat. What emerged was the Agile 'Software Development' Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

Now, a bigger gathering of organizational anarchists would be hard to find, so what emerged from this meeting was symbolic–a *Manifesto for Agile Software Development*–signed by all participants. The only concern with the term *agile* came from Martin Fowler (a Brit for those who don't know him) who allowed that most Americans didn't know how to pronounce the word 'agile'.

Alistair Cockburn's initial concerns reflected the early thoughts of many participants. "I personally didn't expect that this particular group of agilites to ever agree on anything substantive." But his post-meeting feelings were also shared, "Speaking for myself, I am delighted by the final phrasing [of the Manifesto]. I was surprised that the others appeared equally delighted by the final phrasing. So we did agree on something substantive."

Naming ourselves "The Agile Alliance," this group of independent thinkers about software development, and sometimes competitors to each other, agreed on the *Manifesto for Agile Software Development* displayed on the title page of this web site.

But while the Manifesto provides some specific ideas, there is a deeper theme that drives many, but not all, to be sure, members of the alliance. At the close of the two-day meeting, Bob Martin joked that he was about to make a "mushy" statement. But while tinged with humor, few disagreed with Bob's sentiments–that we all felt privileged to work with a group of people who held a set of compatible values, a set of values based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work. At the core, I believe Agile Methodologists are really about "mushy" stuff–about delivering good products to customers by operating in an environment that does more than talk about "people as our most important asset" but actually "acts" as if people were the most important, and lose the word "asset". So in the final analysis, the meteoric rise of interest in–and sometimes tremendous criticism of–Agile Methodologies is about the mushy stuff of values and culture.

For example, I think that ultimately, Extreme Programming has mushroomed in use and interest, not because of pair-programming or refactoring, but because, taken as a whole, the practices define a developer community freed from the baggage of Dilbertesque corporations. Kent Beck tells the story of an early job in which he estimated a programming effort of six weeks for two people. After his manager reassigned the other programmer at the beginning of the project, he completed the project in twelve weeks–and felt terrible about himself! The boss–of course–harangued Kent about how slow he was throughout the second six weeks. Kent, somewhat despondent because he was such a "failure" as a programmer, finally realized that his original estimate of 6 weeks was extremely accurate–for 2 people–and that his "failure" was really the manager's failure , indeed, the failure of the standard "fixed" process mindset that so frequently plagues our industry.

This type of situation goes on every day–marketing, or management, or external customers, internal customers, and, yes, even developers–don't want to make hard trade-off decisions, so they impose irrational demands through the imposition of corporate power structures. This isn't merely a software development problem, it runs throughout Dilbertesque organizations.

In order to succeed in the new economy, to move aggressively into the era of e-business, e-commerce, and the web, companies have to rid themselves of their Dilbert manifestations of make-work and arcane policies. This freedom from the inanities of corporate life attracts proponents of Agile Methodologies, and scares the begeebers (you can't use the word 'shit' in a professional paper) out of traditionalists. Quite frankly, the Agile approaches scare corporate bureaucrats– at least those that are happy pushing process for process' sake versus trying to do the best for the "customer" and deliver something timely and tangible and "as promised"–because they run out of places to hide.

The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker.

The meeting at Snowbird was incubated at an earlier get together of Extreme Programming proponents, and a few "outsiders," organized by Kent Beck at the Rogue River Lodge in Oregon in the spring of 2000. At the Rogue River meeting attendees voiced support for a variety of "Light" methodologies, but nothing formal occurred. During 2000 a number of articles were written that referenced the category of "Light" or "Lightweight" processes. A number these articles referred to "Light methodologies, such as Extreme Programming, Adaptive Software Development, Crystal, and SCRUM". In conversations, no one really liked the moniker "Light", but it seemed to stick for the time being.

In September 2000, Bob Martin from Object Mentor in Chicago, started the next meeting ball rolling with an email; "I'd like to convene a small (two day) conference in the January to February 2001 timeframe here in Chicago. The purpose of this conference is to get all the lightweight method leaders in one room. All of you are invited; and I'd be interested to know who else I should approach." Bob set up a Wiki site and the discussions raged.

Early on, Alistair Cockburn weighed in with an epistle that identified the general disgruntlement with the word 'Light': "I don't mind the methodology being called light in weight, but I'm not sure I want to be referred to as a lightweight attending a lightweight methodologists meeting. It somehow sounds like a bunch of skinny, feebleminded lightweight people trying to remember what day it is."

