Qualité et Stratégie LT

A propos de moi

- Kevin Glass, PhD
- 15 ans d'expérience en développement informatique
- CTO d'une entreprise SaaS

Pour moi, la qualité est …

Racontez moi votre avis…

Avez-vous déjà livré un produit ? Vos clients étaient contents ? Il s'est passé quoi ?

“Je me souviens quand j’ai fait un projet de jeux-vidéo pour une musée…”

Je vous raconte mon expérience en tant que Lead Développeur pour un parcours de jeux-vidéo dans un musée en France

Perspective de qualité

Dev

Facile à maintenir

Bien documenté

Structure logique

… autres ?

Logiciel

Fonctionne comme attendu

Disponibilité

Support technique

Apparence ? UI/UX ?

… autres ?

Utilisateur

Stratégie de qualité longue terme

- Une stratégie se compose en regardant la qualité de plusieurs perspectives
- Touche à chaque étape dans la chaine de production et maintenance
- « Quality engineering »

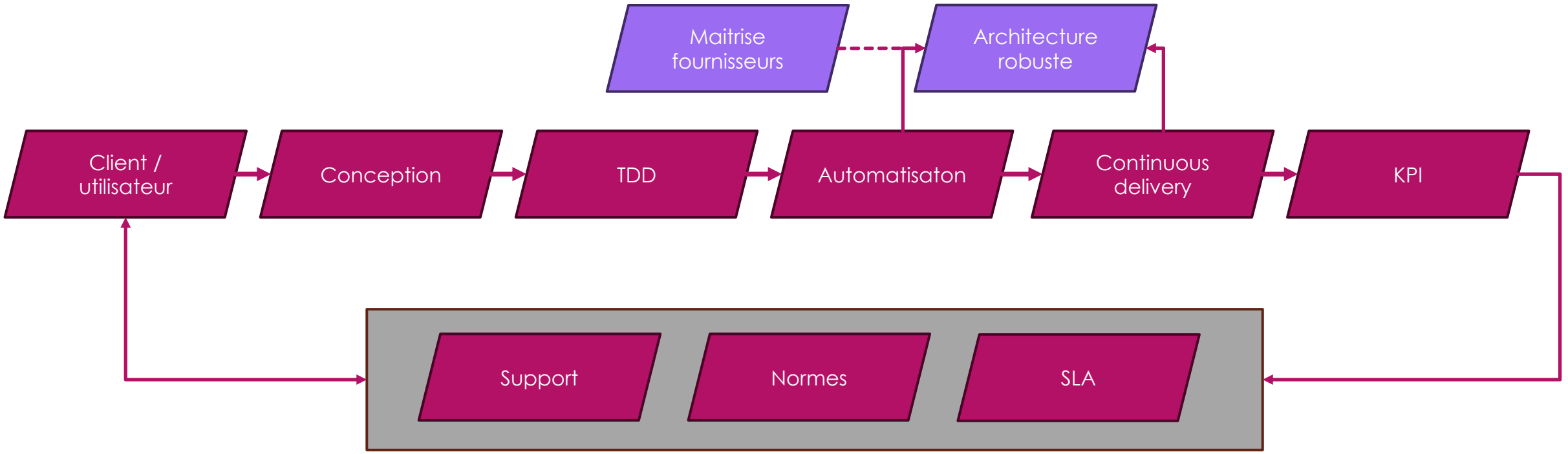
…aussi bon que le maillon le plus faible

Quality Engineering

- **Détection précoce des défauts** La détection précoce des défauts permet d'éviter que les problèmes ne s'aggravent, ce qui réduit les coûts et les efforts nécessaires pour résoudre les problèmes à un stade ultérieur du processus de développement.
- **Tests continus et automatisation :** Elle encourage les tests continus tout au long du cycle de développement des logiciels (SDLC). L'automatisation des processus de test garantit un retour d'information plus rapide et plus fiable, ce qui permet aux équipes d'identifier et de résoudre rapidement les problèmes.
- **Amélioration des performances des logiciels :** Les tests rigoureux effectués par les ingénieurs qualité permettent d'identifier et de rectifier les problèmes liés aux performances, garantissant ainsi que le logiciel fonctionne de manière optimale dans différentes conditions.
- **Amélioration de l'expérience utilisateur** : L'ingénierie de la qualité s'attache à tester non seulement les fonctionnalités, mais aussi la facilité d'utilisation. En garantissant une expérience utilisateur positive, les directeurs techniques peuvent améliorer la satisfaction et la fidélité des clients.
- **Atténuation des risques** : L'ingénierie de la qualité permet d'identifier et d'atténuer les risques potentiels liés à la mise en œuvre des technologies. Cette approche proactive minimise la probabilité de défaillances du système, de violations de la sécurité et d'autres problèmes critiques.
- **Respect des normes et de la conformité :** L'ingénierie de la qualité garantit que les solutions technologiques respectent les normes industrielles et les exigences de conformité dans les secteurs réglementés où la non-conformité peut avoir des répercussions juridiques et financières.
- **Mise sur le marché plus rapide :** Grâce à l'automatisation des tests et aux tests continus, l'ingénierie de la qualité accélère le processus de développement, ce qui permet de fournir plus rapidement des produits et des fonctionnalités de haute qualité.
- **Réduction des coûts** : En détectant et en corrigeant les problèmes dès le début du processus de développement, l'ingénierie de la qualité contribue à réduire le coût global du développement et de la maintenance.
- **Prise de décision fondée sur des données :** L'ingénierie de la qualité s'appuie sur des mesures et des analyses de données pour évaluer la qualité des logiciels. Les directeurs techniques peuvent utiliser ces informations pour prendre des décisions éclairées, allouer des ressources et améliorer les processus.
- **Impact culturel** : L'ingénierie de la qualité contribue à promouvoir une culture de la qualité au sein de l'organisation, en soulignant l'importance de fournir des solutions technologiques fiables et performantes.

L'approche de ce cours

- Une série de politiques axées sur la qualité



Le client

Gestion d'attentes ; produit de qualité

Le client

- Le client est la cible principale de notre stratégie de qualité.
- En fin de compte, il est la source de nos revenus, un client heureux = un chèque de paie à la fin du mois.
- En tant que directeur technique, vous pouvez être confronté à différents types de clients :

  - **direct** : un client qui a commandé un produit et qui paie directement
  - **indirect** : un client qui ne paie pas directement, mais dont l'approbation est obligatoire pour le paiement.
    - **grand public** : vous avez l'intention de vendre au public, et la notoriété et la confiance sont donc cruciales.
    - **financement indirect** : le financement dépend de votre clientèle, par exemple lorsque les annonceurs financent une plateforme gratuite pour les utilisateurs.

