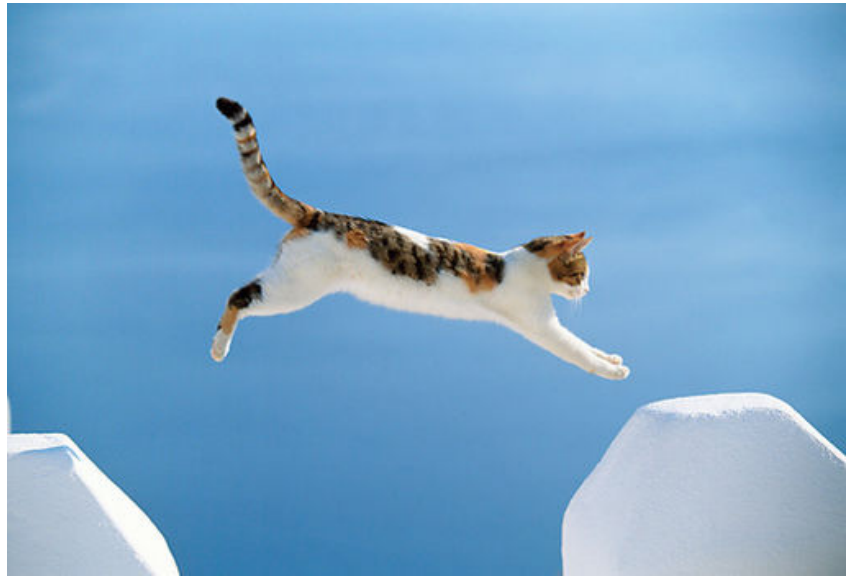
# Les Rôles Agiles

Christophe Grosjean

Wallix

4 octobre 2017



# Qu'est-ce que l'Agilité ?

- L'agilité est une approche de la conduite des projets informatique...
- ... et de la gestion des équipes des équipes de développement
- ... adaptée à la création de valeur client dans un environnement turbulent.
- Elle cherche à réduire les écarts entre le cycle d'évolution d'un produit/service et les processus d'une organisation.
- Elle est omniprésente (et plutôt clivante) dans le milieu du développement informatique.
- Certains essayent d'étendre son périmètre d'utilisation avec plus ou moins de succès.

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils
2. Des logiciels opérationnels plus qu'une documentation exhaustive

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils
2. Des logiciels opérationnels plus qu'une documentation exhaustive
3. La collaboration avec les clients plus que la négociation contractuelle

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils
2. Des logiciels opérationnels plus qu'une documentation exhaustive
3. La collaboration avec les clients plus que la négociation contractuelle
4. L'adaptation au changement plus que le suivi d'un plan

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique
et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils
2. Des logiciels opérationnels plus qu'une documentation exhaustive
3. La collaboration avec les clients plus que la négociation contractuelle
4. L'adaptation au changement plus que le suivi d'un plan

www.agilemanifesto.org

Février 2001

# Manifeste Agile

Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

1. Les individus et leurs interactions plus que les processus et les outils
2. Des logiciels opérationnels plus qu'une documentation exhaustive
3. La collaboration avec les clients plus que la négociation contractuelle
4. L'adaptation au changement plus que le suivi d'un plan

Nous reconnaîssons la valeur des seconds éléments, mais privilégions les premiers .

www.agilemanifesto.org

Février 2001

# Principes sous-jacents du Manifeste 1-4

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

# Principes sous-jacents du Manifeste 1-4

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins , même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

# Principes sous-jacents du Manifeste 1-4

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins , même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.

# Principes sous-jacents du Manifeste 1-4

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins , même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

# Principes sous-jacents du Manifeste 5-8

5. Réalisez les projets avec des personnes motivées.
   Fournissez-leur l'environnement et le soutien dont ils ont besoin
   et faites-leur confiance pour atteindre les objectifs fixés.

# Principes sous-jacents du Manifeste 5-8

5. Réalisez les projets avec des personnes motivées.
   Fournissez-leur l'environnement et le soutien dont ils ont besoin
   et faites-leur confiance pour atteindre les objectifs fixés.
6. 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 .

# Principes sous-jacents du Manifeste 5-8

5. Réalisez les projets avec des personnes motivées.
   Fournissez-leur l'environnement et le soutien dont ils ont besoin
   et faites-leur confiance pour atteindre les objectifs fixés.
6. 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 .
7. Un logiciel opérationnel est la principale mesure d'avancement.

# Principes sous-jacents du Manifeste 5-8

5. Réalisez les projets avec des personnes motivées.
   Fournissez-leur l'environnement et le soutien dont ils ont besoin
   et faites-leur confiance pour atteindre les objectifs fixés.
6. 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 .
7. Un logiciel opérationnel est la principale mesure d'avancement.
8. 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.

# Principes sous-jacents du Manifeste 9-12

9. Une attention continue à l' excellence technique et à une bonne conception renforce l'Agilité.

# Principes sous-jacents du Manifeste 9-12

9. Une attention continue à l' excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.

# Principes sous-jacents du Manifeste 9-12

9. Une attention continue à l' excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d' équipes auto-organisées .

# Principes sous-jacents du Manifeste 9-12

9. Une attention continue à l' excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d' équipes auto-organisées .
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

# Principes sous-jacents du Manifeste 9-12

9. Une attention continue à l' excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d' équipes auto-organisées .
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

# Pas vraiment nouveau

Lean (Toyota)

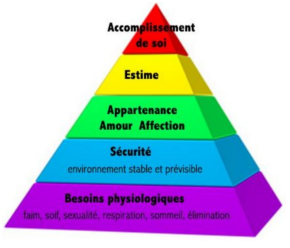
*Kiichiro Toyoda*

# Pas vraiment nouveau

Lean (Toyota)

*Kiichiro Toyoda*

Pyramide de Maslow



# Pas vraiment nouveau

Lean (Toyota)

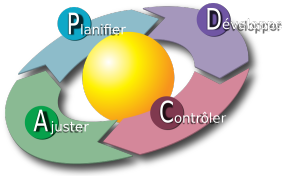
*Kiichiro Toyoda*

Pyramide de Maslow


Cercles de Qualité

# Pas vraiment nouveau

Roue de Deming




Lean (Toyota)

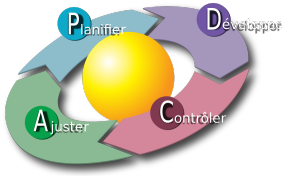
*Kiichiro Toyoda*

Pyramide de Maslow


Cercles de Qualité

# Pas vraiment nouveau

Roue de Deming




Lean (Toyota)

*Kiichiro Toyoda*

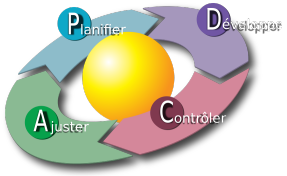
Pyramide de Maslow


Cercles de Qualité

Pédagogie Montessori

# Pas vraiment nouveau

Roue de Deming




Lean (Toyota)

*Kiichiro Toyoda*

Pyramide de Maslow

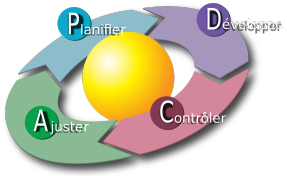

Cercles de Qualité

Émergence

Pédagogie Montessori

# Pas vraiment nouveau

Roue de Deming




Lean (Toyota)

*Kiichiro Toyoda*

Pyramide de Maslow


Cercles de Qualité

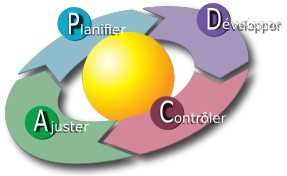
Émergence

Pédagogie Montessori

Complexité

# Pas vraiment nouveau

Roue de Deming




Anarchisme

Lean (Toyota)

*Kiichiro Toyoda*

Pyramide de Maslow


Cercles de Qualité

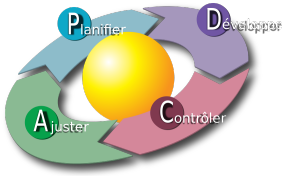
Émergence

Pédagogie Montessori

Complexité

# Pas vraiment nouveau

Roue de Deming




Cybernétique

Lean (Toyota)

*Kiichiro Toyoda*

Anarchisme

Pyramide de Maslow


Cercles de Qualité

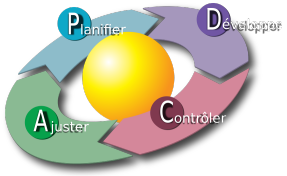
Émergence

Pédagogie Montessori

Complexité

# Pas vraiment nouveau

Roue de Deming




Cybernétique

Lean (Toyota)

*Kiichiro Toyoda*

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

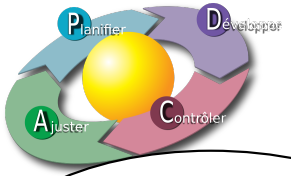
Émergence

Pédagogie Montessori

Complexité

# Pas vraiment nouveau

Roue de Deming




Cybernétique

Théorie Y (McGregor)

Lean (Toyota)

*Kiichiro Toyoda*

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

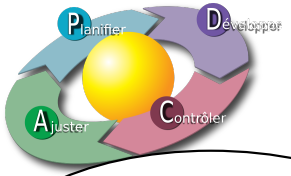
Émergence

Pédagogie Montessori

Complexité

# Pas vraiment nouveau

Roue de Deming




Cybernétique

Théorie Y (McGregor)

Lean (Toyota)

Kiichiro Toyoda

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

Émergence

Pédagogie Montessori

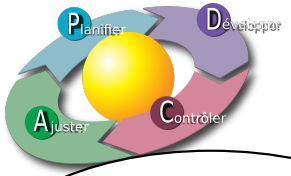
Complexité

Logiciel Libres

# Pas vraiment nouveau

Tests Unitaires, TDD

Roue de Deming




Cybernétique

Théorie Y (McGregor)

Lean (Toyota)

Kiichiro Toyoda

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

Émergence

Pédagogie Montessori

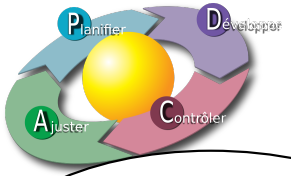
Complexité

Logiciel Libres

# Pas vraiment nouveau

Tests Unitaires, TDD

Roue de Deming




Cybernétique

Théorie Y (McGregor)

Lean (Toyota)

Kiichiro Toyoda

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

RAD

Émergence

Pédagogie Montessori

Complexité

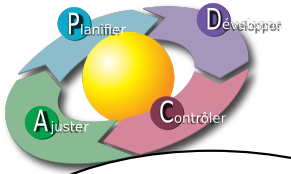
Logiciel Libres

# Pas vraiment nouveau

Tests Unitaires, TDD

Cybernétique

Roue de Deming




Théorie Y (McGregor)

Lean (Toyota)

