Comment exploiter les audits de code

Philippe BRIDON (DSI Renault – philippe.bridon@renault.com)

Agenda

- **Le Groupe Renault et la Direction Informatique**
- **L'audit de code dans le contexte Qualité**
- **Gérer les résultats des audits**
- **Autres gains par effets de bord**
- **Conclusion**

Le Groupe Renault et la Direction Informatique

GROUPE RENAULT A FIN 2008

- **Ventes mondiales (VP + VU) :**
  2 382 230 véhicules
- **Chiffre d'affaires :**
  37 791 millions d'euros
- **Résultat net part du Groupe :**
  571 millions d'euros
- **Effectifs :**
  129 068 personnes

- **Une branche automobile à 3 marques :**

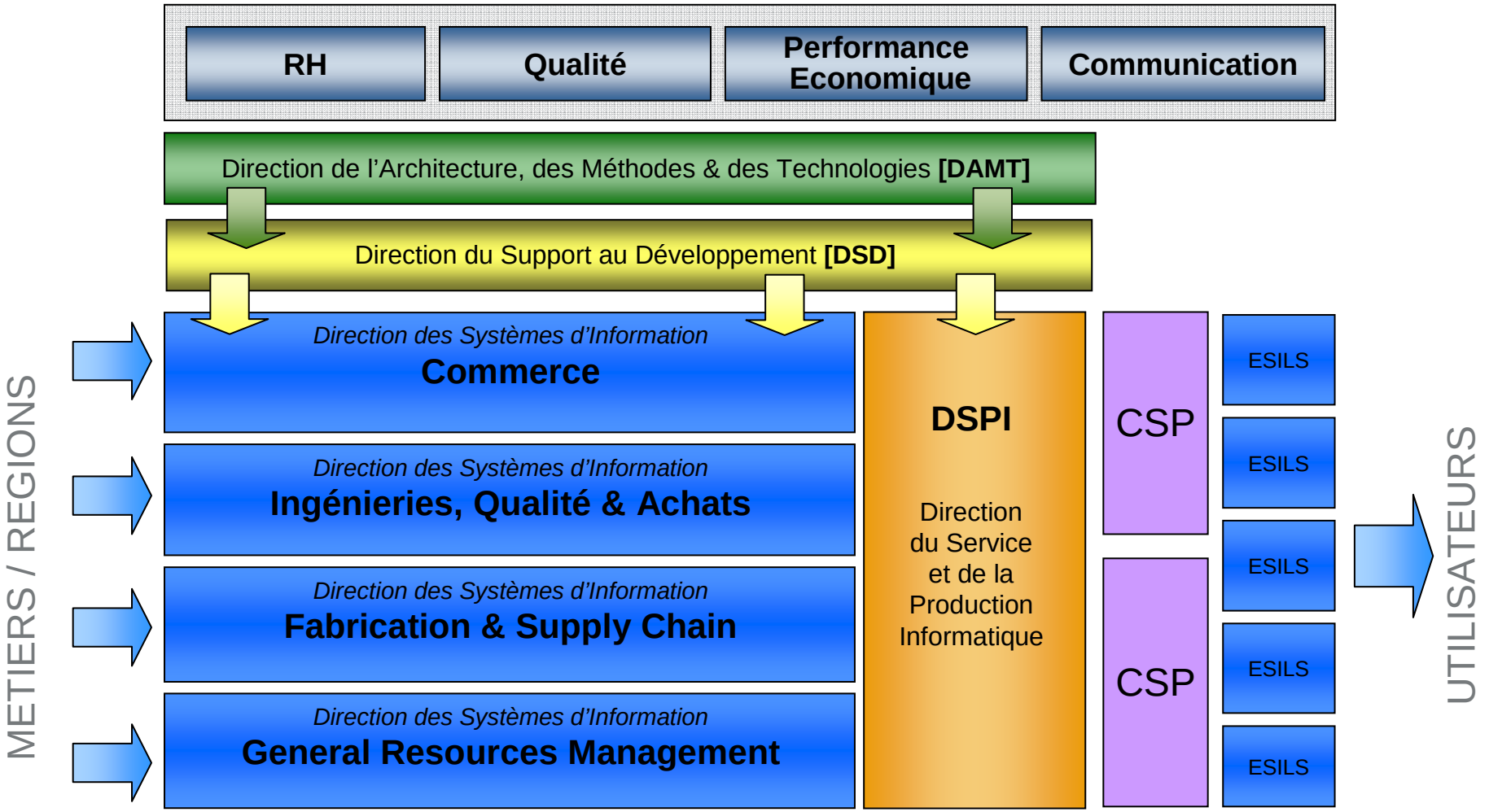
DACIA | RENAULT | RENAULT SAMSUNG MOTORS