Il est essentiel pour votre stratégie de **comprendre votre client** et ce qui motive son estimation de la qualité.

Cas d'exemple

- Un client direct qui commande un logiciel

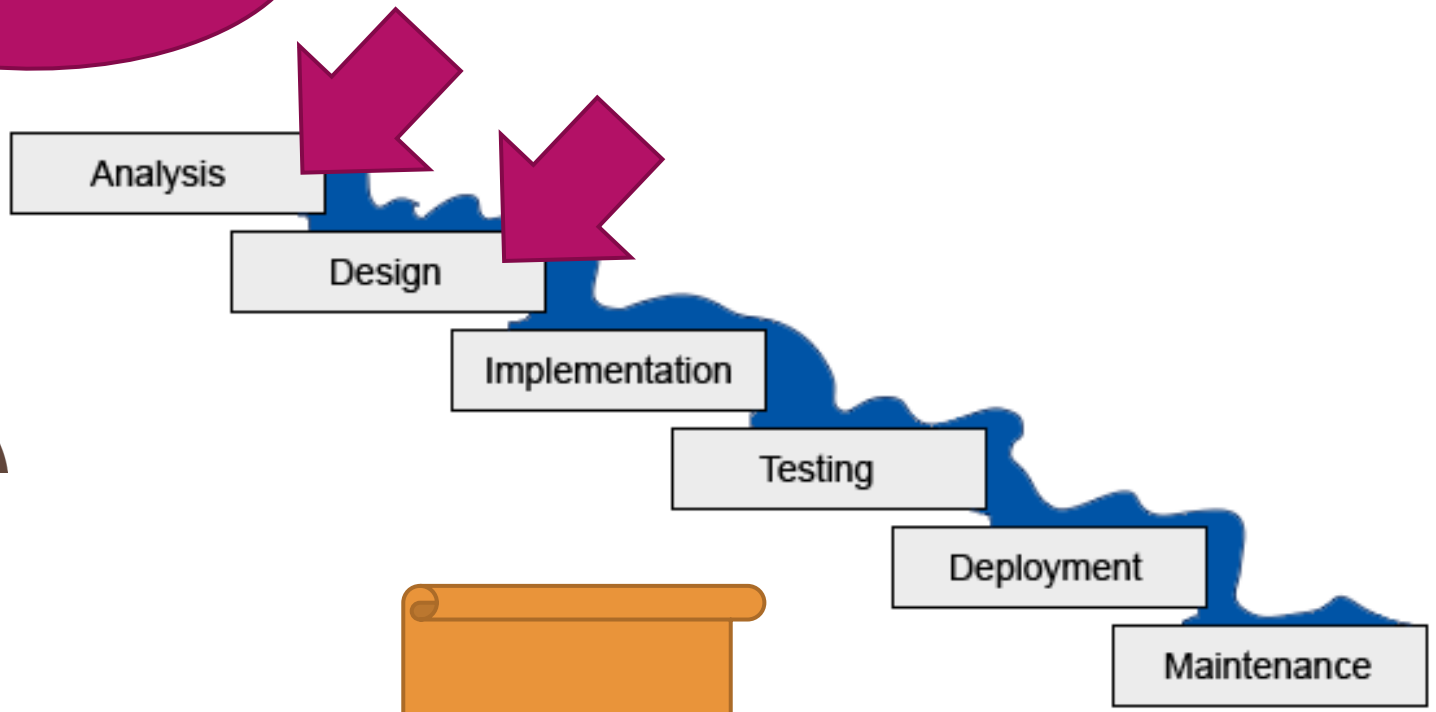
La méthode notoire « Waterfall »

Le projet **VR** est en route, j'assure mes investisseurs

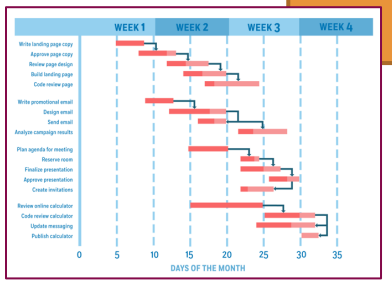
CEO

Project Manager

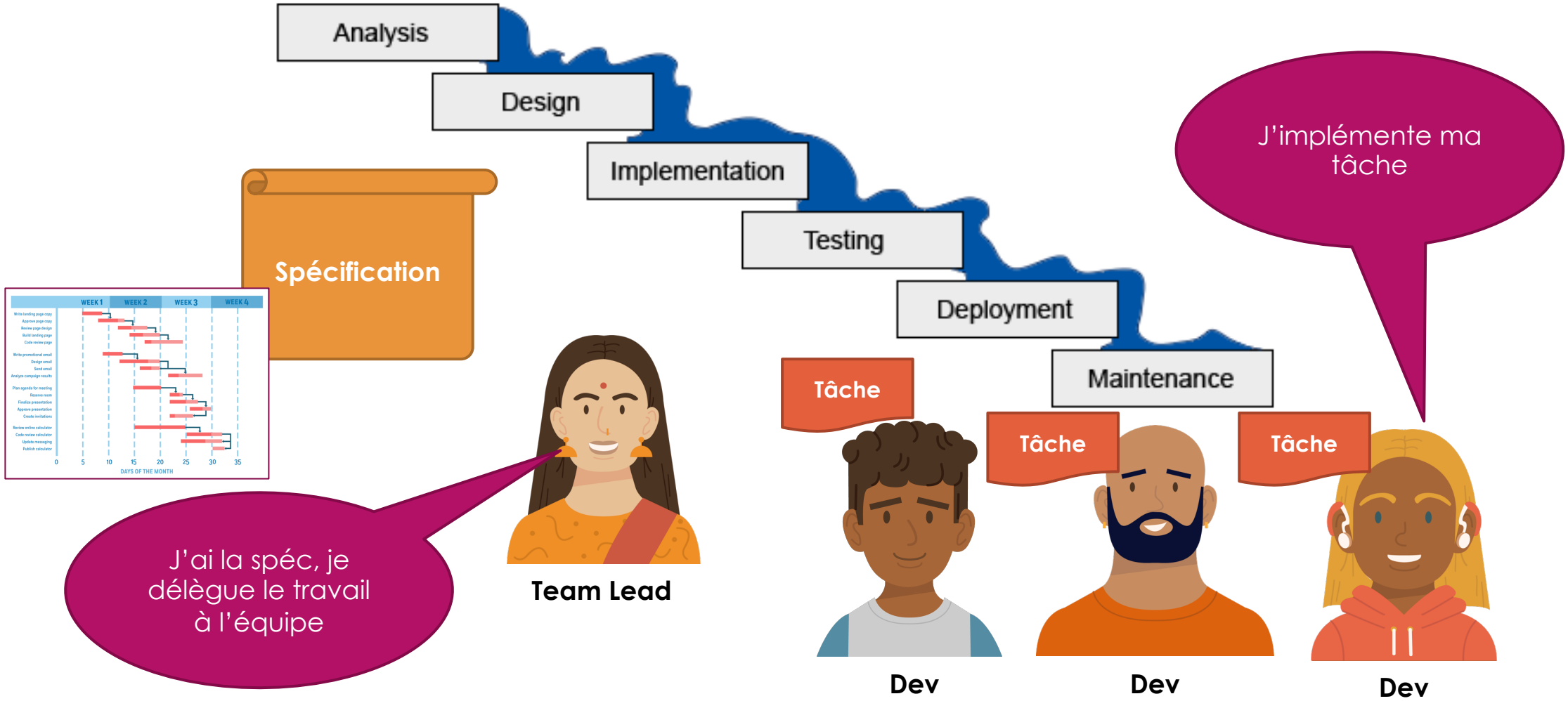
J'ai parlé aux clients, et j'ai crée la spécification et le plan qui dur 2 ans