Kiichiro Toyoda

Anarchisme

Pyramide de Maslow


Real Options

Cercles de Qualité

RAD

Émergence

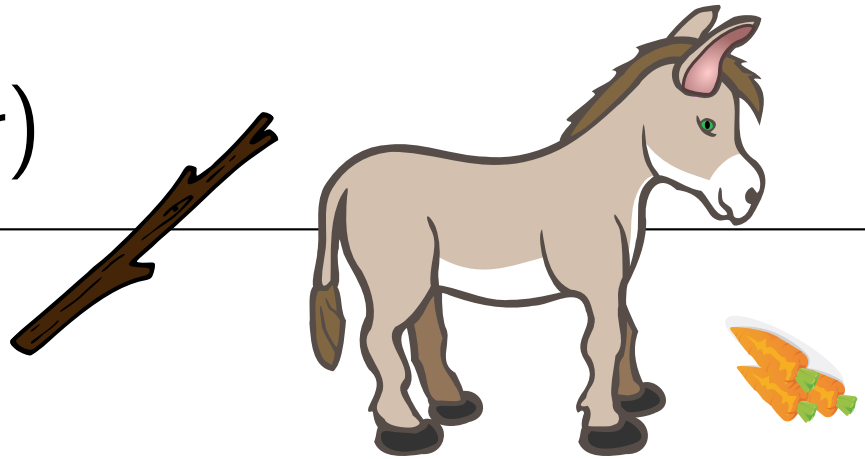
Pédagogie Montessori

Complexité

Internet

Logiciel Libres

# Théorie X (McGregor)



- l'être humain moyen n'aime pas le travail et l'évitera s'il le peut
- les gens doivent être controlés ou menacés pour qu'ils travaillent suffisamment
- l'humain moyen préfère être dirigé, il n'aime pas les responsabilités
- il désire la sécurité par dessus tout
- il ne déploie son intelligence que pour contourner les rêglements

# Théorie Y (McGregor)

- Faire des efforts physiques et mentaux au travail est aussi naturel que s'amuser et se reposer.
- Le contrôle et la punition ne sont pas les seules façons de faire travailler les gens on peut par exemple les associer aux buts de l'organisation.
- Si un travail apporte des satisfactions, alors l'engagement envers l'organisation s'améliore.
- L'homme moyen est capable d'apprendre
- Mis dans de bonnes conditions, il accepte et recherche les responsabilités.
- L'humain est motivé par le désir de se réaliser pleinement, il a besoin du travail pour se développer
- Il est préférable de laisser les gens s'auto-organiser

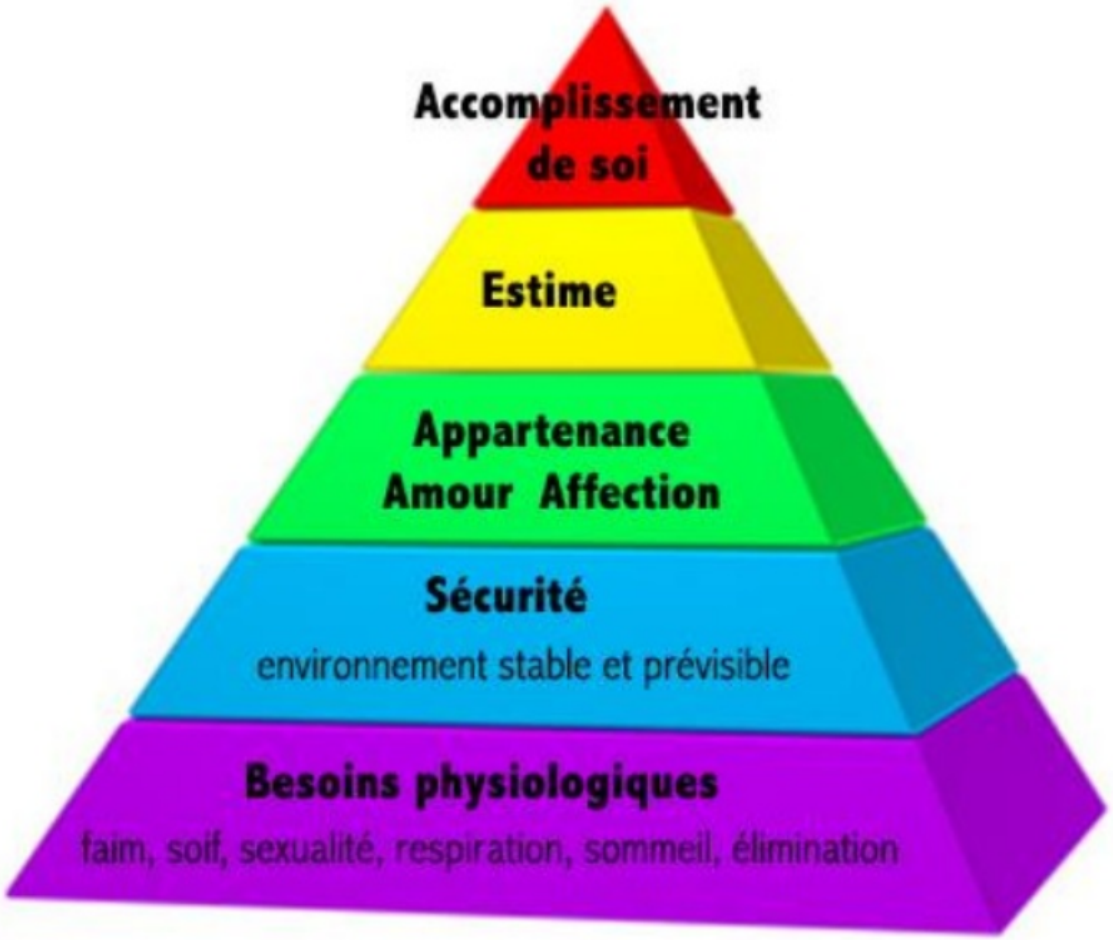
# L'agilité s'appuye sur la motivation intrinsèque

- responsabilité, droit à l'intelligence
- liberté dans certaines limites, auto-organisation des équipes
- vision partagée de l'objectif, reconnaissance du travail (démos)
- travail concret produit, participation à une oeuvre collective
- amélioration continue

# L'agilité répond aux besoins

## Pyramide de Maslow

Hiérarchie des besoins
issue de
"Théorie de la motivation humaine"
1943



# Les valeurs de la Génération X

- Expérience, Loyauté , Sens de la hiérarchie
- Conception hiérachisée de l'entreprise
- Faible capacité de communication
- Esprit de compétition
- Organisé , capable de planifier sur le long terme
- Manque de transparence
- Aversion au changement , besoin de sécurité
- Cherchent un travail valorisant, mais le travail n'est pas central

# Les valeurs de la Génération Y

- Pas de résistance au changement , forte capacité d'adaptation
- Instable, égocentrique, exigences personnelles, micro-absentéisme
- Faible engagement collectif, donnant-donnant
- Peu de respect de la hiérarchie ou de l'age, respect de la compétence
- Facilité d' apprentissage , envie d' entreprendre , mais moindre inventivité
- Ouverture d'esprit , maîtrise de la technologie

# Les valeurs de la Génération Z

- Polyvalence , Transparence , Mondialiste
- Sens des valeurs , idéalistes, originalité , exigence
- Rapidité de pensée et d'action, dispersion , impatience
- En réseau, sens du collectif , dépendance technologique
- Défiance vis-àvis de l' entreprise

# L'agilité, pour quelle génération ?

- Remplacement du modèle hiérarchique
- Porteurs des responsabilités
- Fluidité des rôles et postes
- Objectifs à court terme, reconnaissance
- Pression du collectif, objectifs d'équipe
- Cadre structurant, et amélioration continue
- Transparence, changements constants

# Correspondance Agilité et Génération X

- Remplacement du modèle hiérarchique
- Porteurs des responsabilités
- Fluidité des rôles et postes
- Objectifs à court terme, reconnaissance
- Pression du collectif, objectifs d'équipe
- Cadre structurant, et amélioration continue
- Transparence, changements constants

# Correspondance Agilité et Génération Y

- Remplacement du modèle hiérarchique
- Porteurs des responsabilités
- Fluidité des rôles et postes
- Objectifs à court terme, reconnaissance
- Pression du collectif, objectifs d'équipe
- Cadre structurant, et amélioration continue
- Transparence, changements constants

# Correspondance Agilité et Génération Z

- Remplacement du modèle hiérarchique
- Porteurs des responsabilités
- Fluidité des rôles et postes
- Objectifs à court terme, reconnaissance
- Pression du collectif, objectifs d' équipe
- Cadre structurant, et amélioration continue
- Transparence, changements constants

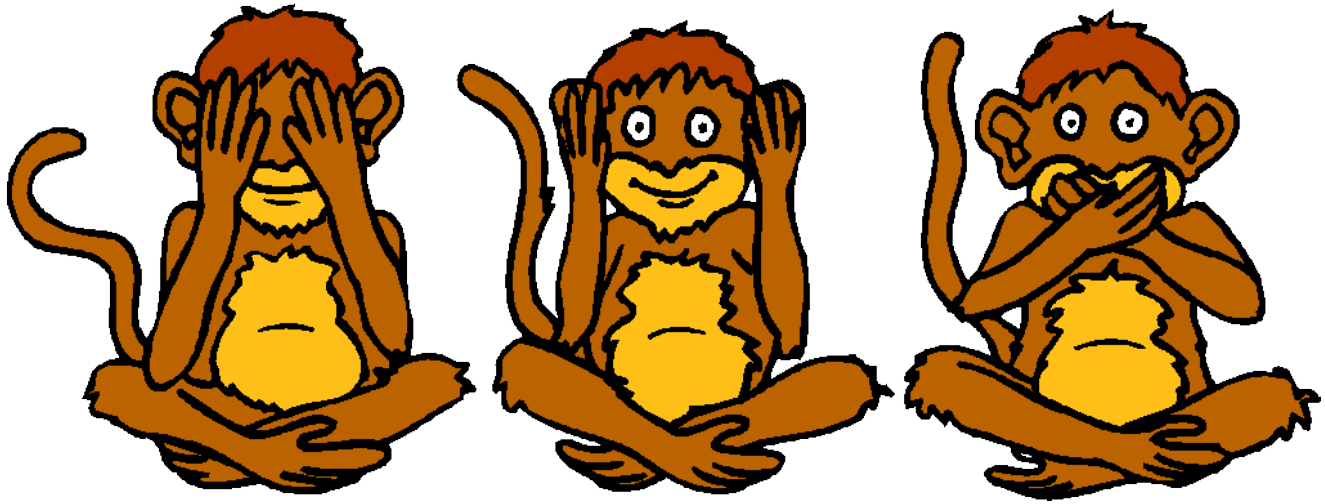
# Un paradis pour les générations Y et Z ?