- **L'Alliance** RENAULT NISSAN
- **Un partenaire stratégique AvtoVaz (LADA)**

ABTOBA3

- **Renault double champion du monde**
  En 2005 et 2006

FONCTIONNEMENT DE LA DSI-Renault



→ Transversalité : vitesse, engagement et qualité du service

L'INFORMATIQUE RENAULT CERTIFIÉE

CMMI 3

AVRIL 2008 : LA DIRECTION INFORMATIQUE de RENAULT SAS est certifiée CMMI (1) NIVEAU 3 par le SEI (2) pour le DÉVELOPPEMENT et la GESTION de son PARC APPLICATIF

→ UN FONCTIONNEMENT STANDARDISÉ UTILISANT LES MEILLEURES PRATIQUES MONDIALES DE DÉVELOPPEMENT INFORMATIQUE

→ DES PROCESSUS OPTIMISÉS POUR UNE MEILLEURE PERFORMANCE QCD AU SERVICE DES MÉTIERS DU GROUPE

(1) CMMI : Capability Maturity Model Integrated | (2) SEI : Software Engineering Institute, Organisme International de Certification

Audits de code 2009 (JAVA uniquement disponible)

- **184 applications (SI) couvertes (+ 15 en pilote)**
  - Dont 100% des 42 SI de la cible (mises en production majeures)
  - Dont 56% ont un audit de référence
- **610 utilisateurs déclarés**
  - France, Espagne, Roumanie, Tchéquie, Inde ;
  - Renault, Renault Offshore Inde , Dacia, Atos Origin, Meconsa, Satiam…
  - Dont 275 prestataires (45%), la plupart Atos Origin
  - >120 utilisateurs distincts par mois
  - 30-50 sessions par jour ; pic de ~20 utilisateurs simultanés

- **> 1023 audits en 2009**
  - ~85 audits par mois / 5-10 audits par jour
  - Audits disponibles sous 1 à 6 heures
- **> 101 Millions de lignes auditées**
  - Patrimoine couvert > 8 Millions de lignes
  - 100 000 lignes par audit en moyenne ; max = 430 000 lignes

