Test de performance et Agilité : comment faire pour répondre aux nouvelles exigences

Henrik Rexed

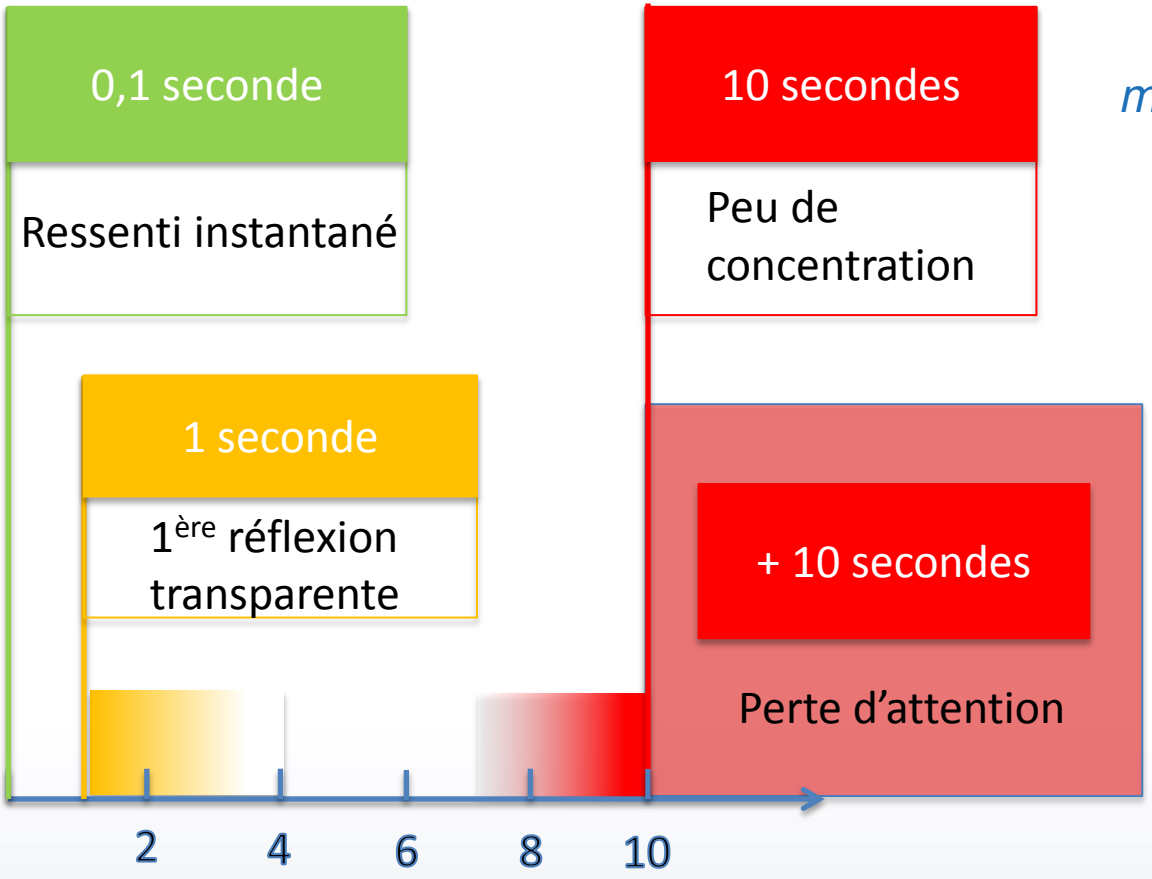
L'importance de la performance

En diminuant les chargements des pages de **2,2s**, Mozilla a augmenté de **15,4%** les téléchargements

Une augmentation de **14%** des donations a été observé en ayant un site **60%** plus performant

Amazon augmente de **1%** son CA lorsque il gagne **100ms** sur le temps de réponse de leur site Web.