Spécification



La méthode notoire « Waterfall »



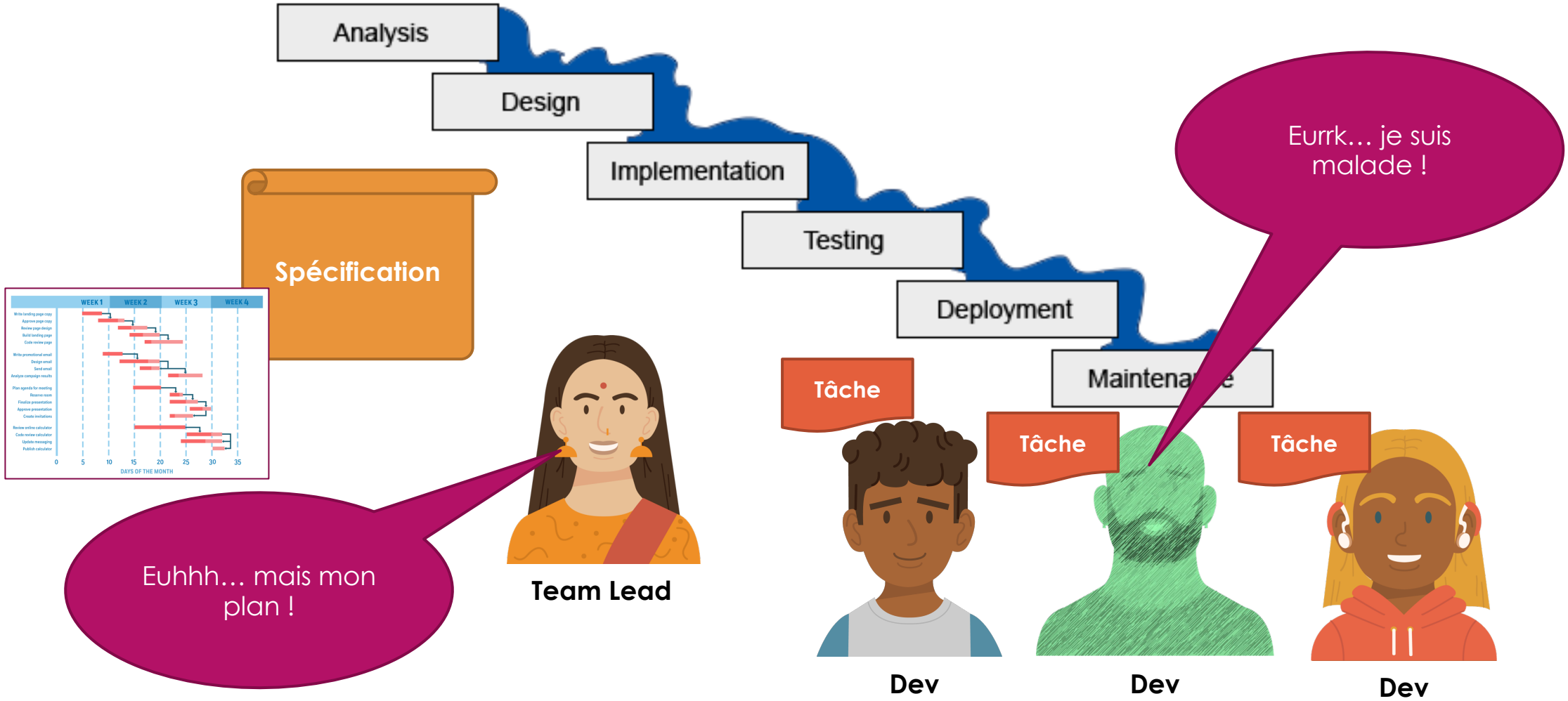
La méthode notoire « Waterfall »

- Le gérant du projet doit tout savoir en avance
  - Toutes les dépendances entre les tâches
  - Le temps nécessaire pour faire une tâche

C'est quoi une dépendance ?

Qui estime le temps ? Et comment ?