L’audit de code dans le contexte Qualité

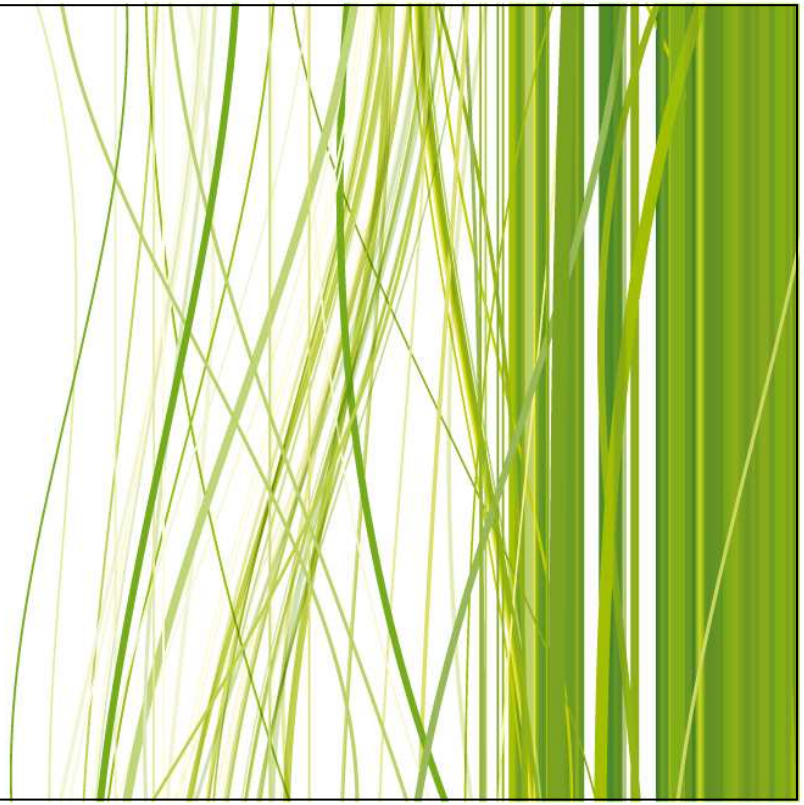
Audit de code et activités Qualité

- **Audit de code dans le cadre de Référentiels Qualité**
  - N'apparaît pas « en titre » dans les modèles ISO ou CMMI
  - Process area CMMI : *technical solution*, product integration, verification
  - Associé à revue de code, revue par les pairs, recette

- **Audit de code intégré aux autres activités de test**
  - Tests unitaires / utilisation de parser, analyseur de code (PMD, Checkstyle)
  - Tests d'intégration / utilisation de portail qualité (Squale, CAQS, CAST…)
  - Revue par les pairs (aide au ciblage)

- **Douane applicative**
  - « PV d'acceptation » du code
  - Elément de validation d'une livraison (contrat de prestation back office)
  - Prérequis pour mise en production

Gérer les résultats des audits



Que faire des résultats et avec qui ?

- **Selon l'objectif de l'audit de code**
  - Amélioration en phase de codage
  - Validation d'une livraison contractuelle
  - Photo du patrimoine applicatif
  - Expertise ponctuelle / « alerte au feu »

- **Envoyé à**
  - Uniquement au projet, au responsable de back office et … front office ?
  - Delivery manager, Contract manager, Ingénieur Qualité et … le « métier » ?
  - Responsable domaine applicatifs, urbanisme
  - Expert techniques langages, auditeurs…

Exploiter et gérer les résultats

- **Exploiter l'audit de code**
  - Dépasser le simple « état des lieux »
  - Réunion de débriefing : avec développeurs, chef de projet back et front
  - Etablissement de plan d'action :
    - Selon criticité des défauts, des axes qualités, quick win, clean-up days…
    - Ex : fiabilité et robustesse = prérequis pour mise en production
  - Idéal : avoir un outil qui aide à prioriser et évaluer les coûts et impacts des modifications envisagées
  - Même sur de l'audit de patrimoine, être prêt pour un plan d'amélioration

- **Ne pas s'arrêter au premier (unique?) audit**
  - … et mise en œuvre du plan d'action
  - vérification (nouvel audit) de l'apport des correctifs
  - Comparaison avec livraison précédente, entre diverses livraisons
  - Idéal : avoir un outil d'audit qui indique les évolutions, la couverture par rapport à un plan d'action