Comment notre cerveau perçoit la performance



Une application ayant de mauvaise performance génère

Un stress utilisateur

Applications lentes *demandent*

50% *de plus de*

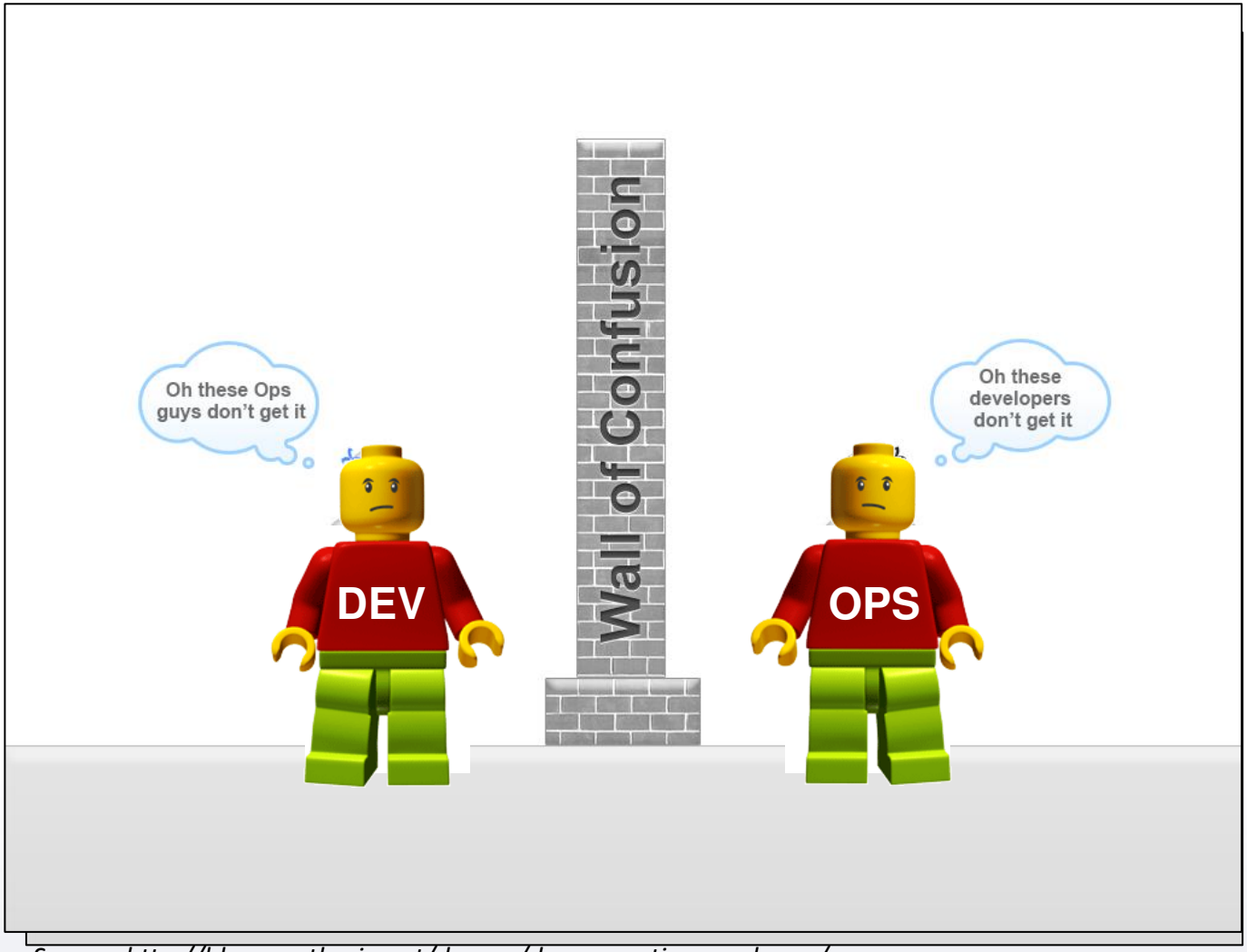
concentration

Agenda

- L'organisation complexifie la tache
- La clé du succès
- Questions

Organisation en silos

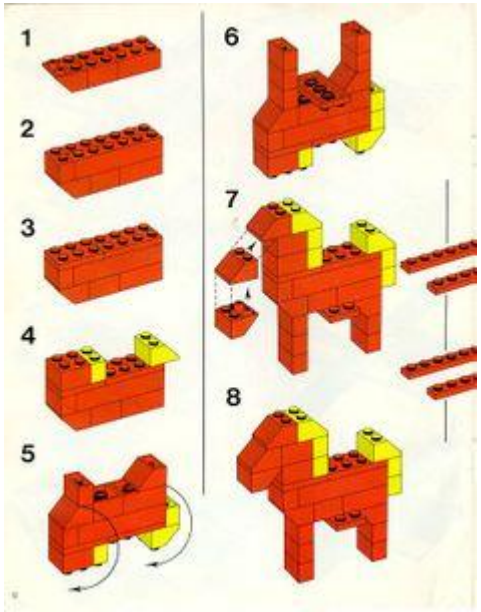