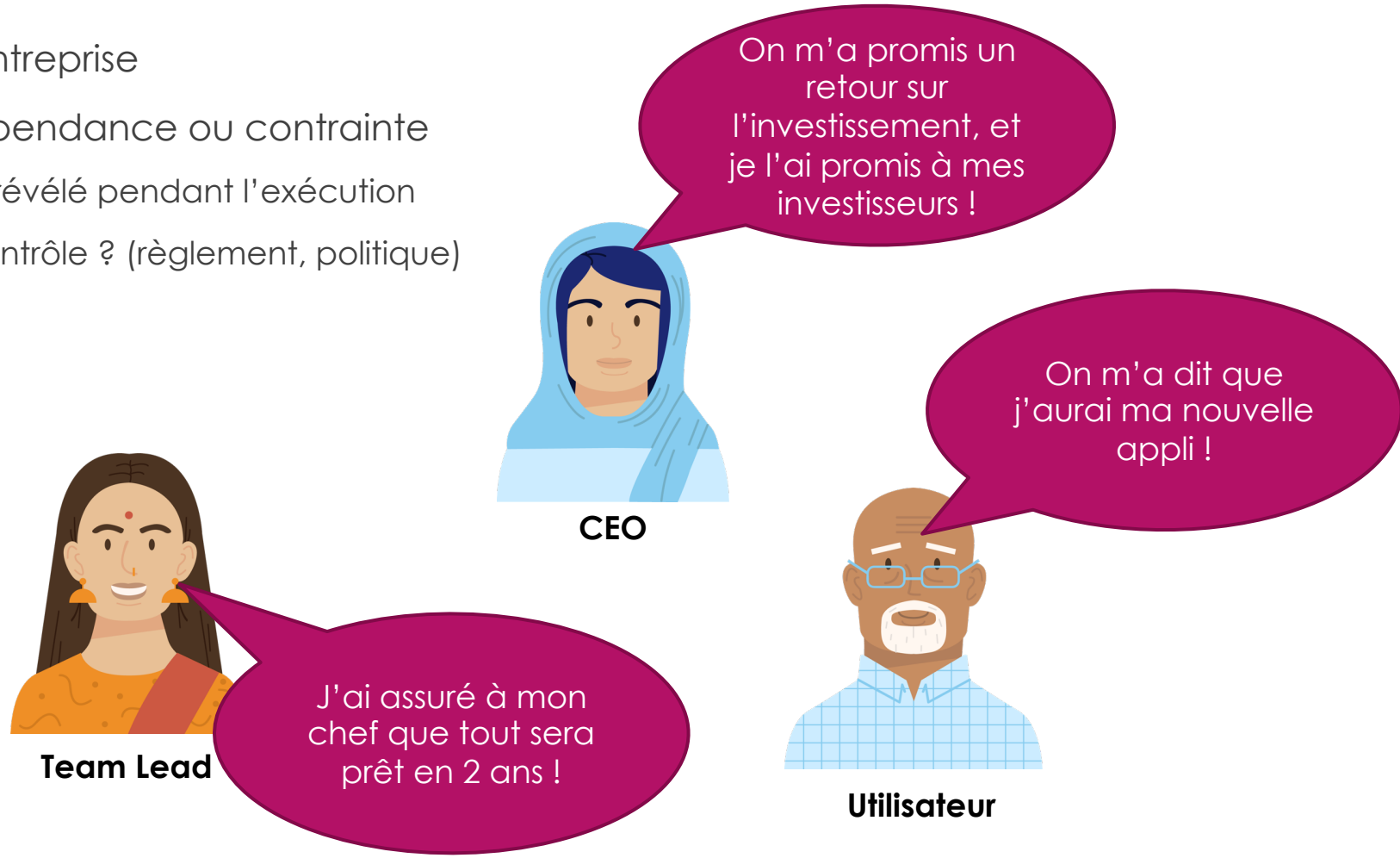
La méthode notoire « Waterfall »



Des imprévus

- Pendant un projet, il y a **toujours** des imprévus
  - On tombe malade
  - Des coéquipiers partent de l'entreprise
  - On découvre une nouvelle dépendance ou contrainte
    - Un détail caché et seulement révélé pendant l'exécution
    - Une modification hors notre contrôle ? (règlement, politique)

- Comment adapter ?