The fiercest debate was over location! There was serious concern about Chicago in wintertime–cold and nothing fun to do; Snowbird, Utah–cold, but fun things to do, at least for those who ski on their heads like Martin Fowler tried on day one; and Anguilla in the Caribbean–warm and fun, but time consuming to get to. In the end, Snowbird and skiing won out; however, a few people–like Ron Jeffries–want a warmer place next time.

We hope that our work together as the Agile Alliance helps others in our profession to think about software development, methodologies, and organizations, in new- more agile - ways. If so, we've accomplished our goals.

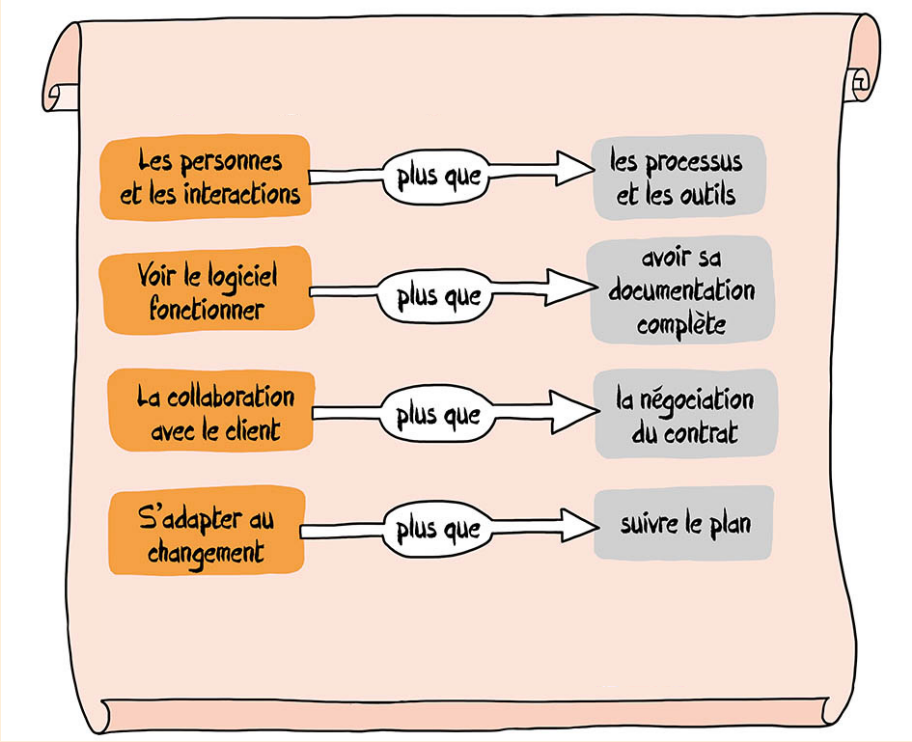
Jim Highsmith, for the Agile Alliance

http://agilemanifesto.org/history.html

Agile un ensemble de 4 valeurs (point)

Affirmation

Agile est une déclaration de quatres valeurs et de 12 principes.



https://manifesteagile.fr/

+ 12 principes

- Notre principale priorité est de satisfaire le client en livrant rapidement et régulièrement des solutions qui apportent de la valeur.
- Accueillez chaleureusement les changements de besoins, même tardifs dans le développement. Les processus agiles tirent parti du changement pour renforcer l'avantage concurrentiel du client.
- Livrez souvent des solutions opérationnelles, à une fréquence allant de quelques semaines à quelques mois, avec une préférence pour les échelles de temps les plus courtes.
- Les personnes en charge du métier ou des affaires et les personnes en charge de la réalisation doivent travailler ensemble chaque jour, tout au long du projet.
- Construisez les projets à partir de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour mener à bien le travail.
- La conversation en face à face est la méthode la plus efficace et la plus économique pour donner des informations à une équipe de réalisation, et pour échanger des informations à l'intérieur de l'équipe.
- La disponibilité de solutions opérationnelles est la principale mesure d'avancement.
- Les processus agiles encouragent à respecter un rythme soutenable lors de la réalisation. Les commanditaires, les réalisateurs et les utilisateurs devraient pouvoir maintenir indéfiniment un rythme constant.
- Porter continuellement attention à l'excellence technique et à la qualité de la conception renforce l'agilité.
- La simplicité – l'art de maximiser la quantité de travail qu'on ne fait pas – est essentielle.
- Les meilleures architectures, les meilleures spécifications de besoins, et les meilleures conceptions émergent d'équipes auto-organisées.
- À intervalles réguliers, l'équipe réfléchit aux façons de devenir plus efficace, puis modifie son comportement et l'ajuste en conséquence.