$X <= Y < Z$

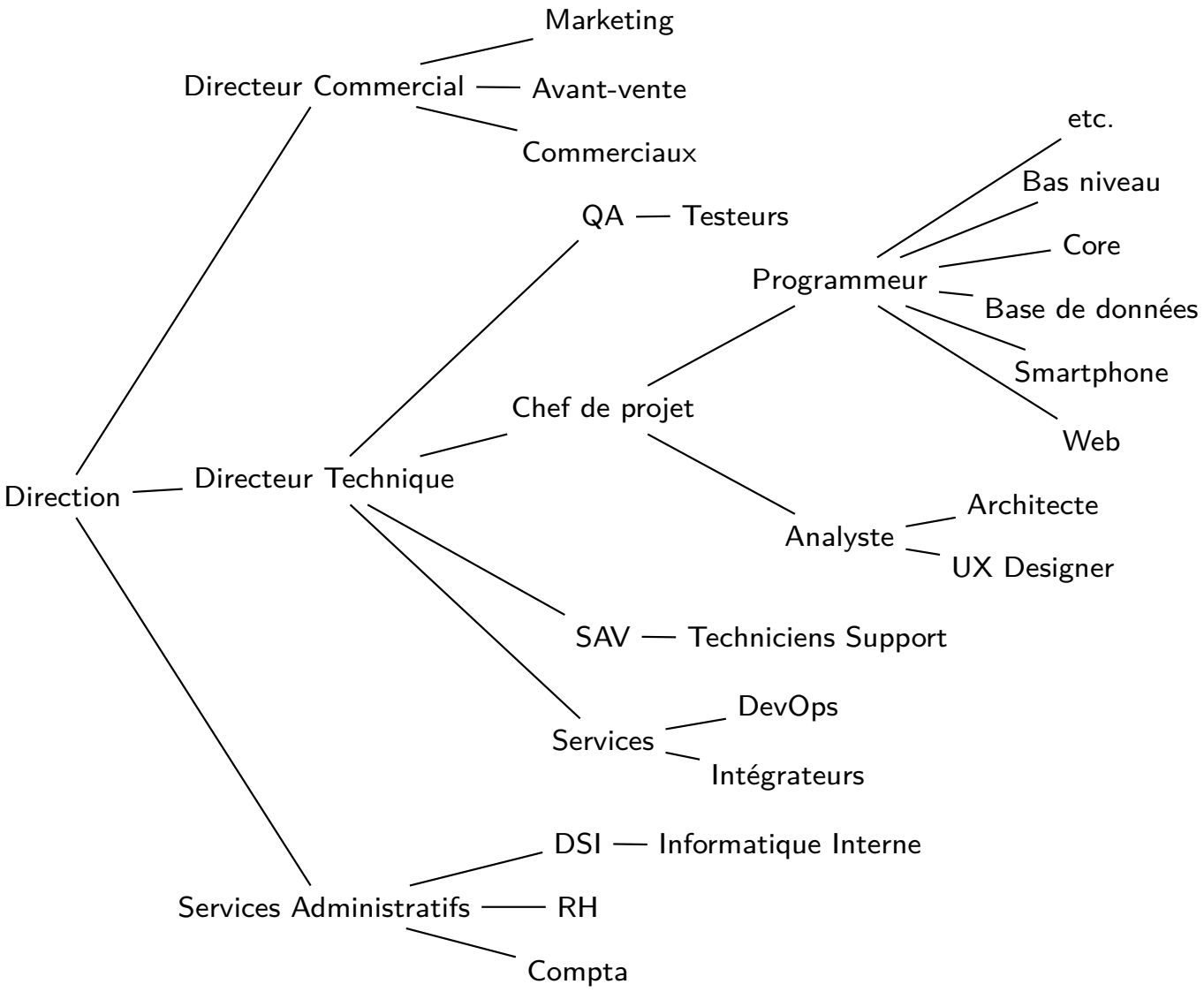
---

- L'Agilité est plus proche du mode de fonctionnement des générations Y que de celui de la génération X, très attachés à la hiérarchie et hostiles au changement.
- Son cadre structurant et la pression du collectif compensent l'individualisme et le manque de transparence de la génération Y.
- Tout les signaux sont au vert pour la génération Z

# Redistribuer les Rôles



# Rôles classiques



# Modèle Classique : une hiérarchie

- Le supérieur hiérarchique donne les ordres
- Les rôles sont spécialisés et figés
- La structure hiérarchique est figée
- L'amélioration vient d'en haut
- Contacts indirects avec le client
- Fonctionnement focalisé sur les processus et les documents

# Modèle Agile : une équipe (1)

- On n'essaye pas de décrire toute l'activité de l'entreprise
- Peu de rôles individuels spécialisés
  - Product Owner : champion du client, décide quel produit faire
  - Développeurs : décident comment réaliser les souhaits du client, indiquent le temps nécessaire.
  - Les développeurs sont polyvalents, fonctions fluides (analyste, programmeur, testeur)
  - Coach/Scrum Master garant du fonctionnement de l' équipe

# Modèle Agile : une équipe (2)

- Pas de hiérarchie au sein de l'équipe et entre rôles
- Des responsabilité différentes
- L'équipe et le Product Owner sont focalisés sur le produit réalisé
- Le Coach prend du recul et se focalise sur l' équipe

# Identifier un Agiliste en une seule Question

Quel est ton poste ?

# Identifier un Agiliste en une seule Question

## Quel est ton poste ?

- Traditionnel : Je suis Chef de Projet chez Megatech

# Identifier un Agiliste en une seule Question

## Quel est ton poste ?

- Traditionnel : Je suis Chef de Projet chez Megatech
- Agile : Je travaille sur l'Inducteur à Impulsion Megatech

## Développeur traditionnel

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

# Agile vs Traditionnel

## Développeur traditionnel

---

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

## Développeur Agile

---

# Agile vs Traditionnel

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

## Développeur Agile

- Généralistes spécialisés, en réseau

# Agile vs Traditionnel

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

## Développeur Agile

- Généralistes spécialisés, en réseau
- Auto-organisés , agissent en fonction des besoins

# Agile vs Traditionnel

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

## Développeur Agile

- Généralistes spécialisés, en réseau
- Auto-organisés , agissent en fonction des besoins
- Responsabilités personnelles, engagement
  - décident comment réaliser les souhaits du client
  - indiquent le temps nécessaire

# Agile vs Traditionnel

## Développeur traditionnel

- Spécialistes , rôle prédéfini , sortent rarement de leur périmètre
- Agissent selon les instructions de leur hiérarchie
- Rendent des comptes sur les résultats individuels

## Développeur Agile

- Généralistes spécialisés, en réseau
- Auto-organisés , agissent en fonction des besoins
- Responsabilités personnelles, engagement
  - décident comment réaliser les souhaits du client
  - indiquent le temps nécessaire
- Rendent collectivement des compte : démo

# Manager traditionnel (1)

- Rend des comptes sur les résultats à sa hiérarchie
- S'engage contractuellement (avec le client et ses chefs) sur :
  - le conformité aux spécifications
  - la date de livraison
  - le budget, les ressources nécessaires

# Manager traditionnel (2)

- Responsable hiérarchique
  - distribue le travail
  - décide de la durée des tâches
  - décide parfois de la façon de faire

# Manager traditionnel (2)

- Responsable hiérarchique
  - distribue le travail
  - décide de la durée des tâches
  - décide parfois de la façon de faire
- Les développeurs lui rendent des comptes
  - mesures de performance individuelle
  - entretien bilan objectifs annuel
  - primes

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes
- Coaching individuel : aide les programmeurs à progresser, gère les individus problématiques.

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes
- Coaching individuel : aide les programmeurs à progresser, gère les individus problématiques.
- Pas d'engagement de sa part sur le produit

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes
- Coaching individuel : aide les programmeurs à progresser, gère les individus problématiques.
- Pas d'engagement de sa part sur le produit
- Des engagements vis à vis de l' équipe et de la direction

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes
- Coaching individuel : aide les programmeurs à progresser, gère les individus problématiques.
- Pas d'engagement de sa part sur le produit
- Des engagements vis à vis de l' équipe et de la direction
- Rend des comptes à la direction (sur le fonctionnement de l'équipe)

# Coach Agile (1)

Rôle radicalement différent de celui du Manager.

- Servant Leader et Guide, au même niveau hiérarchique que l'équipe de développement
- Organise les cérémonies/réunions périodiques, les fêtes
- Coaching individuel : aide les programmeurs à progresser, gère les individus problématiques.
- Pas d'engagement de sa part sur le produit
- Des engagements vis à vis de l' équipe et de la direction
- Rend des comptes à la direction (sur le fonctionnement de l'équipe)
- Rend des comptes à l'équipe (sur le fonctionnement de l'équipe)

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution
  - protège l'équipe

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution
  - protège l'équipe
  - leur fournit les moyens de travailler

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution
  - protège l'équipe
  - leur fournit les moyens de travailler
  - mesure de performance de l'équipe

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution
  - protège l'équipe
  - leur fournit les moyens de travailler
  - mesure de performance de l'équipe
  - Force de proposition

# Coach Agile (2)

- Des engagements vis à vis de l'équipe et de la direction
  - détecte les problèmes de préférence très tôt.
  - aide à leur résolution
  - protège l'équipe
  - leur fournit les moyens de travailler
  - mesure de performance de l'équipe
  - Force de proposition
  - n'impose rien

# Client Traditionnel

- Pas représenté dans l'entreprise pendant le développement
- Défini le travail initial : Cahier des charges
- Engagement contractuel : une fois le cahier des charges défini on ne change plus rien...

# Client Traditionnel

- Pas représenté dans l'entreprise pendant le développement
- Défini le travail initial : Cahier des charges
- Engagement contractuel : une fois le cahier des charges défini on ne change plus rien... que ça lui plaise ou non
- Il verra le produit fini au moment de la recette
- Celui qui passe le contrat c'est celui qui paye, rarement le véritable utilisateur
- Pourra signaler des *bugs* une fois le projet livré

# Product Owner (1)

- Champion du client intégré à l'équipe de développement

# Product Owner (1)

- Champion du client intégré à l'équipe de développement
- Partage sa vision du produit (long terme)

# Product Owner (1)

- Champion du client intégré à l'équipe de développement
- Partage sa vision du produit (long terme)
- Gère la roadmap (prévisionnelle) à long terme

# Product Owner (1)

- Champion du client intégré à l'équipe de développement
- Partage sa vision du produit (long terme)
- Gère la roadmap (prévisionnelle) à long terme
- Défini les scénario d'utilisation (User Stories) : concrets , testables , apportant de la valeur

# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs

# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs
- Choisit quelles User Stories réaliser dans une itération selon leur poids et valeur

# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs
- Choisit quelles User Stories réaliser dans une itération selon leur poids et valeur
- Renonce (temporairement) aux User Stories trop compliquées ou trop longues

# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs
- Choisit quelles User Stories réaliser dans une itération selon leur poids et valeur
- Renonce (temporairement) aux User Stories trop compliquées ou trop longues
- Parle avec les développeurs, il est l'arbitre ultime du produit

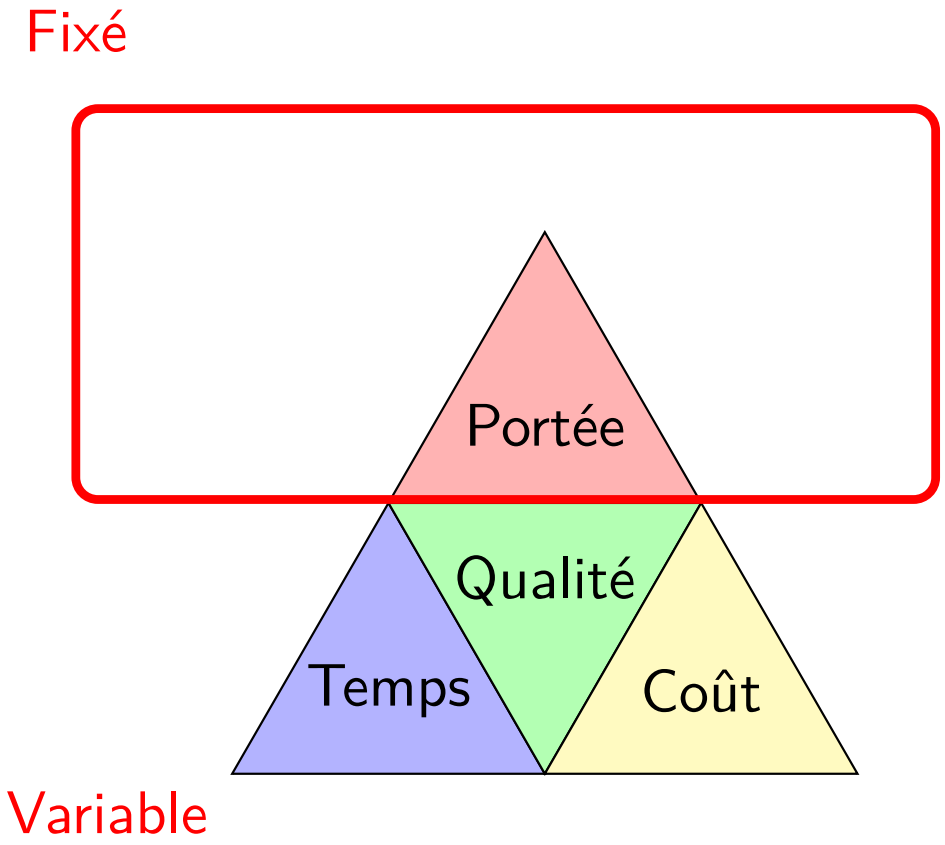
# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs
- Choisit quelles User Stories réaliser dans une itération selon leur poids et valeur
- Renonce (temporairement) aux User Stories trop compliquées ou trop longues
- Parle avec les développeurs, il est l'arbitre ultime du produit
- Assiste aux démos : en général ça marche,

# Product Owner (2)

- Écoute les évaluations des User Stories par les développeurs
- Choisit quelles User Stories réaliser dans une itération selon leur poids et valeur
- Renonce (temporairement) aux User Stories trop compliquées ou trop longues
- Parle avec les développeurs, il est l'arbitre ultime du produit
- Assiste aux démos : en général ça marche, mais est-ce que ça convient ?