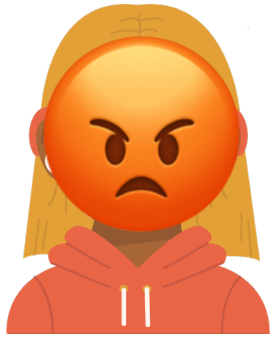


Désolé les gars, on va travailler les soirs et weekends

Team Lead

Dev

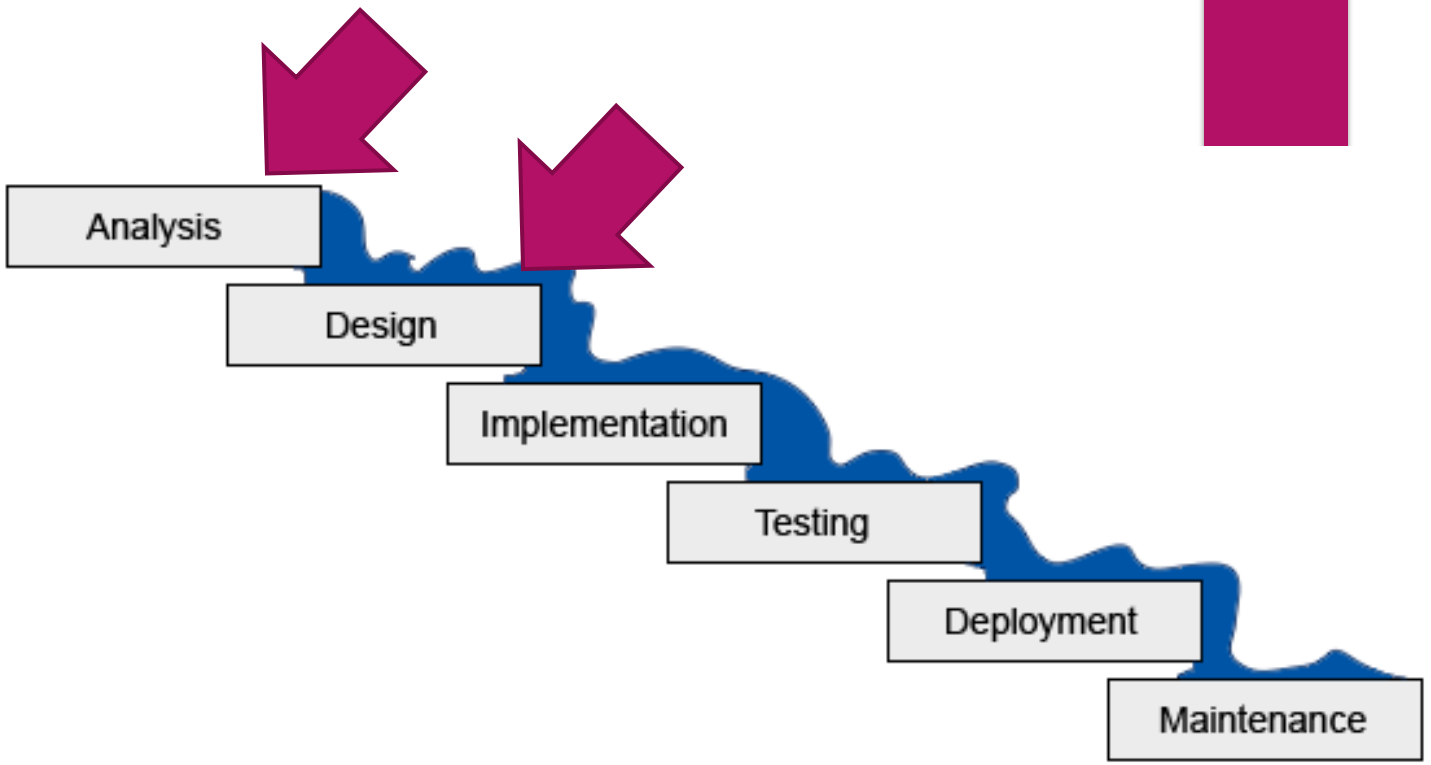
Dev



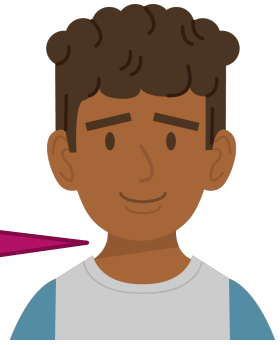
Dev

On fait marche arrière et on adapte la spec et le plan

Team Lead

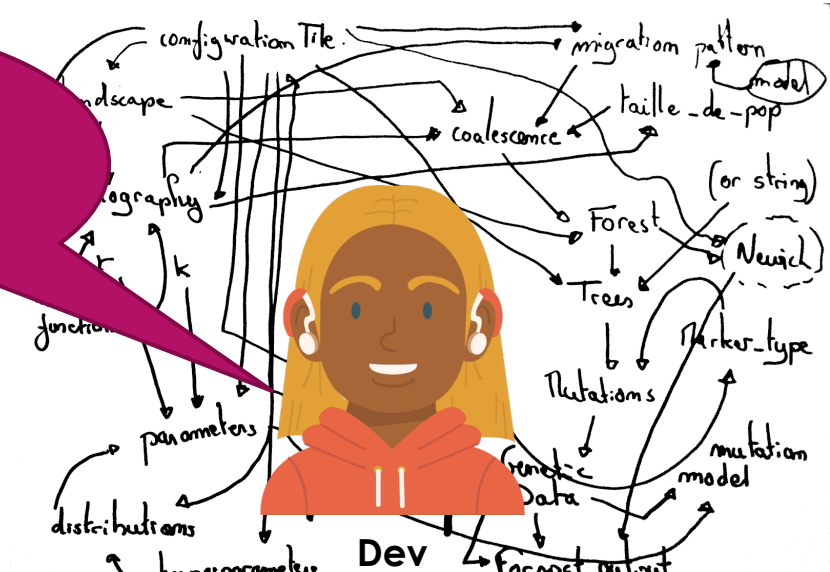


Bon, à la poubelle tout mon travail…. C'est la vie

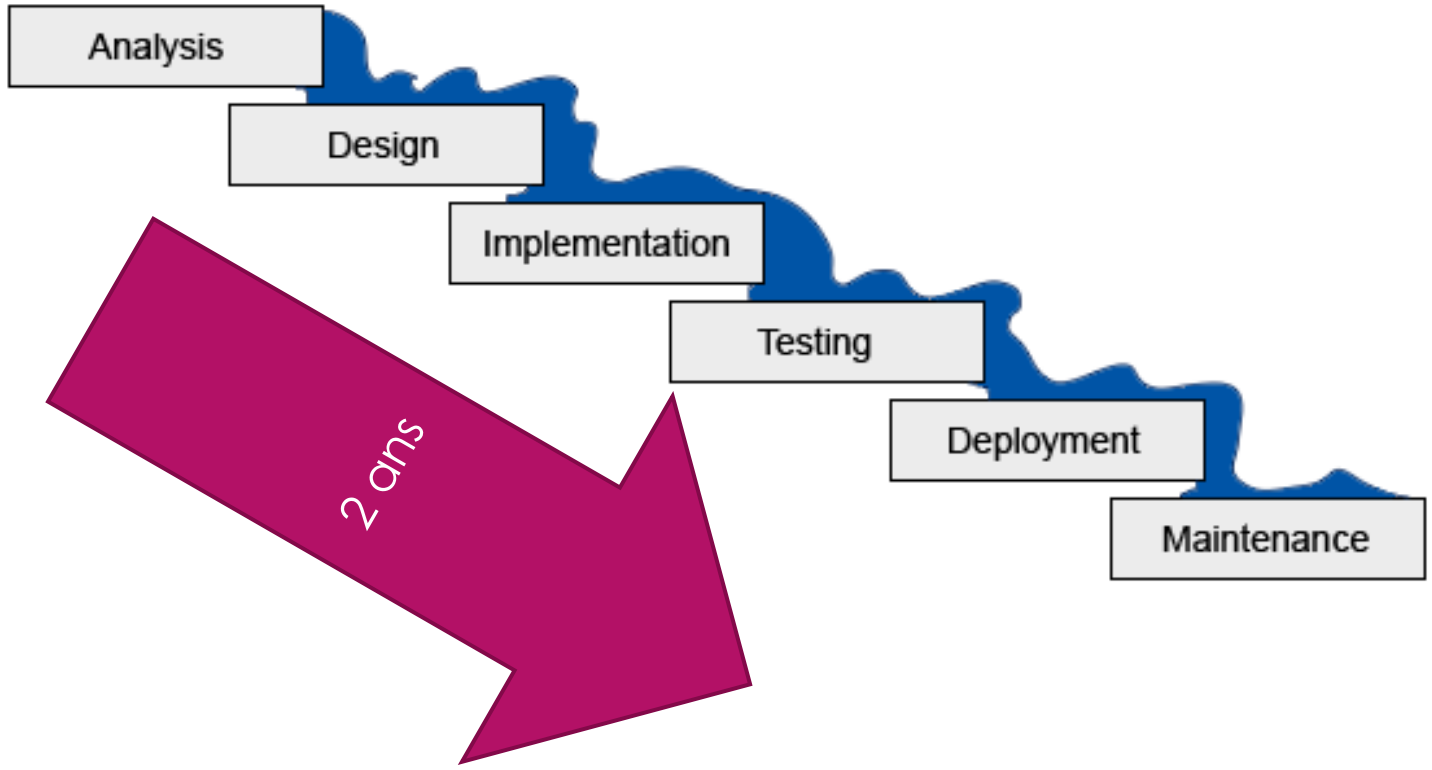


Dev

On n'a pas le temps de tout refaire, je vais adapter mon code existant



La méthode notoire « Waterfall »



C’est fini !!!!