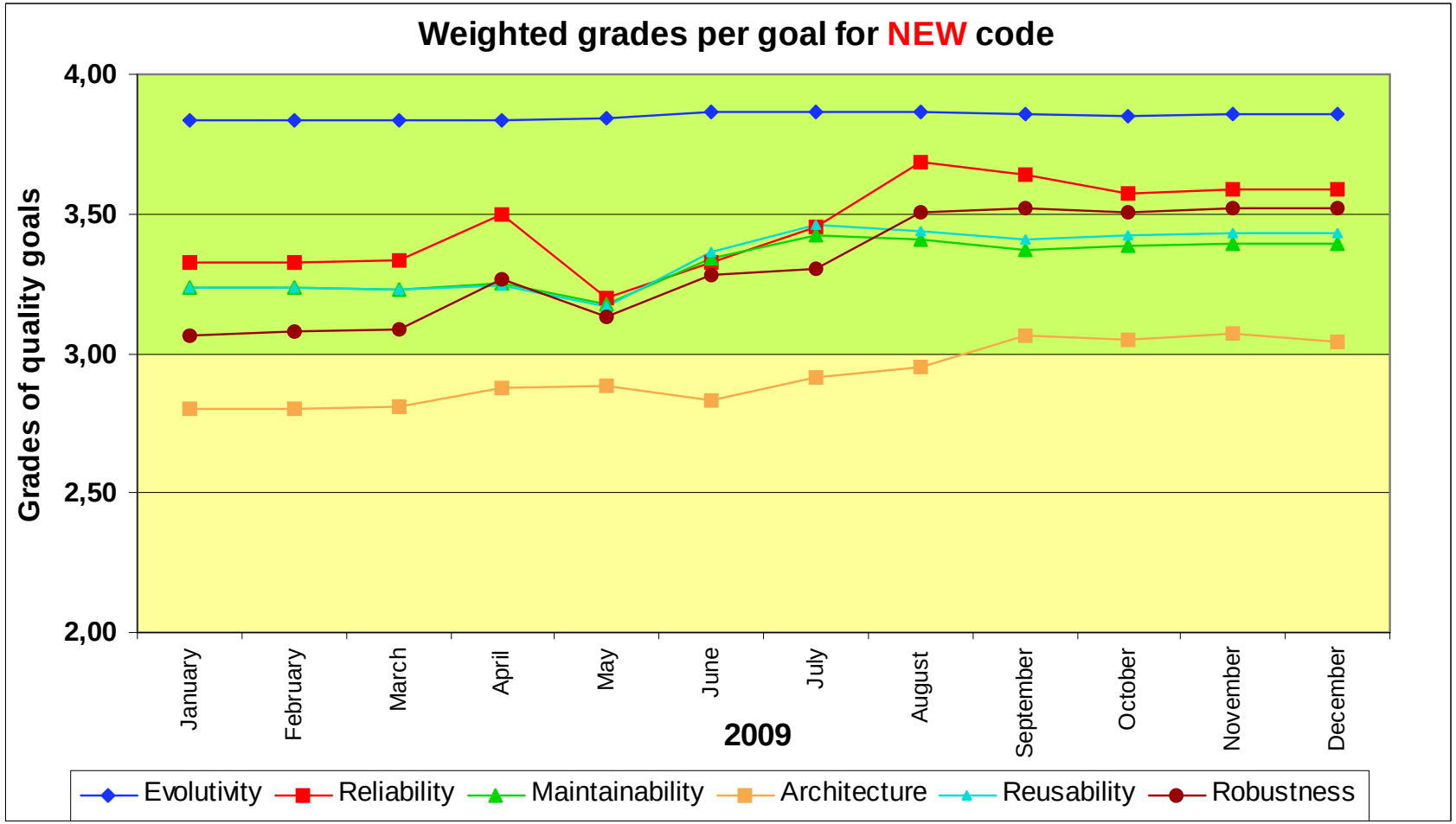
Réalisation de tableau de bord

- **Etablir des statistiques et suivis transversaux**
  - Ne pas s'arrêter aux notes d'un projet mais à un ensemble de projets
  - Tableau de bord sur le patrimoine : le dernier audit de chaque projet
  - Tableau de bord sur l'évolution du parc : suivi mois par mois