Nous travaillons en Silos



Source: http://blog.anotheria.net/devops/devopsruntime-prologue/

L'Agilité augmente la complexité des déploiements

Le cycle en V vous permettais de jouer simplement



L'Agilité a complexifié votre jeu préféré

Le vendredi soir, vous préférez …

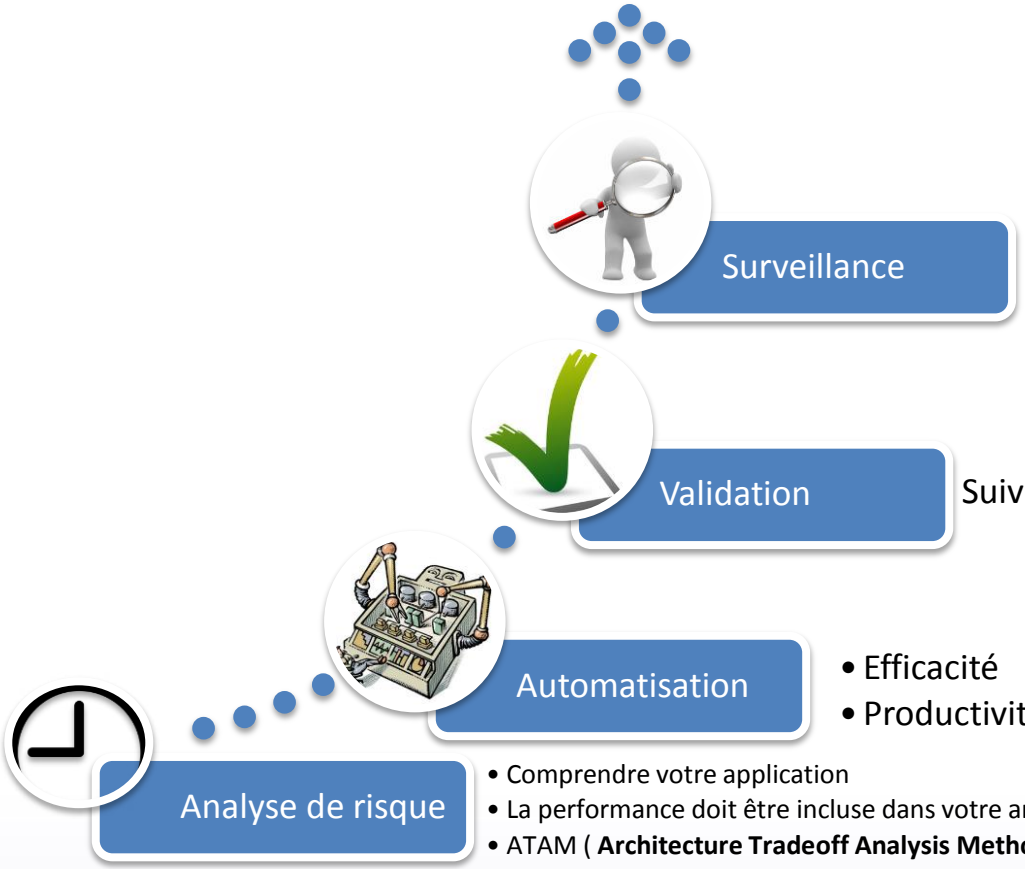
1) Gérer une situation de crise suite au déploiement hebdomadaire ?