Logiciel

Dev

Team Lead

Dev

Dev

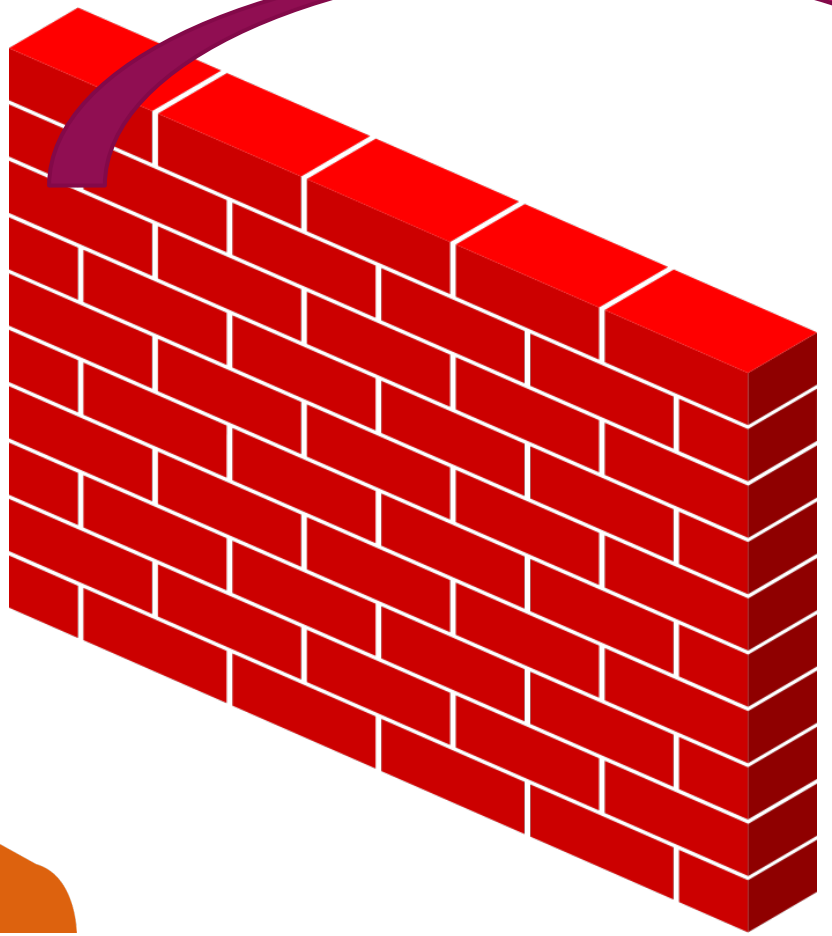
Utilisateur

Team Lead

Dev

Dev

Dev



Logiciel

Utilisateur

Mais … ce n'est pas ce que je voulais !

La méthode notoire « Waterfall »

- Pendant le 2 ans (ou plus) d'exécution
  - Les besoins ont changés
  - Le monde a changé

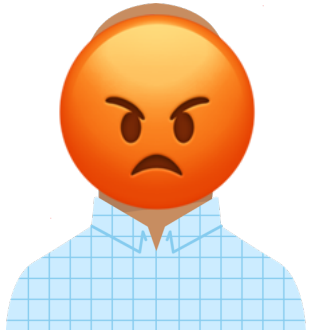
Comment cela peut se faire ?

Est-ce qu'on est certain des besoins initiaux ?

Comment le monde peut changer ? Pensez à l'exemple de la **VR !**

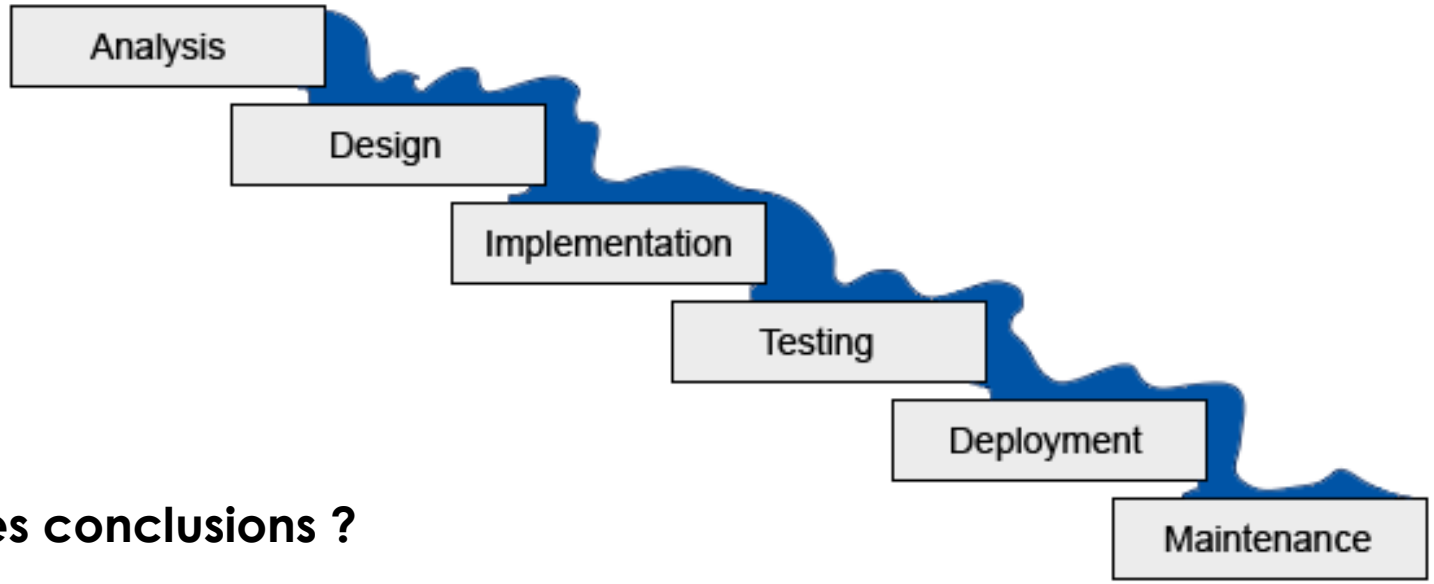
CEO

- Logiciel instable
  - Bugs crées à cause des modifications
  - Code « spaghetti » difficile à maintenir



Utilisateur

La méthode notoire « Waterfall »



Avez-vous des conclusions ?

Des solutions ?

“J’ai entendu parler d’agile…”

Racontez moi votre idée d’agile !