# Gestion de Projet Classique



## Engagement (contractuel) sur le périmètre

- Les ressources et le temps sont variables mais évalués en début de projet)
- On est ensuite dans le triangle de fer : plus de marge de manoeuvre

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )
  - ne marche pas, on perd du temps en debug
  - résultat non livrable
  - la qualité n'est pas une variable d'ajustement.

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )
  - ne marche pas, on perd du temps en debug
  - résultat non livrable
  - la qualité n'est pas une variable d'ajustement.
- Allonger les heures de présence des programmeurs

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )
  - ne marche pas, on perd du temps en debug
  - résultat non livrable
  - la qualité n'est pas une variable d'ajustement.
- Allonger les heures de présence des programmeurs
  - ne marche pas non plus
  - un programmeur fatigué est inefficace
  - problème du turn-over

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )
  - ne marche pas, on perd du temps en debug
  - résultat non livrable
  - la qualité n'est pas une variable d'ajustement.
- Allonger les heures de présence des programmeurs
  - ne marche pas non plus
  - un programmeur fatigué est inefficace
  - problème du turn-over
- en bout de chaîne au terme du projet c'est souvent l'échec, pas de valeur client

# La deadline approche

- Négocier des avenants, tricher un peu sur le périmètre
- Essayer de gagner du temps
- Couper les angles ( qualité , peu de tests )
  - ne marche pas, on perd du temps en debug
  - résultat non livrable
  - la qualité n'est pas une variable d'ajustement.
- Allonger les heures de présence des programmeurs
  - ne marche pas non plus
  - un programmeur fatigué est inefficace
  - problème du turn-over
- en bout de chaîne au terme du projet c'est souvent l'échec, pas de valeur client
- Le découpage par lots fonctionne mieux,
  mais attention à faire de l' incrémental , par de l' itératif

# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :

# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"

# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"
- "les spécifications ne sont pas assez complètes/précises"

# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"
- "les spécifications ne sont pas assez complètes/précises"
- "les développeurs ne testent pas assez leur code avant de l'envoyer à la QA"

# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"
- "les spécifications ne sont pas assez complètes/précises"
- "les développeurs ne testent pas assez leur code avant de l'envoyer à la QA"
- "je ne connais pas la technologie utilisée par l'autre équipe, l'autre programmeur"

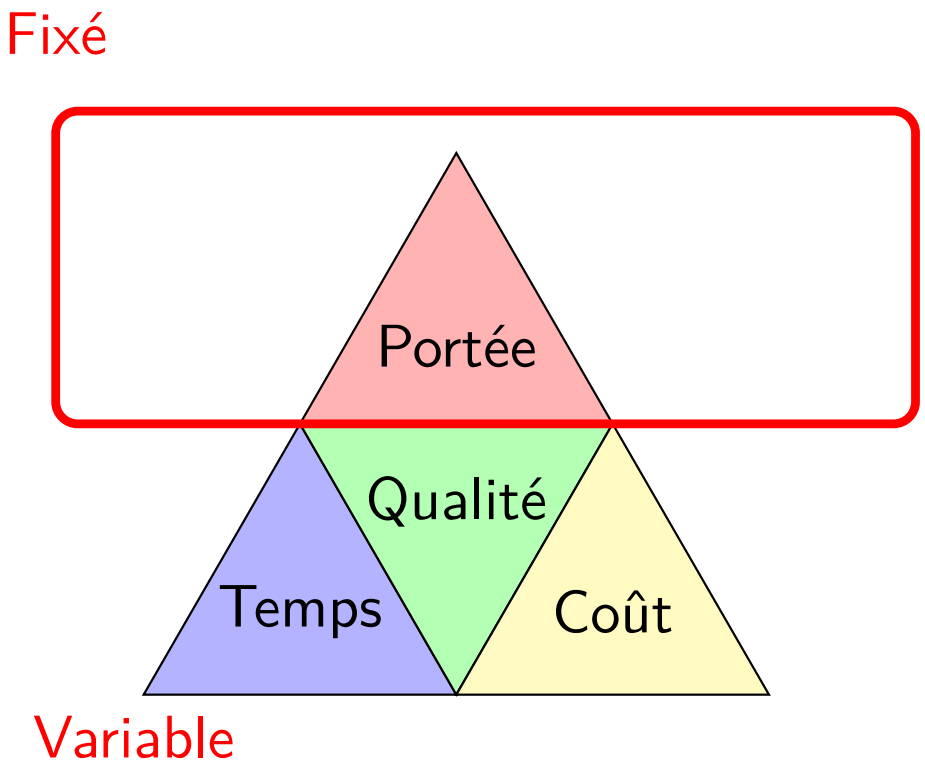
# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"
- "les spécifications ne sont pas assez complètes/précises"
- "les développeurs ne testent pas assez leur code avant de l'envoyer à la QA"
- "je ne connais pas la technologie utilisée par l'autre équipe, l'autre programmeur"
- "le manager a fixé un délai trop court"

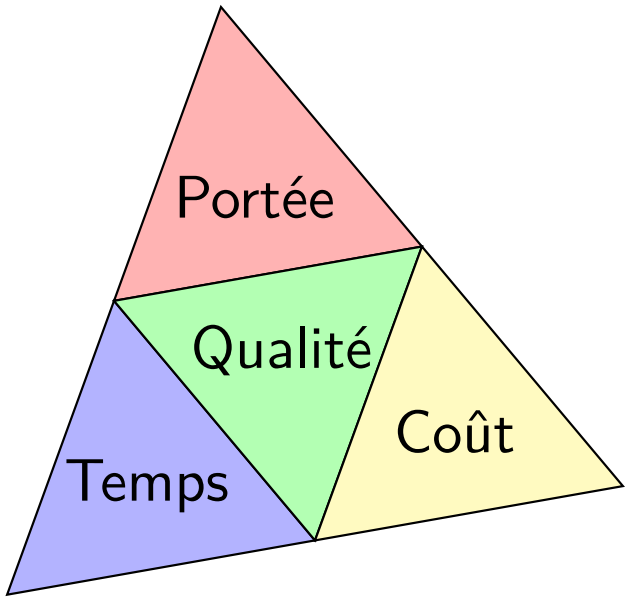
# C'est pas ma faute !

- Chacun essaye de rejeter la faute sur les autres :
- "ce n'est pas un bug c'est ce qui était prévu, c'est l'analyse qui est mauvaise"
- "les spécifications ne sont pas assez complètes/précises"
- "les développeurs ne testent pas assez leur code avant de l'envoyer à la QA"
- "je ne connais pas la technologie utilisée par l'autre équipe, l'autre programmeur"
- "le manager a fixé un délai trop court"