2) Boire une bière avec vos amis ?

Agenda

- L’organisation complexifie la tache
- **La clé du succès**
- Questions

La clé du succès



Intégrer la performance au plus tôt

Une implication très tôt est bénéfique

- Poser les questions clés le plus tôt possible
- N'attendez pas que tous se mette en place

La réalité : Nous impliquons trop tardivement la performance

- Les marges de manœuvre sont très limitées

Prioriser les composants à tester

Analyse de risque

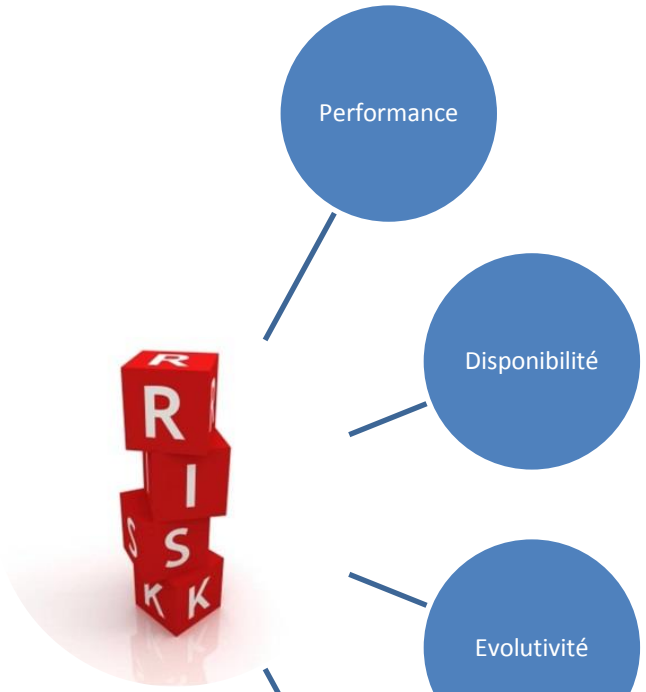
Identifier les processus métier stratégiques

Identifier les composants stratégiques

Limiter vos risque projet lors de chaque phase de build

Limiter vos risques lors de chaque fin d'assemblage

Organiser vos risques de performance



Prendre le temps de comprendre notre application

La charge est générée par :

- Les utilisateurs finaux
- Des systèmes tiers
- Des messages entrants…etc

Se poser les bonnes questions :

- Comment l'application est-elle utilisée ?
- Quelles sont les habitudes des utilisateurs?
- Quand et combien de fois utilisent-ils l'application?
- Allons-nous ouvrir notre service dans plusieurs zones géographique?
- Y-a-t-il une campagne marketing pour promouvoir l'application?
- etc

Les scénarios de tests

Application existante

- Impliquer le référent fonctionnel, RH, chef de projet…etc
- Comprendre le comportement de l'application en s'appuyant sur des logs systèmes.
- Projet de migration: Ne sous-estimer la valeur des données sur les systèmes existant.

Nouvelle application/service

- Impliquer le référent fonctionnel, le chef de projet..etc.
- Essayer de comprendre l'objectif de l'application par rapport au business plan de l'entreprise

Le test de performance unitaire

Chaque partie du système peut être testée

Ce n'est pas une démarche standard

Pourquoi attendre que l'application soit assemblée?

Les cas de tests sont plus simples

La plupart des système sont monolithiques

- Le « Test-Driven Development » peut être une solution

Composants des systèmes tiers

Changer de mentalité

AVANT

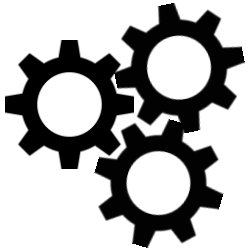
- Le test de performance “record/playback” Tardif
- Critères d’acceptabilité métier
- "Black Box"

AUJOURD’HUI

- Implication au plut tôt de l’ingénieur de performance
- Critères d’acceptabilité pour chaque composant
- "Grey Box”

Le worflow

1) Spécifications



3) Cycle de développement

2) Analyse de risque

Test unitaire

Test du Business Case

Cycle de vie du projet

Quand peut-on automatiser ?