« Mes » définitions

Agile

- C’est une philosophie
- Qui défini un ensemble de valeurs et de principes
- Pour réaliser un produit de haute qualité
- Dans un monde à forte incertitude

Agile

- adopter une approche itérative et incrémentale
- menée dans un esprit collaboratif
- pour générer un produit de haute qualité
- en tenant compte de l’évolution client

- **flexibilité, répétition, ouverture aux changements**

« Mes » Définitions complètes

- C’est une philosophie qui définit un ensemble de valeurs et de principes
- Pour réaliser un produit de haute qualité
- En adoptant une approche itérative et incrémentale menée dans un esprit collaboratif
- Et en tenant compte de l’évolution client (monde à forte incertitude)

Déclaration des droits

Affirmation

Agile n’est pas simplement un ensemble de règles. Bien au contraire, Agile est un ensemble de droits, d’attentes et de disciplines formant la plateforme conceptuelle d’une profession éthique

Question :

Pourquoi la déclaration des droits ?

- Kent Beck avait déclaré que le but de Agile **était d’abord de réparer la fracture entre l’entreprise et les développeurs**, d’où l’apparition des droits suivants
  - droits pour le client
  - droits pour le développeurs

Les droits

Les clients/managers ont le droit

- d'avoir un plan d'ensemble : qu'est-ce qui peut être accompli, quand et à quel prix ?
- d'obtenir le plus de valeur après chaque itération
- de voir les progrès réalisés sur l'avancement du logiciel
- d'être informés des changements de calendrier et d'estimation.
- d'annuler à tout moment et de rester avec un système fonctionnel.

Les développeurs ont le droit …

- de savoir ce que l'on attend d'eux et quelles sont les priorités.
- de produire du code de haute qualité.
- de demander et recevoir de l'aide.
- de mettre à jour leurs estimations.
- d'accepter des responsabilités au lieu de se les voir infligés.

Droits client #1

Customers have the right to an overall plan and to know what can be accomplished when and at what cost.

Many people have claimed that up-front planning is not part of Agile development. The very first customer right belies that claim. Of course the business needs a plan. Of course that plan must include schedule and cost. And, of course that plan should be as accurate and precise as practical.

In short, we cannot agree to deliver fixed scopes on hard dates. Either the scopes or the dates must be soft.

Customers have the right to this kind of probability-based plan because they cannot manage their business without it.

Droits client #2

Customers have the right to get the most possible value out of every iteration.

Agile breaks up the development effort into fixed time boxes called *iterations*. The business has the right to expect that the developers will work on the most important things at any given time, and that each iteration will provide them the maximum possible *usable* business value. This priority of value is specified by the customer during the planning sessions at the beginning of each iteration. The customers choose the stories that give them the highest return on investment and that can fit within the developer's estimation for the iteration.

Droits client #5

Customers have the right to be informed of schedule and estimate changes in time to choose how to alter the scope to meet the required date.

Customers may cancel at any time and be left with a useful working system reflecting investment to date.

Note that customers do not have the right to demand conformance to the schedule. Their right is limited to managing the schedule by changing the scope. The most important thing this right confers is the right to *know* that the schedule is in jeopardy so that it can be managed in a timely fashion.

Droits développeur #2

Developers have the right to produce high-quality work at all times.

This may be the most profound of all these rights. Developers have the right to do good work. The business has no right to tell developers to cut corners or do low-quality work. Or, to say this differently, the business has no right to force developers to ruin their professional reputations or violate their professional ethics.

Droits développeur #4

Developers have the right to make and update their own estimates.

No one can estimate a task for you. And if you estimate a task, you can always change your estimate when new factors come to light. Estimates are guesses. They are intelligent guesses to be sure, but they're still guesses. They are guesses that get better with time. Estimates are never commitments.

Estimation != Engagement

Droits développeur #5

Developers have the right to accept their responsibilities instead of having them assigned.

Professionals *accept* work, they are not assigned work. A professional developer has every right to say "no" to a particular job or task. It may be that the developer does not feel confident in their ability to complete the task, or it may be that the developer believes the task better suited for someone else. Or, it may be that the developer rejects the tasks for personal or moral reasons.[6]

In any case, the right to accept comes with a cost. Acceptance implies responsibility. The accepting developer becomes responsible for the quality and execution of the task, for continually updating the estimate so that the schedule can be managed, for communicating status to the whole team, and for asking for help when help is needed.