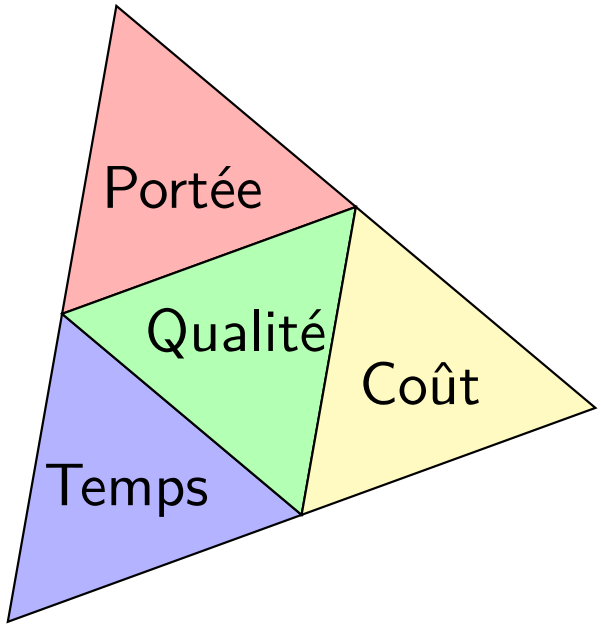
# Du mode projet au mode incrémental



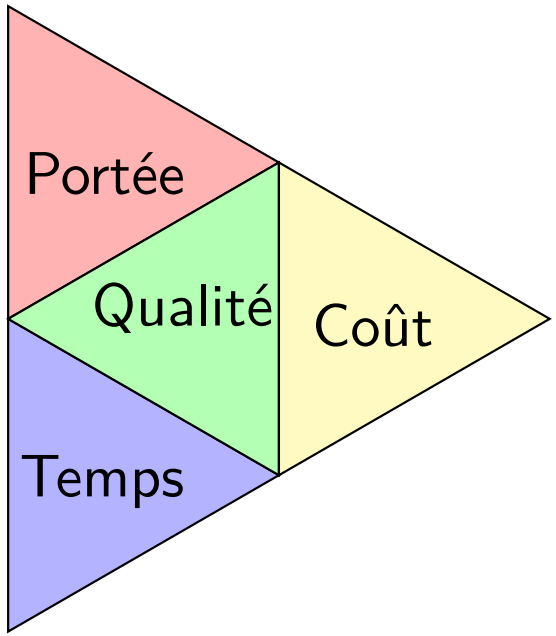
# Du mode projet au mode incrémental



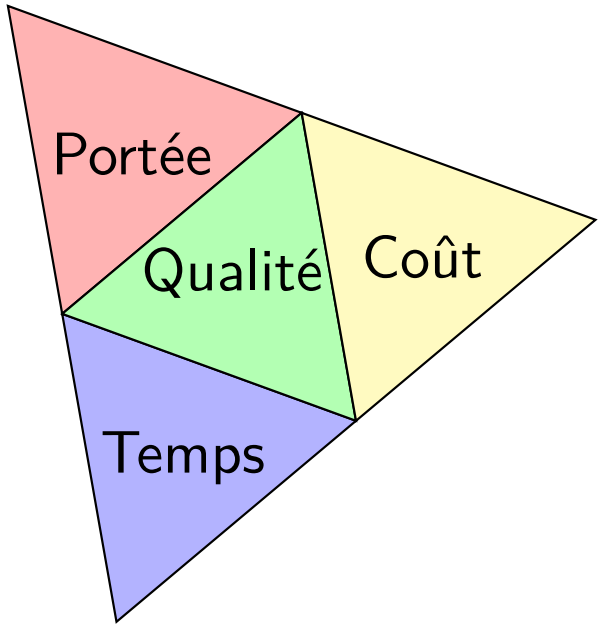
# Du mode projet au mode incrémental



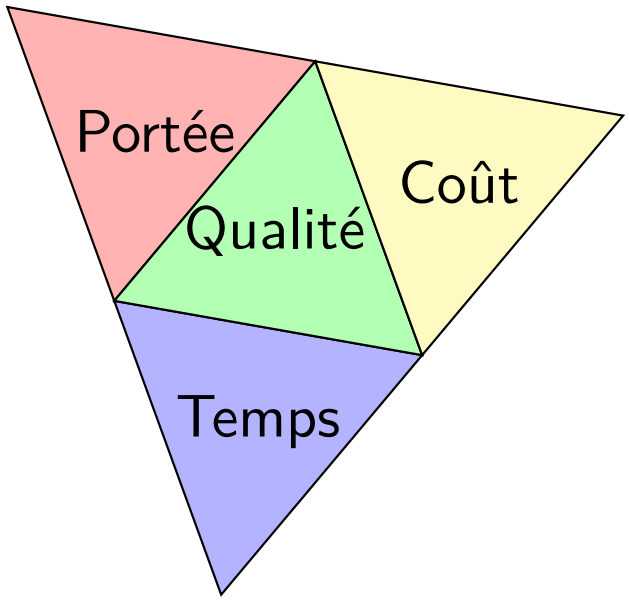
# Du mode projet au mode incrémental



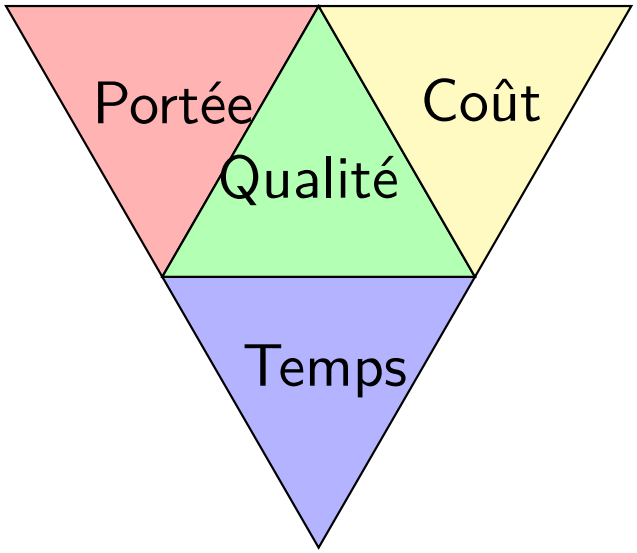
# Du mode projet au mode incrémental



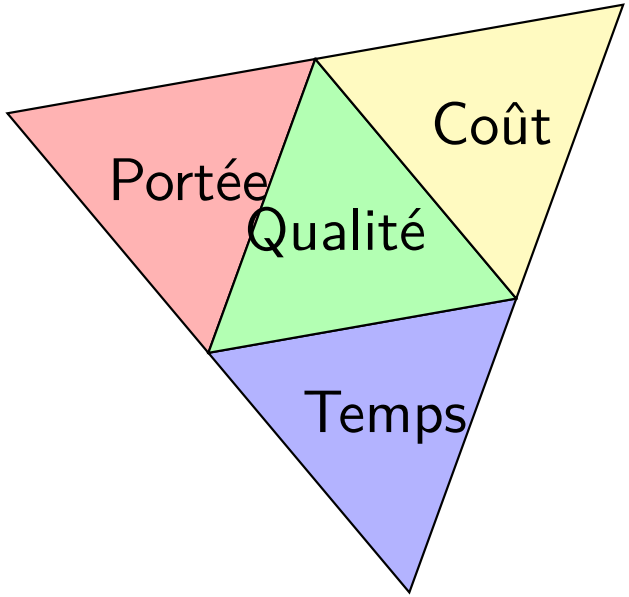
# Du mode projet au mode incrémental



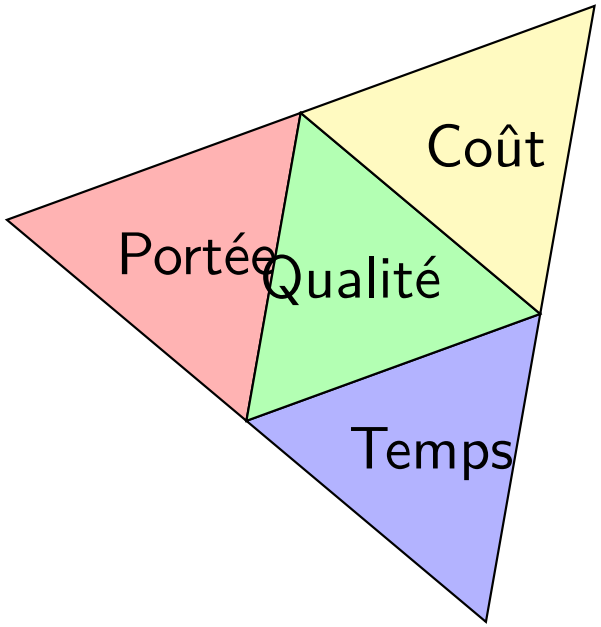
# Du mode projet au mode incrémental



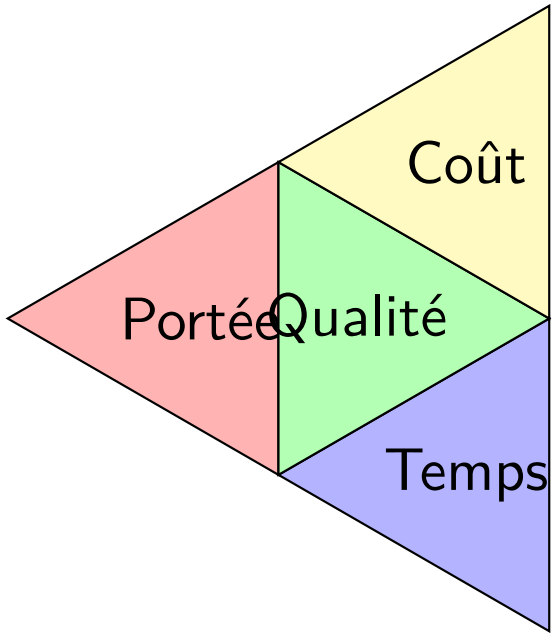
# Du mode projet au mode incrémental



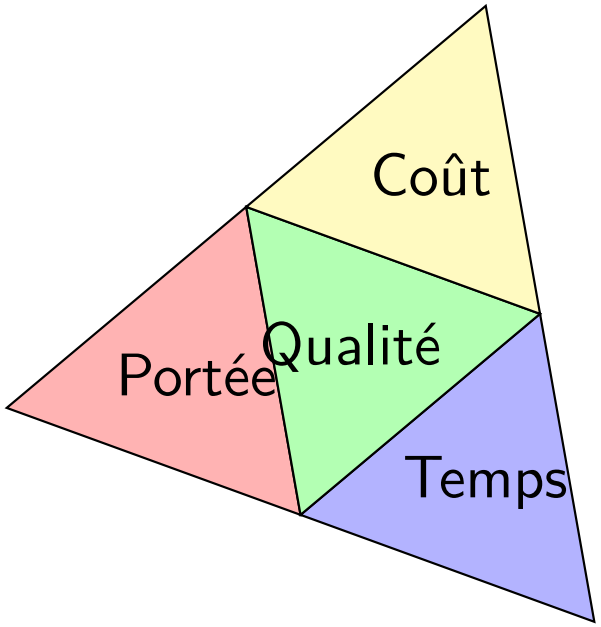
# Du mode projet au mode incrémental



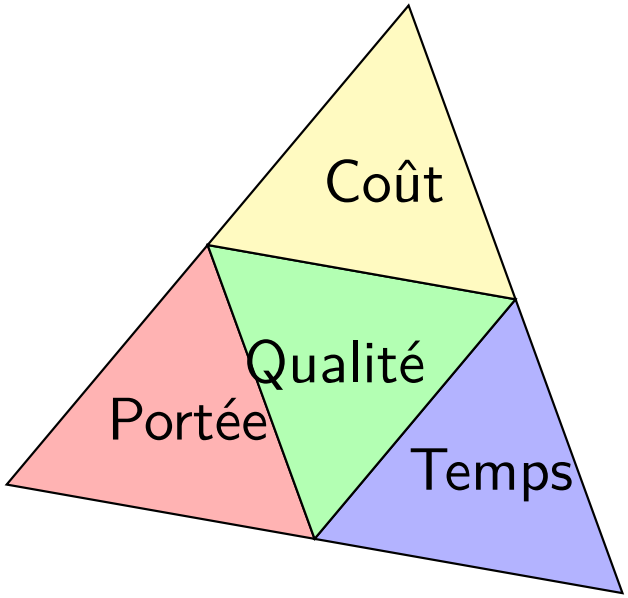
# Du mode projet au mode incrémental



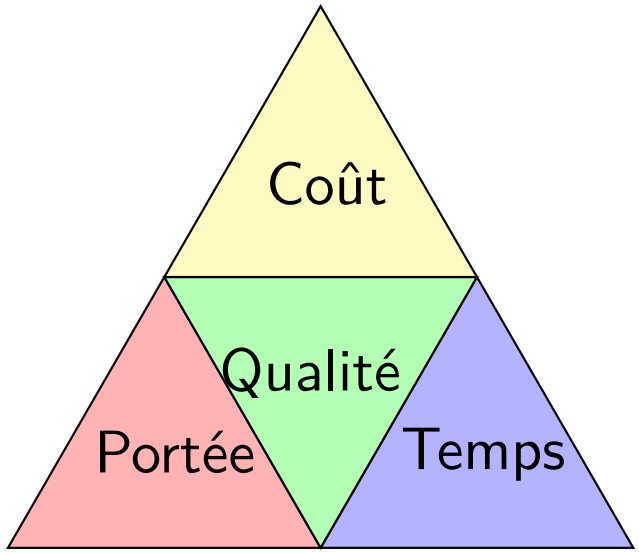
# Du mode projet au mode incrémental



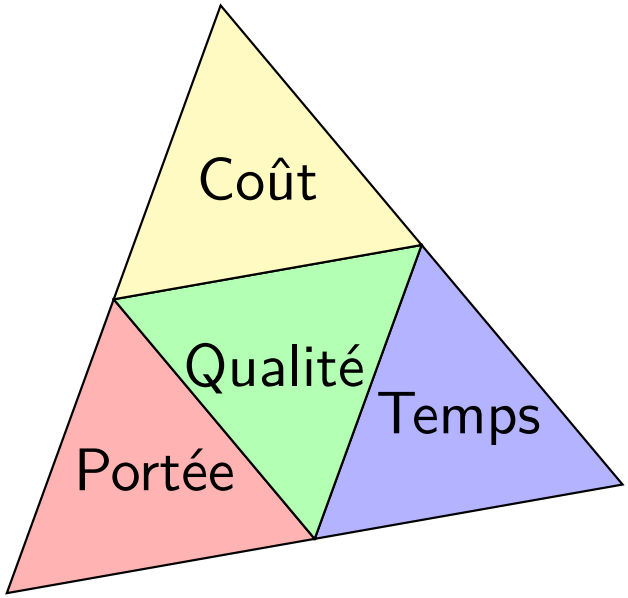
# Du mode projet au mode incrémental



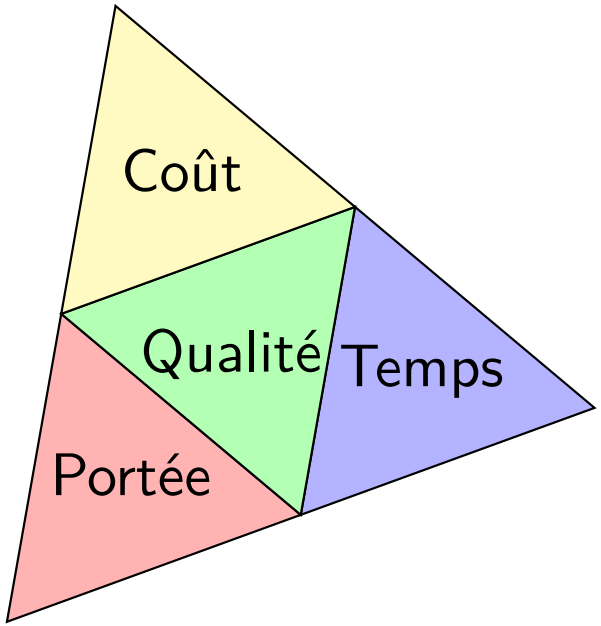
# Du mode projet au mode incrémental



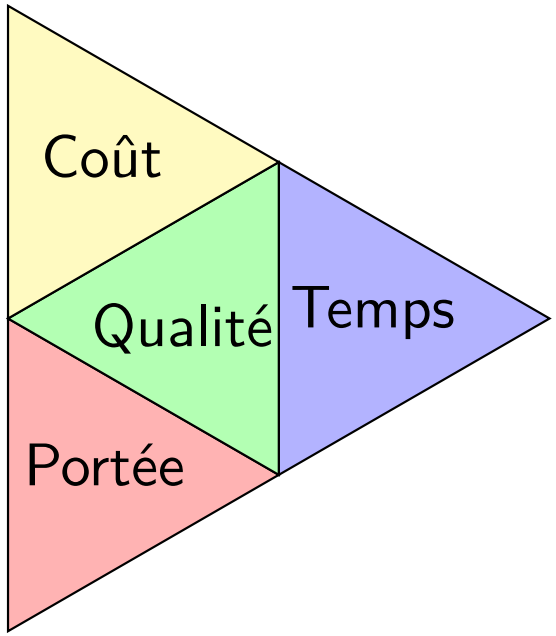
# Du mode projet au mode incrémental



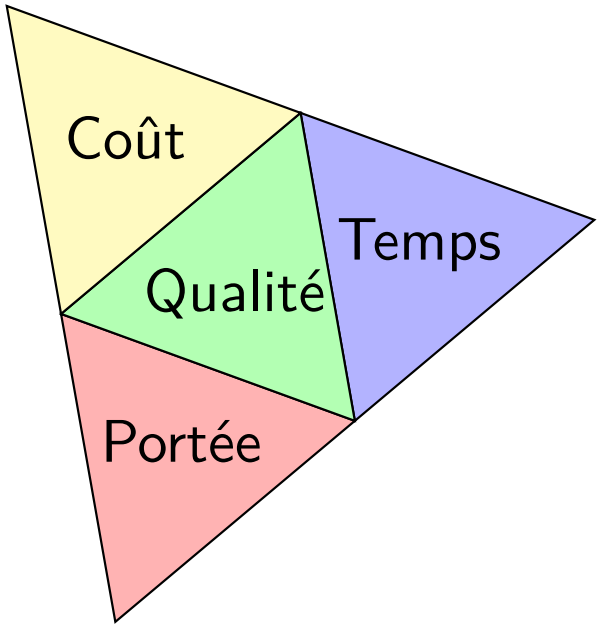
# Du mode projet au mode incrémental



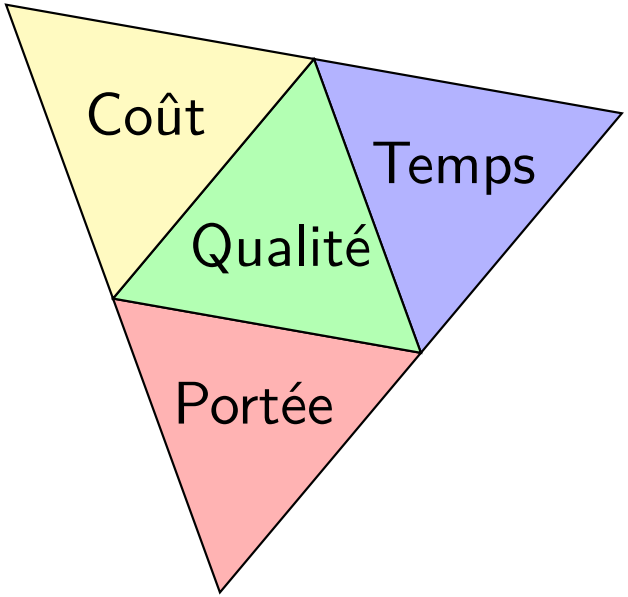
# Du mode projet au mode incrémental



# Du mode projet au mode incrémental

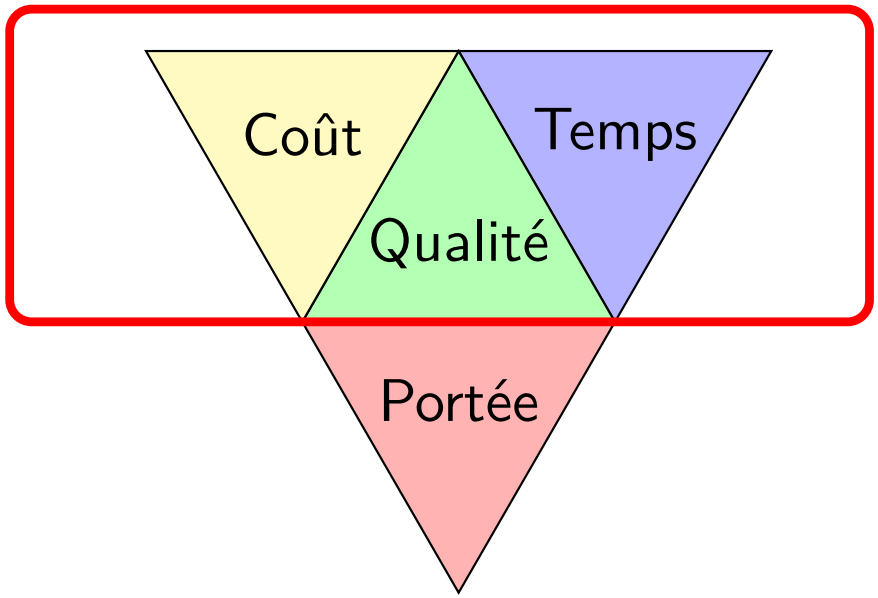


# Du mode projet au mode incrémental



# Du mode projet au mode incrémental

Fixé




Variable

# Développement Incrémental

- Le périmètre fonctionnel est variable, choisi au départ de l'itération
- Les ressources et le temps disponibles sont fixes.
- On travaille à court terme plutôt qu'à long terme...
- ...par petits incréments, typiquement de moins d'un mois.
- Démo aux parties prenantes en fin d'itération
- Chaque itération doit apporter de la valeur.

# L'échec est normal et souhaitable

- Ne pas réussir à tenir parfaitement les objectifs de l'itération est normal
- Si celà n'arrive jamais c'est que l'équipe ne s'engage pas assez et reste dans sa zone de confort.
- L'équipe apprend de ses échecs, elle n'est pas sanctionnée, donc peut prendre des risques.
- On peut aussi avoir de bonnes surprises et que l'équipe réalise plus de travail que prévu.
- L'itération doit tout de même fournir de la valeur...

## Gestion du risque

- Les difficultés sont identifiées rapidement : une itération.
- Possibilité de management par les risques : commencer par les étapes risquées ou douteuses
- On peut insister si c'est important, ou chercher une alternative
- Parfois on peut se rendre compte lors de la démo qu'un scénario réalisé n'a pas de valeur

# L'équipe apprend de ses échecs

- Ne pas miser sur un seul cheval : plusieurs scénarios en même temps (mais pas trop)
- Mauvaise évaluation a priori de la durée de certains scénarios : responsabilité des programmeurs
- La vélocité de l'équipe (la quantité de travail d'une itération) est révisée
- Scénario mal défini, produit mauvais : responsabilité du product owner
- Engagement de l'équipe insuffissant, manque de compétence de l'équipe, autres éléments perturbateurs : responsabilité du coach

# L'Agilité dans la pratique

- En théorie une équipe Agile est totalement équivalente : n'importe quel membre peut tenir n'importe quel rôle
- C'est rarement vrai dans la pratique, l'équipe s'organise avec des rôles spécialisés souvent proches des rôles traditionnels.
- L'équipe est polyvalente, pas les individus, même s'il y a transfert de compétences sur le moyen terme
- Par contre on supprime les barrières artificielles liées aux définitions de poste spécialisées