L'automatisation n'est rentable que lorsque vous avez une bonne connaissance de votre application

Si votre application est nouvelle, le coût sera trop important

- Presque pas d'automatisation
- Explorer votre application pour bien la connaitre

Si votre application a été testé à plusieurs reprises

- Il devient pertinent de réfléchir à une stratégie d'automatisation

Les application utilisant des interfaces facilitent l'automatisation

- Les API sont souvent stables et les interfaces ne sont pas impactées par les évolutions

Comment automatiser

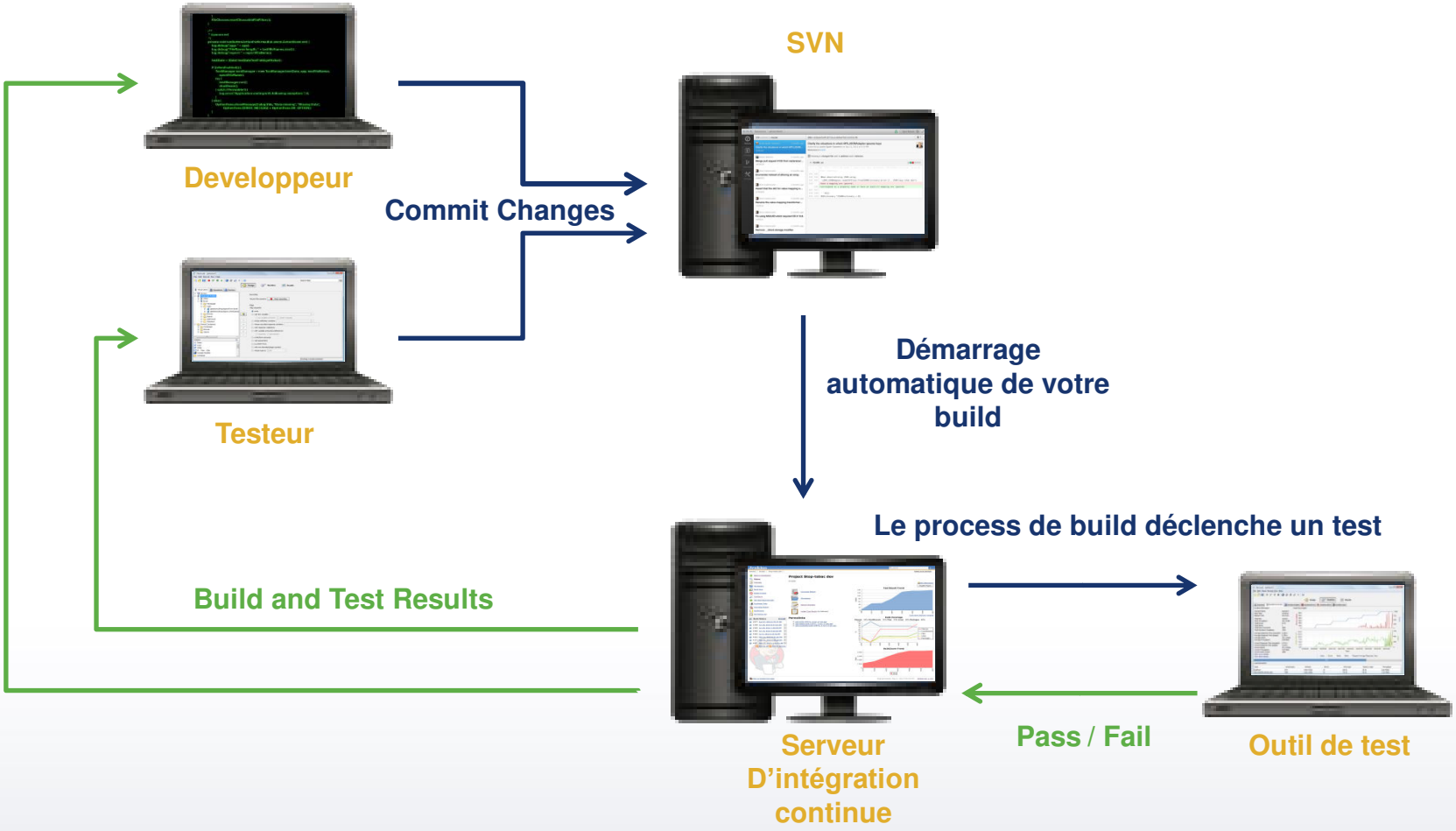
Intégrer les tests de performance dans votre process de build