La promesse de l'Agile

Projets livrés à temps

Meilleure qualité

Performance et productivité époustouflante !

Utilisateurs satisfaits

Equipe heureux et qui travail des horaires « normaux »

« Agile Manifesto »

- AGILE est avant tout un **système de valeurs**

Nous découvrons de meilleures façons de développer des solutions,
par notre propre pratique et en aidant les autres dans leur pratique.

Grâce à ce travail, nous en sommes venus à valoriser :

Les individus et leurs interactions, de préférence aux processus et aux outils,
Des solutions opérationnelles, de préférence à une documentation exhaustive,
La collaboration avec les clients, de préférence aux négociations contractuelles,
La réponse au changement, de préférence au respect d'un plan.

Précisément, même si les éléments à droite ont de la valeur,
nous reconnaissons davantage de valeur dans les éléments à gauche.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler,
James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick,
Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

« Agile Manifesto »

- AGILE est avant tout un **<u>système de valeurs</u>**

Nous découvrons de meilleures façon
par notre propre pratique et en aid

Grâce à ce travail, nous en

Les individus et leurs interactions, de préfé tils,
Des solutions opérationnelles, de préférence à une documentation exhaustive,
La collaboration avec les clients, de préférence aux négociations contractuelles,
La réponse au changement, de préférence au respect d'un plan.

Précisément, même si les éléments à droite ont de la valeur,
<u>nous reconnaissons davantage de valeur dans les éléments à gauche</u>.

On aurait pu éviter des problèmes de nos exemple si il y a eu des meilleurs communications entre l'équipe et le client :
plus **<u>souvent</u>**, plus **<u>efficace</u>**
et **<u>en personne</u>**

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

« Agile Manifesto »

- AGILE est avant tout un **système de valeurs**

Nous découvrons de meilleures façons de développer des solutions,
par notre propre pratique et en aidant les autres dans leur pratique.

Grâce à ce travail, nous en sommes venus à valoriser :

Les individus et leurs interactions, de préférence aux processus et aux outils,
Des solutions opérationnelles, de préférence à un
La collaboration avec les clients, de p
La réponse au changement, de pré

Précisément, même si les éléments à
nous reconnaissons davantage de valeur dans l

Si on avait montré au fur et à mesure notre team lead et ou client, on aurait pu détecter plus tôt des problèmes.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler,
James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick,
Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

« Agile Manifesto »

- AGILE est avant tout un **<u>système de valeurs</u>**

Nous découvrons de meilleures façons de développer des s
par notre propre pratique et en aidant les autres dans

Grâce à ce travail, nous en somme

Les individus et leurs interactions, de préfér
Des solutions opérationnelles, de préf
La collaboration avec les clients, de préférence aux négoci
La réponse au changement, de préférence au respect d'un plan.

Collaboration régulière, des retours clients, l'agilité de changer le projet avec des besoins.

Adhérer à un contrat ou imposer de la rigidité assure des mauvais résultats

Précisément, même si les éléments à droite ont de la valeur,
<u>nous reconnaissons davantage de valeur dans les éléments à gauche</u>.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler,
James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick,
Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

« Agile Manifesto »

- AGILE est avant tout un **système de valeurs**

Nous découvrons de meilleures façons de développer des solutions,
par notre propre pratique et en aidant les autres dans leur pratique.

Grâce à ce travail, nous en sommes venus à valoriser :

Les individus et leurs interactions, de préférence aux processus et aux outils,
Des solutions opérationnelles, de préférence à une documentation exhaustive,
La collaboration avec les clients, de préférence aux négociations contractuelles,
La réponse au changement, de préférence au respect d'un plan

Précisément, même si les éléments à d
nous reconnaissons davantage de valeur da

Trop souvent on adhère au « plan » fait par ou approuvé par des autres pour se dégager de notre responsabilité.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunnin
James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian
Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Conception

Construire une plateforme maintenable, compréhensible, extensible

Qualité dans l'architecture

- Concevez vos architectures comme étant :
  - modulaires
  - faciles à tester
  - le moins de dépendances possible
  - SOLID

- Pour ce faire, il existe un certain nombre de tendances et de normes industrielles qui ont été développées :
  - Architectures de Conception
    - Clean
    - Hexagonal
    - …
  - Patrons de Design (**design patterns**)
    - MVC
    - Object Component
    - …

- De plus, il ne faut pas réinventer la roue. Utilisez des modules éprouvés, testés et fiables :
  - Frameworks
  - ORMs
  - …

SOLID

- **S**ingle Responsibility Principle.
- **O**pen-closed principle.
- **L**iskov substitution principle.
- **I**nterface segregation principle.
- **D**ependency inversion principle.

- Pourquoi ?
  - Fréquence et effets des changements
  - Plus facile à comprendre
  - Valider la conception à l'aide d'une question : est-il SOLID ?

TDD

Spécification avant l'implémentation

Test Driven Design

- Une philosophie et une technique de développement
  - Rédiger nos spécifications sous la forme d'une série de tests
  - Ces tests spécifient le résultat concret attendu d'un module
  - Initialement, tous les tests échouent
  - Au fur et à mesure que le développement progresse, de plus en plus de tests réussissent.
  - Exemples : https://docs.glassworks.tech/devops/ci/010-tests-unitaires

- Pourquoi ?
  - Les détails spécifiques d'une mise en œuvre deviennent clairs au fur et à mesure que nous écrivons les tests.
  - Par exemple, les détails et les besoins d'une interface peuvent apparaître clairement lors de l'écriture des tests. Cela permettra également de garantir les principes SOLID de notre module final.

Automatisation des tests - CI

- Intégrez vos tests dans un pipeline de validation automatique sur votre serveur GIT.