- Les individus travaillent dans leur domaine de compétence si c'est utile à l'équipe
- Ou apprennent de nouvelles compétences

# Manager ou Chef de Projet

- potentiellement un bon Product Owner, s'il peut se mettre à la place du client
- en tant que Chef de Projet Agile, peut prendre en charge la gestion des indicateurs d'avancement du produit : BurnDown Chart, Vélocité, Master Story List.
- planification des tâches hors User Story : dette technique, infrastructure, correction de bugs et prise en compte des remontées clients.
- Pas de micro management des développeurs ! Ne doit pas faire obstacle vis à vis du client ou du Product Owner

# Équipe QA

- Dans l'idéal totalement intégrée à l'équipe de développement.
- Le test n'est pas une activité séparée
- En amont : définition des tests pour les User Stories
- En aval : vérification que les tests passent
- Écriture de tests automatisés
- Garant du sentiment de réussite : dans l'idéal un test commence par échouer puis passe au vert

# Analyste programmeur, Architecte logiciel

- Celà fait partie des tâches de développement, le besoin est toujours là peut réaliser la même activité (Analyse, Architecture) au sein de l'équipe. Peut aussi former des collègues ou travailler en binomes avec eux
- Souvent les Analystes ou les Architectes savent et aiment coder : ce sont des développeurs confirmés obligés de changer de poste à cause de leur expérience.
- S'il a la tournure d'esprit adéquate peut tenir le rôle de Product Owner

# Peaux de Banane

# Product Owner et développeur ?

- La responsabilité du PO est de dire quoi faire, celle du développeur de dire comment le faire et le temps nécessaire
- On court le risque qu'une même personne dise quoi faire, comment le faire et le temps nécessaire.
- Ou de choisir des solutions simples (à faire) convenant mal au client. Le comment faire prend alors le pas sur le quoi faire
- Difficile pour le PO de maintenir une Vision indépendante de la mise en oeuvre.
- Dépend des activités : problématique pour un rôle d'Analyse, moins pour un rôle de programmeur, pas de conflit avec un rôle de testeur.
- mais ça veut surtout dire que le PO peut aussi assumer le rôle de testeur

# Coach et Product Owner ?

- Le rôle du Coach concerne le changement et l'évolution de l'équipe, et les protéger des pressions externes pour qu'ils puissent travailler sereinement.
- Le Product Owner a la pression des résultats : les deux sont directement en conflit
- Un coach neutre peut dire à un développeur qu'il ne s'engage pas assez... s'il est PO le programmeur pensera qu'on parle de ses résultats individuels
- L'équipe ne verra plus le coach comme neutre, l'un des rôles va s'effacer devant l'autre
- Le cas échéant on peut prendre un Product Owner externe à l'équipe (avant-vente, commercial ou marketing par exemple), éventuellement à temps partiel

# Coach et développeur ?

- Le coach Agile est un Servant Leader et un agent de changement.
- Il est souhaitable que la personne tenant le rôle de Coach ait de l'expérience
- Il est compliqué pour un Coach de maintenir une position neutre et du recul s'il participe au développement
- Il y a aussi conflit sur l'utilisation du temps pour programmer ou assumer les responsabilités de Coach
- Pour l'Agiliste les réunions du Coach Agile peuvent être vus comme une perte de temps

# Hiérarchie interne ?

- En théorie tous les programmeurs sont identiques et polyvalents
- hiérarchie de fait basée sur l' expérience et les domaines de compétence .
- peuvent aussi avoir des salaires différents (ancienneté, profil)
- il est préférable que les primes soient collectives et partagées (et non annoncées, après une vraie réussite).