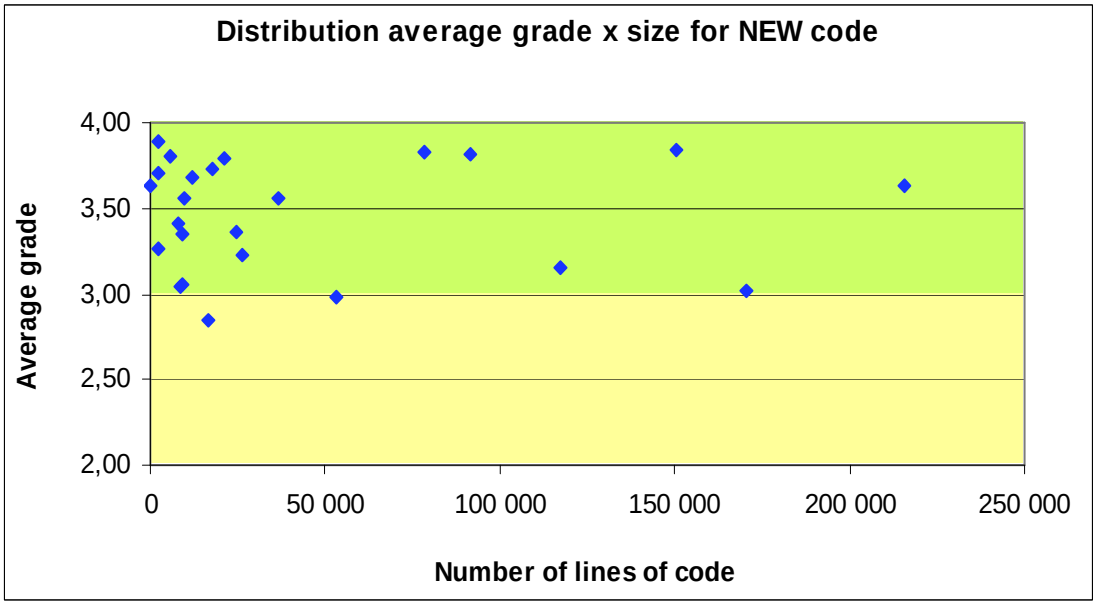
- **Sujets d'étude**
  - Par langage (Java, PhP, Cobol), par technologie (web, temps réel, transactionnel)
  - Par axe qualité (Fiabilité, Maintenabilité, Architecture)
  - Par taille en lignes de code
  - Par criticité (Stratégique, Critique, Standard, composants réutilisables)
  - Par domaine d'applications (ex : commerce, fabrication…)
  - Par centre de développement, par fournisseur

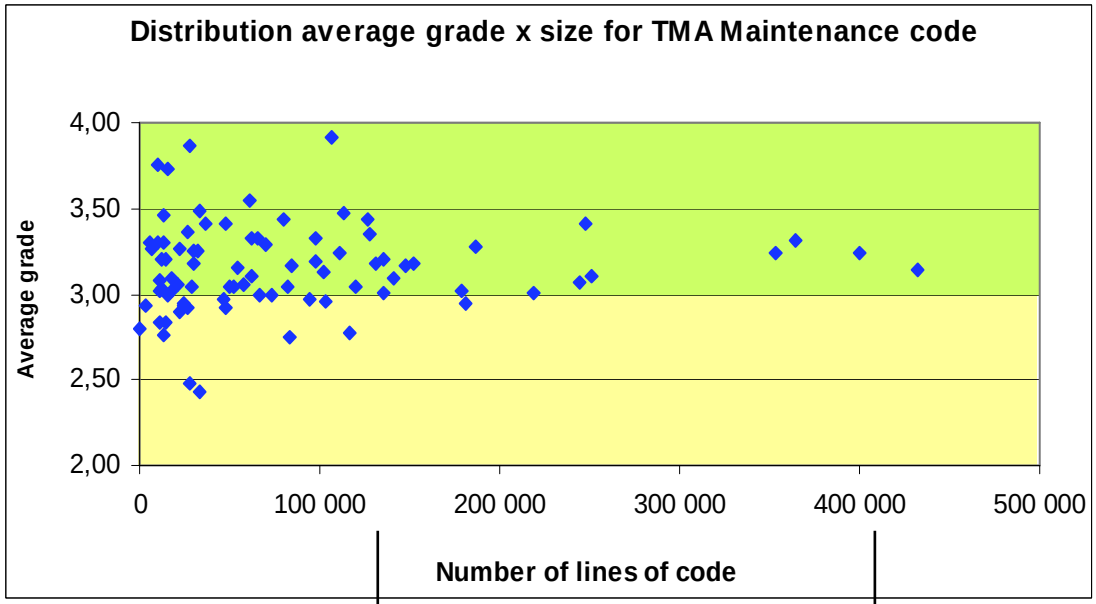
  Attention à distinguer nouveaux développements et « legacy »