Définir l'ensemble de vos SLA

Constituer un Tableau de bord sur votre serveur d'intégration continue

Adapter la complexité de vos tests en fonction de la taille de votre build

Architecture de test de perf. automatisée



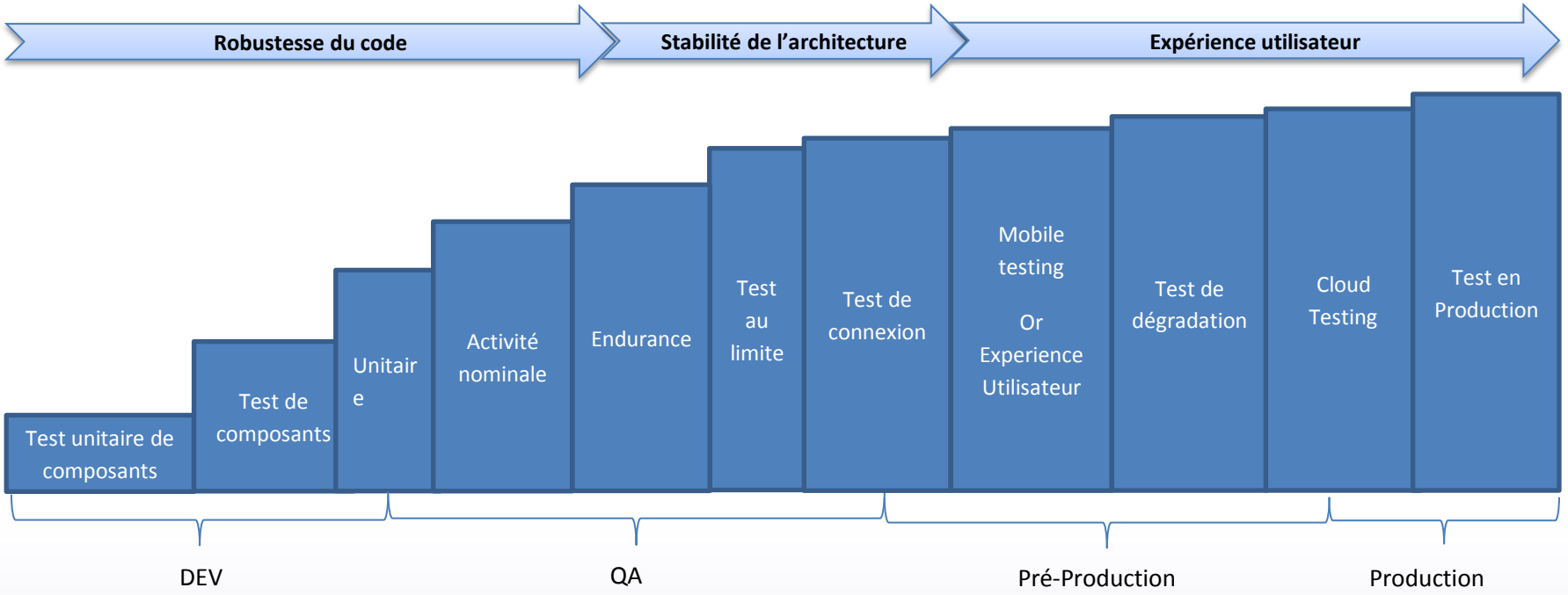
Validation

Créer un tableau de bord orienté performance présentant :