# Modèle hybride : Chef de Projet Agile ?

- Possible, mais le *Chef de Projet doit veiller à se focaliser sur le produit*
  - l'équipe décide quoi faire et indique le poids/temps
  - l'équipe prends des décisions de manière autonome.
  - l'équipe dialoguer directement avec le Coach et le Product Owner.

# L'Agilité qui vient d'en haut

- Parfois l'entreprise veut passer à l'agilité pour les résultats...
- mais n'est pas prête à renoncer au modèle hiérarchique.
- ni à donner aux programmeurs le degré de liberté dont ils ont besoins.
- Un fonctionnement agile a de nombreux impacts
  - sur la chaine de prise de décision
  - sur les interlocuteurs
  - sur la négociation des contrats
  - sur la présentation de la Roadmap à l'extérieur
  - sur les process internes
  - sur les outils utilisés

# Le Coach en tant que chef ?

- Attention, si le Coach est un supérieur hiérarchique on retombe facilement sur les modèles classique de déresponsabilisation.
- Il est compliqué pour un Coach de maintenir une position neutre et du recul dans ce cas de figure.
- L'équipe doit se voir comme autonome pour vraiment s'investir et accepter des responsabilités
- C'est souvent un consultant externe spécialisé (Scrum Master) dans ce cas il est de fait hors hiérarchie

# Pas de Coach expérimenté disponible ?

Faute de Coach il faut que l'équipe de développer et le Product Owner assument son rôle. Pas idéal mais possible.

- Les responsabilités du Coach sont abondamment décrites sur Internet, celà peut remplacer l'expérience
- Coach peut éventuellement être un rôle tournant, par exemple pour l'organisation des réunions, permet d'impliquer l'équipe.
- Le Coach est potentiellement un rôle transitoire si l'équipe réussit vraiment à s'auto gérer.
- Il est aussi possible dans certains cas de partager un Coach entre plusieurs équipes

# Pas de Product Owner disponible ?

Il faut absolument en trouver un ! Allez débaucher dans le reste de l'entreprise !

- un commercial ou un responsable marketing peuvent éventuellement tenir le Rôle
- ce n'est pas forcément un rôle à temps complet (mais il y a du boulot)
- ou même un Chef de Projet ou un testeur QA expérimenté
- L'équipe et le Coach peuvent aider à formaliser les User Stories
- Mais il faut absolument qu'il se mette *dans les bottes du client*

# Difficulté du lâcher-prise

- L'une des grosses difficulté de l'agilité pour le manager c'est de faire confiance
- il ne faut pas donner des instructions mais un objectif et faire confiance
  - au Product Owner pour définir progressivement le bon produit
  - aux développeurs pour qu'ils travaillent sans qu'on les surveille, ne trichent pas sur les temps et s'engagent
  - au Coach pour qu'il motive les uns et les autres et les aide à s'organiser

# Difficulté du lâcher-prise (2)

- La moitié du travail d'un Coach Agile consiste à faire lacher prise à la hiérarchie...

# Difficulté du lâcher-prise (2)

- La moitié du travail d'un Coach Agile consiste à faire lacher prise à la hiérarchie...
- ...l'autre moitié consiste à lutter contre les résistances au changement de l'équipe.

# Difficulté du lâcher-prise (2)

- La moitié du travail d'un Coach Agile consiste à faire lacher prise à la hiérarchie...
- ...l'autre moitié consiste à lutter contre les résistances au changement de l'équipe.
- L'équipe doit apprendre à s'organiser et décider, pas attendre des instructions

# Difficulté du lâcher-prise (2)

- La moitié du travail d'un Coach Agile consiste à faire lacher prise à la hiérarchie...
- ...l'autre moitié consiste à lutter contre les résistances au changement de l'équipe.
- L'équipe doit apprendre à s'organiser et décider, pas attendre des instructions
- Les managers doivent faire confiance à l'équipe et la laisser s'organiser.

# Danger des indicateurs

- Les équipes agiles créent quantité d'indicateurs : burn-down chart, vélocité, kanban, niko-niko, etc.
- Ces indicateurs ne sont pas destinés au management, mais à usage interne.
- Eviter surtout les évaluations individuelles (pire : avec primes)
- Ou les membres de l'équipe agile risquent de travailler en concurrence les uns avec les autres.
- Effet catastrophique garanti.

# L'agilité est souvent un *Cargo Cult*

# L'agilité est souvent un *Cargo Cult*

- On peut faire semblant d'être agile sans l'être,
  le recours à des Scrum Master inexpérimentés conduit parfois à cette situation :

adopter les rituels et les pratiques agiles
mais sans changements de fond.

# L'agilité est souvent un *Cargo Cult*

- On peut faire semblant d'être agile sans l'être,
  le recours à des Scrum Master inexpérimentés conduit parfois à cette situation :

  adopter les rituels et les pratiques agiles
  mais sans changements de fond.

- Danger en cas de mise en place des pratiques manageriales agiles sans les pratiques techniques ni les degrés de liberté
- Fort risque de micro-management.
- Donne une image très négative de l'agilité à certains Senior Développeurs.

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap
- tous les travaux réalisés sont attachés à une User Story ce qui permet une visibilité optimale sur l'activité des équipes pour le Product Owner

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap
- tous les travaux réalisés sont attachés à une User Story ce qui permet une visibilité optimale sur l'activité des équipes pour le Product Owner
- les User Stories sont définies au début de la phase d'Analyse et placées sur la Roadmap

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap
- tous les travaux réalisés sont attachés à une User Story ce qui permet une visibilité optimale sur l'activité des équipes pour le Product Owner
- les User Stories sont définies au début de la phase d'Analyse et placées sur la Roadmap
- l'équipe utilise un logiciel de tableau de bord Agile (Rally/VersionOne/Jira/etc.), les pratiques sont adaptées au logiciel

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap
- tous les travaux réalisés sont attachés à une User Story ce qui permet une visibilité optimale sur l'activité des équipes pour le Product Owner
- les User Stories sont définies au début de la phase d'Analyse et placées sur la Roadmap
- l'équipe utilise un logiciel de tableau de bord Agile (Rally/VersionOne/Jira/etc.), les pratiques sont adaptées au logiciel
- une partie des équipes a suivi une formation à l'Agilité.

# Agile en toc

Les exemples suivant partent d'une bonne intention, mais risquent d'avoir peu d'effet si on en reste là, si ce n'est alourdir le process.

- derrière le bureau du chef il y a un Kanban couvert de post-its montrant l'avancée de la Roadmap
- tous les travaux réalisés sont attachés à une User Story ce qui permet une visibilité optimale sur l'activité des équipes pour le Product Owner
- les User Stories sont définies au début de la phase d'Analyse et placées sur la Roadmap
- l'équipe utilise un logiciel de tableau de bord Agile (Rally/VersionOne/Jira/etc.), les pratiques sont adaptées au logiciel
- une partie des équipes a suivi une formation à l'Agilité.
- l'équipe est encadrée par des Coachs Agiles, des Scrum Masters et des Product Owners.

# Agile en toc !

Encore quelques exemples...

- on travaille en itérations d'une semaine et on respecte les trois réunions principales : début d'itération, le Scrum Quotidien et la Rétrospective.

# Agile en toc !

Encore quelques exemples...

- on travaille en itérations d'une semaine et on respecte les trois réunions principales : début d'itération, le Scrum Quotidien et la Rétrospective.
- on définit une vélocité qui permet de connaître l'efficacité de l'équipe et l'efficacité respective de chaque membre

# Agile en toc !

Encore quelques exemples...

- on travaille en itérations d'une semaine et on respecte les trois réunions principales : début d'itération, le Scrum Quotidien et la Rétrospective.
- on définit une vélocité qui permet de connaître l'efficacité de l'équipe et l'efficacité respective de chaque membre
- le Scrum Master est l'interlocuteur privilégie pour les demandes faites à l'équipe, cela fait partie de son rôle de protecteur de l'équipe

# Agile en toc !

Encore quelques exemples...

- on travaille en itérations d'une semaine et on respecte les trois réunions principales : début d'itération, le Scrum Quotidien et la Rétrospective.
- on définit une vélocité qui permet de connaître l'efficacité de l'équipe et l'efficacité respective de chaque membre
- le Scrum Master est l'interlocuteur privilégie pour les demandes faites à l'équipe, cela fait partie de son rôle de protecteur de l'équipe
- on définit des User Stories, elles sont estimées en heures, ce qui permet d'être plus précis sur la date de livraison.

# Sabotagile !

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous défendons le Status Quo et visons à défendre nos carrières, les experts de l'industrie ont découvert les techniques suivantes permettant de simuler l'agilité sans rien changer en pratique :

# Sabotagile !

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous défendons le Status Quo et visons à défendre nos carrières, les experts de l'industrie ont découvert les techniques suivantes permettant de simuler l'agilité sans rien changer en pratique :

1. L'Engagement contractuel via des spécifications, des dates et des coûts prévus plutôt que les individus ou les interactions.

# Sabotagile !

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous défendons le Status Quo et visons à défendre nos carrières, les experts de l'industrie ont découvert les techniques suivantes permettant de simuler l'agilité sans rien changer en pratique :

1. L'Engagement contractuel via des spécifications, des dates et des coûts prévus plutôt que les individus ou les interactions.
2. Le suivi des projets à partir de diagrammes de Gantt, de pourcentages de complétion et de tableaux de bord synthétiques plutôt qu'un logiciel qui marche.

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous défendons le Status Quo et visons à défendre nos carrières, les experts de l'industrie ont découvert les techniques suivantes permettant de simuler l'agilité sans rien changer en pratique :

1. L'Engagement contractuel via des spécifications, des dates et des coûts prévus plutôt que les individus ou les interactions.
2. Le suivi des projets à partir de diagrammes de Gantt, de pourcentages de complétion et de tableaux de bord synthétiques plutôt qu'un logiciel qui marche.
3. La définition de processus rigoureux avec des Dates de Livraison, des Recettes Partielles et des traces auditables plutôt que la collaboration avec le client.

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous défendons le Status Quo et visons à défendre nos carrières, les experts de l'industrie ont découvert les techniques suivantes permettant de simuler l'agilité sans rien changer en pratique :

1. L'Engagement contractuel via des spécifications, des dates et des coûts prévus plutôt que les individus ou les interactions.
2. Le suivi des projets à partir de diagrammes de Gantt, de pourcentages de complétion et de tableaux de bord synthétiques plutôt qu'un logiciel qui marche.
3. La définition de processus rigoureux avec des Dates de Livraison, des Recettes Partielles et des traces auditables plutôt que la collaboration avec le client.
4. Empêcher l'émergence du chaos et bloquer tout changement plutôt que l'adaptation au changement.

# Sabotagile ! (2)

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous reconnaissons que les méthodes de développement Agile fonctionnent peut-être pour d'autres et nous acceptons d'utiliser son vocabulaire car c'est la mode du jour, mais en aucun cas nous ne changerons de comportement.

# Sabotagile ! (2)

## Manifeste pour le sabotage du développement logiciel Agile (Parodie)

---

Nous reconnaissons que les méthodes de développement Agile fonctionnent peut-être pour d'autres et nous acceptons d'utiliser son vocabulaire car c'est la mode du jour, mais en aucun cas nous ne changerons de comportement.