Exemple de tableaux de bord : Objectifs dans le temps

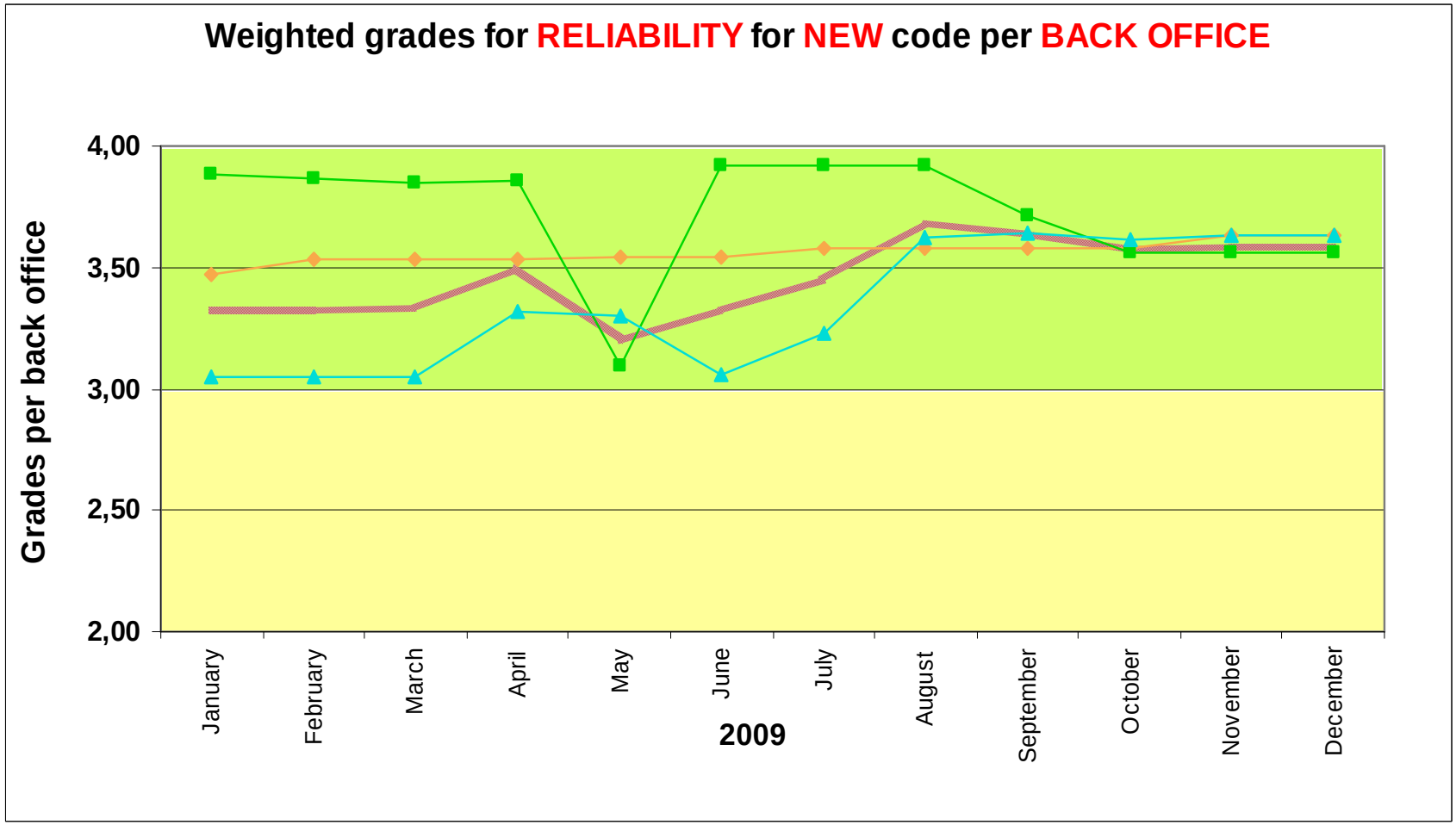


Exemple de tableaux de bord : Qualité x Taille





Exemple de tableaux de bord : Qualité x Back Office



Amélioration du modèle qualimétrique : Analyse

- **Analyse des défauts constatés**
  - Identification des défauts récurrents / fréquents
  - Identification des fausses détections de défauts
  - Analyse des notes « trop dures » sur critère ou objectif
  - Analyse des défauts constatés en recette, ou en production

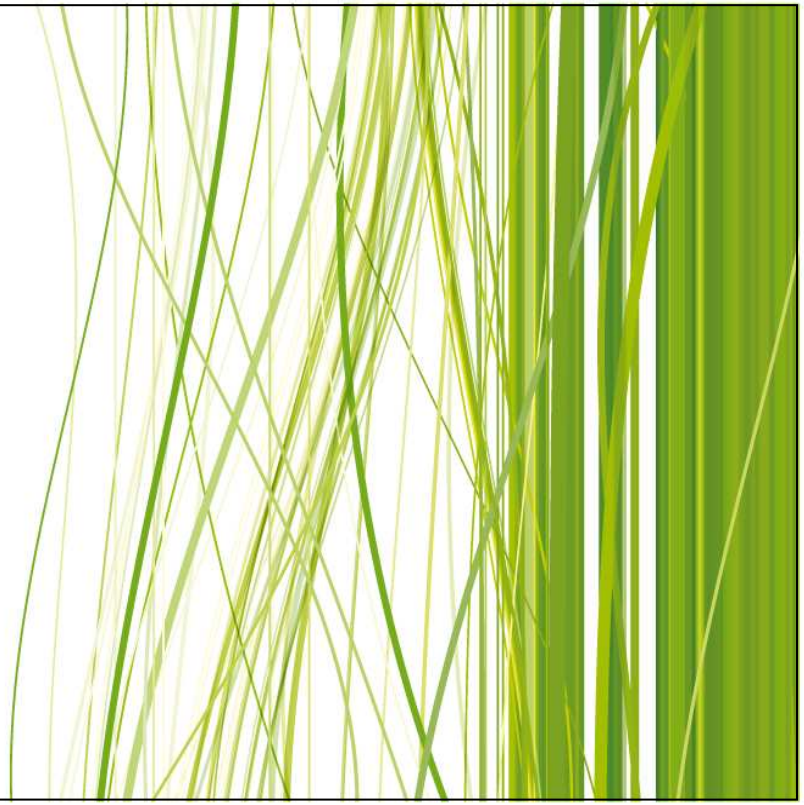
- **Analyse des plan d'actions et évolution des notes**
  - Identification des défauts incompris par les développeurs
    - Appel à l'aide pour interpréter les résultats
  - Identification des défauts « facilement » corrigés (ex : ponctuels et localisés)
  - Identification des défauts « souvent » corrigés (ex : liés à fiabilité)
  - Identification des défauts résiduels (ex : architecture sur « legacy »)
  - Identification des demandes de justification
    - Trop lourd à corriger, mauvaises pratiques / maîtrises des langages

Amélioration du modèle qualimétrique : plan d'action

- **Modification du modèle qualimétrique**
  - Rédaction des explications et exemples des critères
  - Réglages des critères (seuils, pondération, criticité)
    - Règles rédhibitoires, règles « warning »
  - Supprimer, ajouter des règles
    - Sans intérêt pour un applicatif, défaut non détecté
  - Filtrage de certains fichiers (code généré)
    - Fichiers techniques liés aux EJB
  - Cotation des coûts de correction

- **Communication**
  - Rappel sur le modèle qualimétrique appliqué
  - Rappel sur les bonnes pratiques de développement, du langage
  - Aide à l'élaboration de plan d'action