« Conclusion »

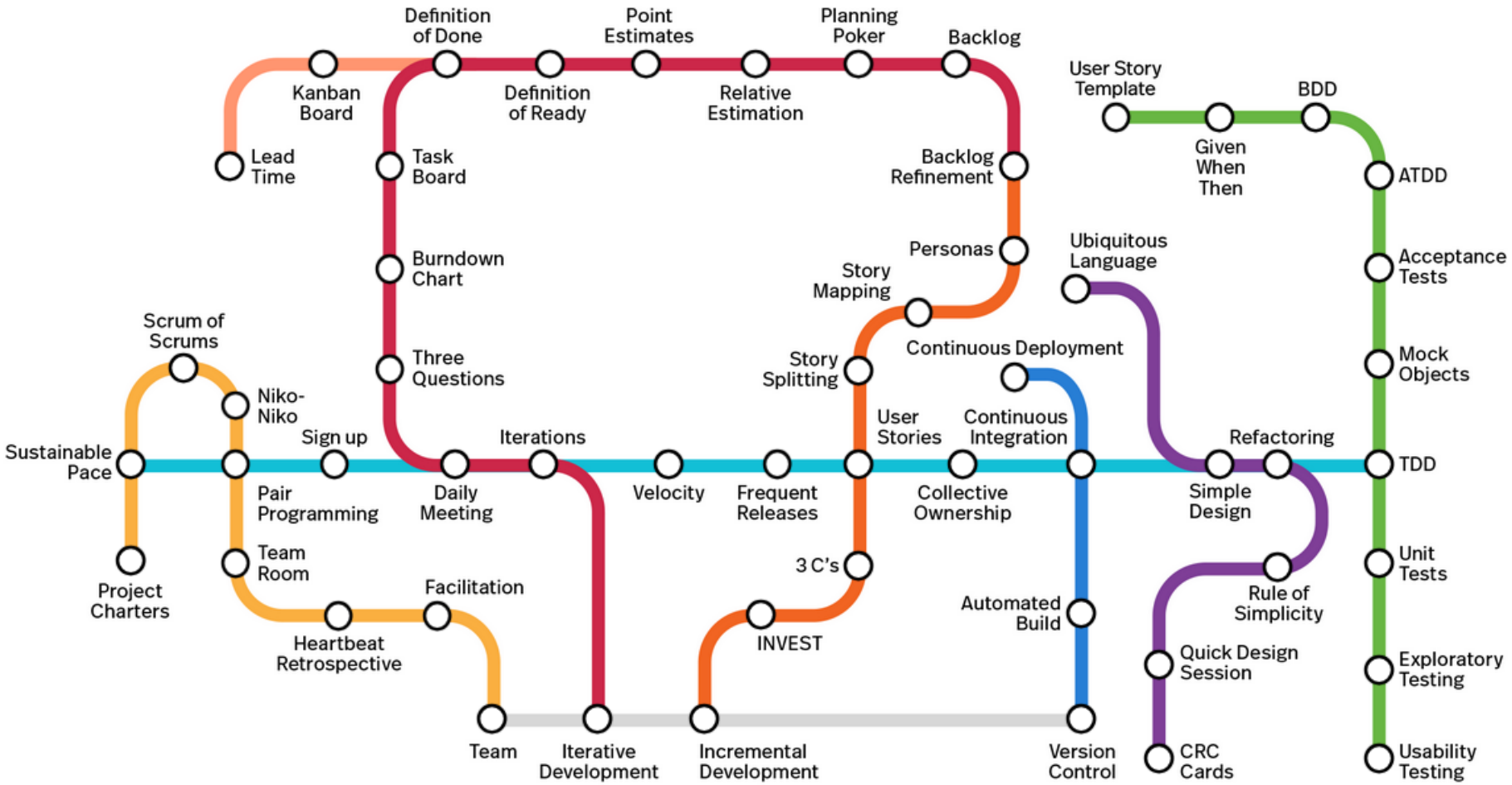
- Agile n’est ni un processus ni un engouement passager
- Agile n’est pas simplement un ensemble de règles
- Agile est un ensemble
  - De droits
  - D’attentes
  - Et de discipline

  - Formant la plateforme conceptuelle d’une profession éthique

> Agile is not a process. Agile is not a fad. Agile is not merely a set of rules. Rather, Agile is a set of rights, expectations, and disciplines of the kind that form the basis of an ethical profession.

IV
Pratiques Agile

The Agile Subway Map



The colored "subway" lines represent practices from the various Agile approaches or areas of concern.
Click on the "station" circles to read the full definition for each term.

Extreme Programming	Scrum	Design
Teams	Product Management	Testing
Lean	DevOps	Fundamentals