Nous sommes des gens sérieux , nous construisons et déployons des logiciels dans le monde réel à notre manière depuis des décennies et nous avons la ferme intention de continuer à faire exactement pareil à l'avenir que ça soit efficace ou pas.

Traduction personnelle, liens vers le contenu original :
http ://smoothapps.com/index.php/2017/01/sabotagile-waste-loss-challenge
http ://smoothapps.com/index.php/2016/11/sabotagile-quotient
Novembre 2016

# Principes sous-jacents du Manifeste 1-4

1. Notre priorité absolue c'est que nos ressources informatiques livrent les fonctionnalités promises , auxquelles nous nous sommes contractuellement engagés , dans les temps et sans dépassement de budget.

# Principes sous-jacents du Manifeste 1-4

1. Notre priorité absolue c'est que nos ressources informatiques livrent les fonctionnalités promises , auxquelles nous nous sommes contractuellement engagés , dans les temps et sans dépassement de budget.
2. Tout écart par rapport au plan initial doit être repéré et réprimé. Nous obligerons les développeurs à tenir leurs engagements sans décaler les livraisons prévues.

# Principes sous-jacents du Manifeste 1-4

1. Notre priorité absolue c'est que nos ressources informatiques livrent les fonctionnalités promises , auxquelles nous nous sommes contractuellement engagés , dans les temps et sans dépassement de budget.
2. Tout écart par rapport au plan initial doit être repéré et réprimé. Nous obligerons les développeurs à tenir leurs engagements sans décaler les livraisons prévues.
3. Nous pouvons augmenter l'efficacité des développeurs et réaliser des économies d'échelle en livrant moins souvent des produits plus riches fonctionnellement. Les livraisons intermédiaires sont inutiles.

# Principes sous-jacents du Manifeste 1-4

1. Notre priorité absolue c'est que nos ressources informatiques livrent les fonctionnalités promises , auxquelles nous nous sommes contractuellement engagés , dans les temps et sans dépassement de budget.
2. Tout écart par rapport au plan initial doit être repéré et réprimé. Nous obligerons les développeurs à tenir leurs engagements sans décaler les livraisons prévues.
3. Nous pouvons augmenter l'efficacité des développeurs et réaliser des économies d'échelle en livrant moins souvent des produits plus riches fonctionnellement. Les livraisons intermédiaires sont inutiles.
4. Toujours passer par des intermédiaire entre les commerciaux et les développeurs. Les commerciaux peuvent ainsi se concentrer sur les clients et accomplir un vrai travail.

# Principes sous-jacents du Manifeste 5-8

5. Utiliser le talent de prévision des managers, les objectifs de performances individuels et les courbes de Gauss pour maximiser la productivité des ressources.

# Principes sous-jacents du Manifeste 5-8

5. Utiliser le talent de prévision des managers, les objectifs de performances individuels et les courbes de Gauss pour maximiser la productivité des ressources.
6. Le mode privilégié de communication avec les équipes de développement passe par des spécifications écrites , des documents de référence et les e-mails. Ceux-ci laissent des traces auditables et forcent les développeurs à tenir leurs engagements.

# Principes sous-jacents du Manifeste 5-8

5. Utiliser le talent de prévision des managers, les objectifs de performances individuels et les courbes de Gauss pour maximiser la productivité des ressources.
6. Le mode privilégié de communication avec les équipes de développement passe par des spécifications écrites , des documents de référence et les e-mails. Ceux-ci laissent des traces auditables et forcent les développeurs à tenir leurs engagements.
7. La mesure des progrès réalisés c'est le tableau de bord synthétique de la direction qui répertorie les fonctionnalités prévues mises dans le produit.

# Principes sous-jacents du Manifeste 5-8

5. Utiliser le talent de prévision des managers, les objectifs de performances individuels et les courbes de Gauss pour maximiser la productivité des ressources.
6. Le mode privilégié de communication avec les équipes de développement passe par des spécifications écrites , des documents de référence et les e-mails. Ceux-ci laissent des traces auditables et forcent les développeurs à tenir leurs engagements.
7. La mesure des progrès réalisés c'est le tableau de bord synthétique de la direction qui répertorie les fonctionnalités prévues mises dans le produit.
8. Les développeurs travaillent tard le soir , les week-ends et annulent leurs congés pour tenir les deadlines .

# Principes sous-jacents du Manifeste 5-8

5. Utiliser le talent de prévision des managers, les objectifs de performances individuels et les courbes de Gauss pour maximiser la productivité des ressources.
6. Le mode privilégié de communication avec les équipes de développement passe par des spécifications écrites , des documents de référence et les e-mails. Ceux-ci laissent des traces auditables et forcent les développeurs à tenir leurs engagements.
7. La mesure des progrès réalisés c'est le tableau de bord synthétique de la direction qui répertorie les fonctionnalités prévues mises dans le produit.
8. Les développeurs travaillent tard le soir , les week-ends et annulent leurs congés pour tenir les deadlines .

# Principes sous-jacents du Manifeste 9-11

9. Les développeurs tiennent leurs engagements en mettant le produit en production le jour dit, puis en fournissant rapidement les correctifs de bugs aux clients.

# Principes sous-jacents du Manifeste 9-11

9. Les développeurs tiennent leurs engagements en mettant le produit en production le jour dit, puis en fournissant rapidement les correctifs de bugs aux clients.
10. Profiter des économies d'échelle en s'assurant que les ressources mettent autant de fonctionnalités que possible dans chaque version. Même si personne ne les utilise c'est toujours bien sur la fiche technique .

# Principes sous-jacents du Manifeste 9-11

9. Les développeurs tiennent leurs engagements en mettant le produit en production le jour dit, puis en fournissant rapidement les correctifs de bugs aux clients.
10. Profiter des économies d'échelle en s'assurant que les ressources mettent autant de fonctionnalités que possible dans chaque version. Même si personne ne les utilise c'est toujours bien sur la fiche technique .
11. Mieux vaut embaucher à prix d'or des Architectes Logiciels brillants pour concevoir le produit, des développeurs débutants moins coûteux pourront se charger de la réalisation sous leur direction.

# Principes sous-jacents du Manifeste 12-13

11. A la fin du projet - si le temps le permet - on peut envisager des analyses postmortem avec les ressources, bien entendu ça ne doit pas affecter la capacité des ressources à accomplir un vrai travail.

# Principes sous-jacents du Manifeste 12-13

11. A la fin du projet - si le temps le permet - on peut envisager des analyses postmortem avec les ressources, bien entendu ça ne doit pas affecter la capacité des ressources à accomplir un vrai travail.
12. Apporter une certaine fraicheur aux ressources en utilisant à leur intention le vocabulaire à la mode cette décennie, que ce soit pour décrire les processus, mettre à jour les manuels de procédure, ou bien dans les formations ou les communications de la direction, ceci sans changer en rien notre manière de penser ou nos comportement.

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile !
Ce ne sont malheureusement pas des caricatures.

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile ! Ce ne sont malheureusement pas des caricatures.

- Les équipes de développement Agile accumulent trois fois plus de travail que possible à chaque itération

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile ! Ce ne sont malheureusement pas des caricatures.

- Les équipes de développement Agile accumulent trois fois plus de travail que possible à chaque itération
- le but de l'Agilité c'est d'augmenter la productivité et de développer plus vite

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile ! Ce ne sont malheureusement pas des caricatures.

- Les équipes de développement Agile accumulent trois fois plus de travail que possible à chaque itération
- le but de l'Agilité c'est d'augmenter la productivité et de développer plus vite
- utiliser des méthodes Agile permet de sous-traiter le développement logiciel offshore

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile ! Ce ne sont malheureusement pas des caricatures.

- Les équipes de développement Agile accumulent trois fois plus de travail que possible à chaque itération
- le but de l'Agilité c'est d'augmenter la productivité et de développer plus vite
- utiliser des méthodes Agile permet de sous-traiter le développement logiciel offshore
- la méthode Agile consiste à lister les tâches à faire sur des post-it et les coller sur un tableau blanc

# Sabotagile !

Les exemples suivants sont de tels contresens qu'ils relèvent du Sabotagile ! Ce ne sont malheureusement pas des caricatures.

- Les équipes de développement Agile accumulent trois fois plus de travail que possible à chaque itération
- le but de l'Agilité c'est d'augmenter la productivité et de développer plus vite
- utiliser des méthodes Agile permet de sous-traiter le développement logiciel offshore
- la méthode Agile consiste à lister les tâches à faire sur des post-it et les coller sur un tableau blanc
- Pour une équipe Agile, en cas de retard il suffit de jouer sur la qualité et de tester moins pour gagner du temps

# Sabotagile !

Encore quelques contre exemples, toujours basés sur des cas réels...

# Sabotagile !

Encore quelques contre exemples, toujours basés sur des cas réels...

- Le Coach Agile découpe les demandes du Product Owner en tâches et les distribue à l'équipe

# Sabotagile !

Encore quelques contre exemples, toujours basés sur des cas réels...

- Le Coach Agile découpe les demandes du Product Owner en tâches et les distribue à l'équipe
- Si une tâche est complexe il est normale qu'elle dure plusieurs itérations

# Sabotagile !

Encore quelques contre exemples, toujours basés sur des cas réels...

- Le Coach Agile découpe les demandes du Product Owner en tâches et les distribue à l'équipe
- Si une tâche est complexe il est normale qu'elle dure plusieurs itérations
- Dans une équipe Agile, il est préférable d'avoir une équipe dédiée par composant et technologie : ex : interace utilisateur, back-end

# Sabotagile !

Encore quelques contre exemples, toujours basés sur des cas réels...

- Le Coach Agile découpe les demandes du Product Owner en tâches et les distribue à l'équipe
- Si une tâche est complexe il est normale qu'elle dure plusieurs itérations
- Dans une équipe Agile, il est préférable d'avoir une équipe dédiée par composant et technologie : ex : interace utilisateur, back-end
- Le diagramme de Gantt est un outil privilégié des équipes Agiles

# Sabotagile !

- Les tests sont de la seule responsabilité de l'équipe QA, les developpeurs ne testent rien (encore moins des tests automatisés)

# Sabotagile !

- Les tests sont de la seule responsabilité de l'équipe QA, les developpeurs ne testent rien (encore moins des tests automatisés)
- Ce qui compte ce sont les heures de présence, les meilleurs employés arrivent tôt et repartent tard.

# Sabotagile !

- Les tests sont de la seule responsabilité de l'équipe QA, les developpeurs ne testent rien (encore moins des tests automatisés)
- Ce qui compte ce sont les heures de présence, les meilleurs employés arrivent tôt et repartent tard.
- la meilleure motivation pour une équipe Agile ce sont les primes individuelles au résultat

# L'Agilité ça marche

## Le chaos

Standish Group

2015 Chaos Report

Comparaison
du résultat des projets en TI

| Taille | Méthode | Succès | Challenge | Echec |
|---|---|---|---|---|
| Toute taille de projets | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| Grands projets | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Projets intermédiaires | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Petits projets | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

# L'Agilité ça marche

## Critères de succès

Standish Group

2015 Chaos Report

Facteurs qui contribuent ENSEMBLE au succès des projets

| Critère | Points |
|---|---|
| Soutien de la direction | 15 |
| Maturité émotionnelle | 15 |
| Implication des utilisateurs | 15 |
| Optimisation de l'entreprise | 15 |
| Ressources compétentes | 10 |
| Normes d'architecture (SAME) | 8 |
| Processus Agile | 7 |
| Exécution modeste | 6 |
| Expertise en gestion de projets | 5 |
| Objectifs d'affaires clairs | 4 |

70

30