Autres gains par effets de bord



Bénéfices induits par les audits de code (1/2)

- **Qualité des développements en amont**
  - Peur du gendarme, du radar
  - Cercle vertueux : aiguillon du bon travail, du bon correctif
  - La carotte et le bâton

- **Formation des développeurs**
  - Auto-formation par les explications
  - Apprendre les règles de programmation
  - Mise en place d'experts pour aider les développeurs « de base » à développer et/ou lire les audits

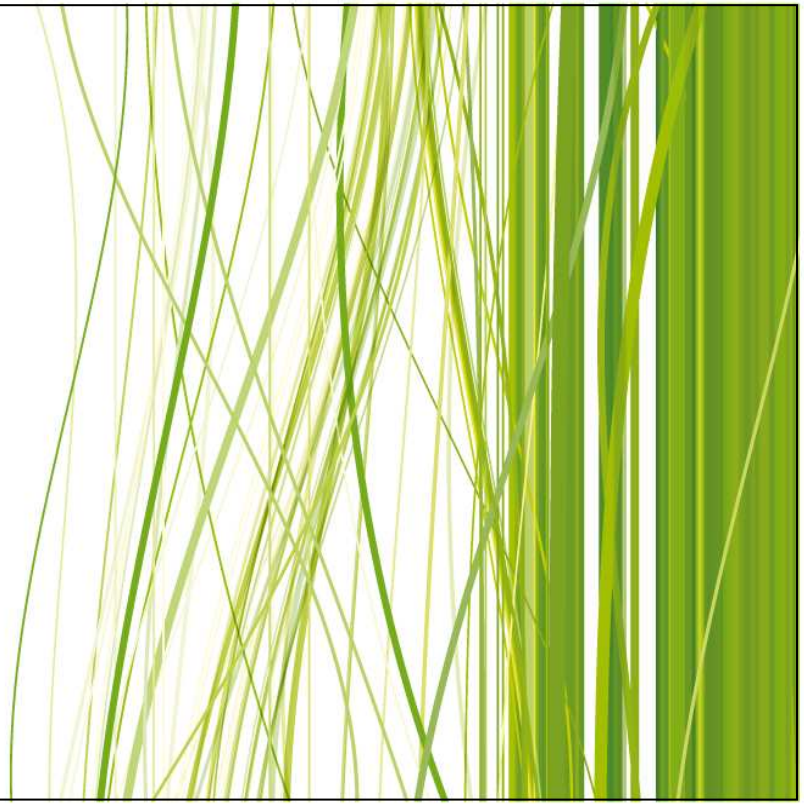
- **Super débugger**
  - Audit de code valide la qualité de code … d'un code « qui marche »
  - Mais détecte des pratiques dangereuses, bombes à retardement, parfois performances

Bénéfices induits par les audits de code (2/2)

- **Gestion du patrimoine**
  - Mise en gestion de configuration du parc logiciel -> ISO, CMMI
  - Mise en gestion de configuration du code source -> réversibilité
  - Mise en conformité des espaces de développements
    - Organisation des répertoires, des projets, des includes…
  - Détection de projets dans des langages / versions non homologuées

- **Gestion des prestations de maintenance**
  - Indicateur partiel mais existant de la qualité du parc applicatif -> urgences
  - Aide à l'estimation de charge pour une nouvelle TMA
  - Contrat factuel d'état des lieux
  - Suivi de la qualité de la prestation
  - Uniformisation des mesures / indicateurs de qualité de code
  - Règles de programmation vérifiables et vérifiées

Conclusion



Conclusion

- **Audit de code doit être un outil de travail**
  - et pas seulement une note qualité, un status
- **Le meilleur gain est en phase de développement**
  - Quand il est encore temps de corriger et ré-auditer
  - Quand il améliore la compétences des développeurs
  - Quand il prépare (« blinde ? ») l'avenir
- **L'audit de code peut être un précieux indicateur**
  - De performance des back office
  - De qualité du parc applicatif
  - Mais pas à lui tout seul
- **Et ne pas oublier de faire vivre le modèle qualimétrique**