- SLA sur les test composants
- SLA sur les actions métiers

Disposer d'un graphique mettant en avant la régression entre différents builds

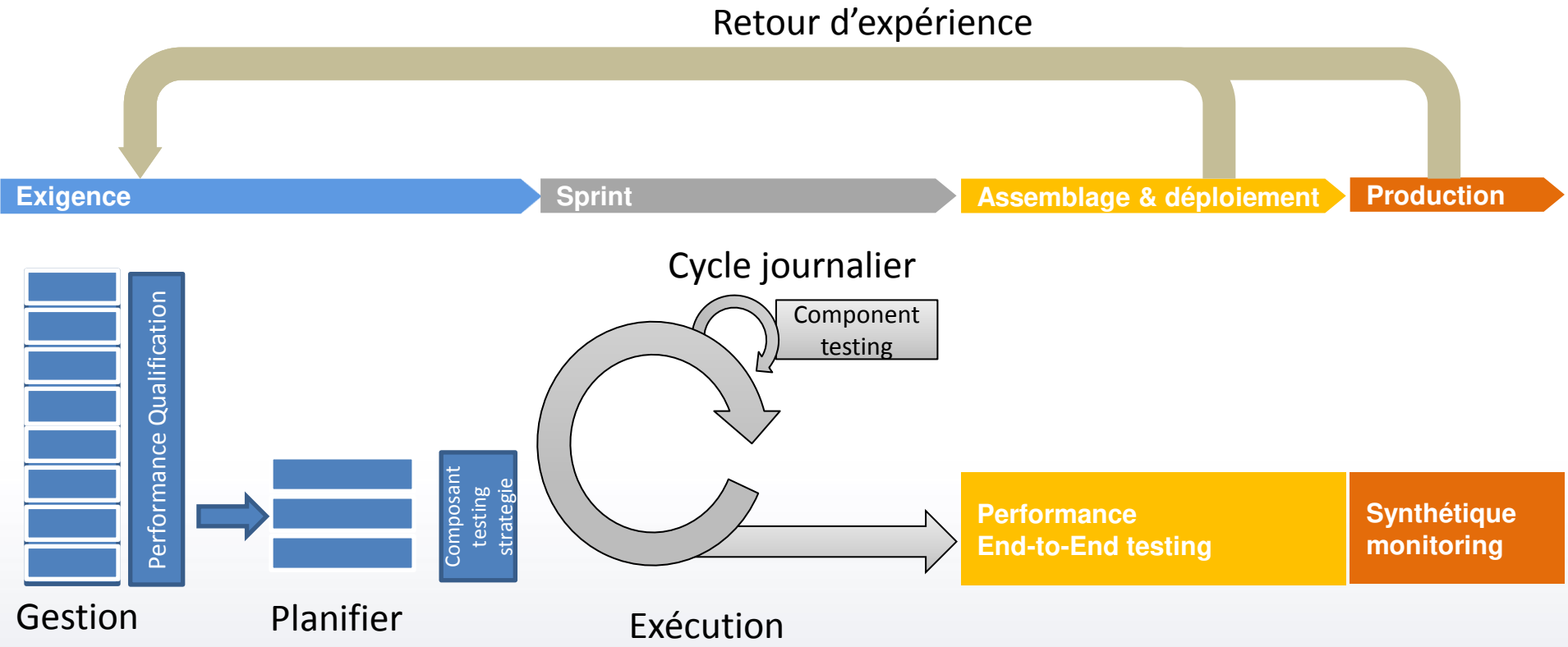
La complexité des tests évolue avec le cycle de vie du projet



Surveillance

Mettre en place un monitoring synthétique

- Evaluer la QoS de votre application au cours du temps
- Utiliser les retours d'expériences de votre production pour orienter vos évolutions



Agenda

- L’organisation complexifie la tache
- La clé du succès
- **Questions**