- Lorsqu'une demande de fusion est effectuée
  - votre responsable technique appréciera immédiatement la qualité du code.
  - les régressions seront automatiquement évitées

- Exemple d'un pipeline de validation ici : https://docs.glassworks.tech/devops/ci/040-ci

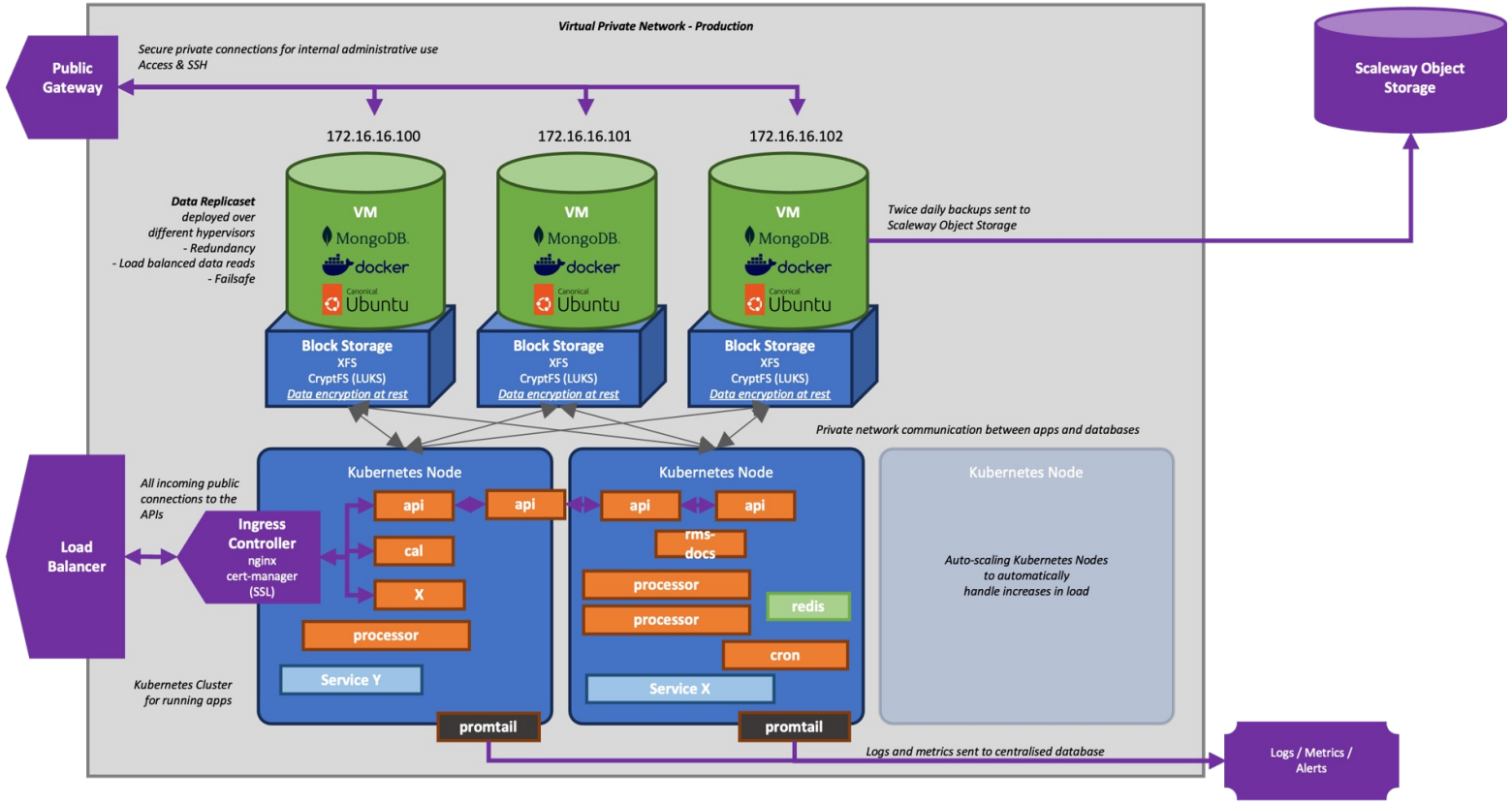
Architectures robustes

Assurer le « uptime » de vos services

Architecture « redondant »

Internal Maintenance Connections

Public Connections



Architecture

- Mission principale : assurer le « uptime » de nos services
  - Réagir automatiquement aux plantages, fautes, disparition des services
  - Gérer automatiquement des pics de charges

- Résilience et rédondance
- Rolling updates
- Sauvegardes « transparentes »
- Orchestration

Déploiement

Déploiement automatisé

Déploiement

- Gestion des environnements
  - Dev
  - Test / Staging
  - Production

- Automatisation du déploiement
  - Kubernetes : https://docs.glassworks.tech/devops/deploiement-avec-k8s/kubernetes
  - Containerisation et mise en production
    - La notion de « étiquettes » (Tags) en GIT
    - Builds automatisés
    - Déploiements automatisés
    - Exemple avec Gitlab et Kubernetes : https://docs.glassworks.tech/devops/cd/010-cd

Risques

Anticiper et limiter les risques

Risques

- Réfléchissez, planifiez et rédigez vos politiques de gestion des risques.
- Dans la mesure du possible, testez vos stratégies de gestion des risques !
- Quels types de risques ?
  - Perte de données ?
  - Perte de service ?
  - Les rançons ?
  - Vol de données ?
  - Corruption délibérée de données ?
  - Corruption accidentelle de données ?
  - Perte de secrets ?

Normes

Ses normes industrielles qui rassurent les clients

Normes

- ISO9001
  - Certification qui atteste de l'existence d'un certain nombre de politiques de qualité au sein d'une organisation.
  - La mise en œuvre des politiques en interne et la conduite du processus d'audit et de certification peuvent s'avérer coûteuses.
  - Toutefois, une entreprise peut afficher le badge, ce qui rassure les clients potentiels.

- ISO/IEC 27001
  - Examiner systématiquement les risques liés à la sécurité de l'information de l'organisme, en tenant compte des menaces, des vulnérabilités et des incidences ;
  - concevoir et mettre en œuvre un ensemble cohérent et complet de contrôles de la sécurité de l'information et/ou d'autres formes de traitement des risques (comme l'évitement ou le transfert des risques) pour faire face aux risques jugés inacceptables ; et
  - adopter un processus de gestion global pour s'assurer que les contrôles de sécurité de l'information continuent à répondre aux besoins de l'organisation en matière de sécurité de l'information sur une base continue.

Accords

Accepter contractuellement un niveau de qualité

SLAs et KPIs

- **S**ervice **L**evel **A**greements
  - Uptime
  - Délai de réponse en cas d'incident
  - Calendrier de maintenance
  - Procédures à suivre lors des mises à jour
  - Protocoles de communication
  - …

- **K**ey **P**erformance **I**ndicators
  - Des mesures concrètes qui permettent de vérifier si un accord de niveau de service est respecté
  - « Total Downtime »
  - Nombre de tickets
  - …

Utilisateurs

Assurer la satisfaction du client

Utilisateurs

- Helpdesk
- Communauté
- Documentation
- Système de tickets et suivi des demandes
- ….

Autres ?

Voyez-vous d'autres facteurs susceptibles d'influencer la qualité ?