

Audit de Code Automatisé chez Renault

Philippe BRIDON (DSIR / DQ – philippe.bridon@renault.com)



Agenda

- **Le Groupe Renault et la Direction Informatique**
- **Historique**
- **Solution technique et mode opératoire**
- **Exemple d'utilisation par un projet**
- **Bilan**
 - Gains qualitatifs directs, effets de bord et points durs
- **Objectifs 2009**
- **Conclusion**

GRUPE 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



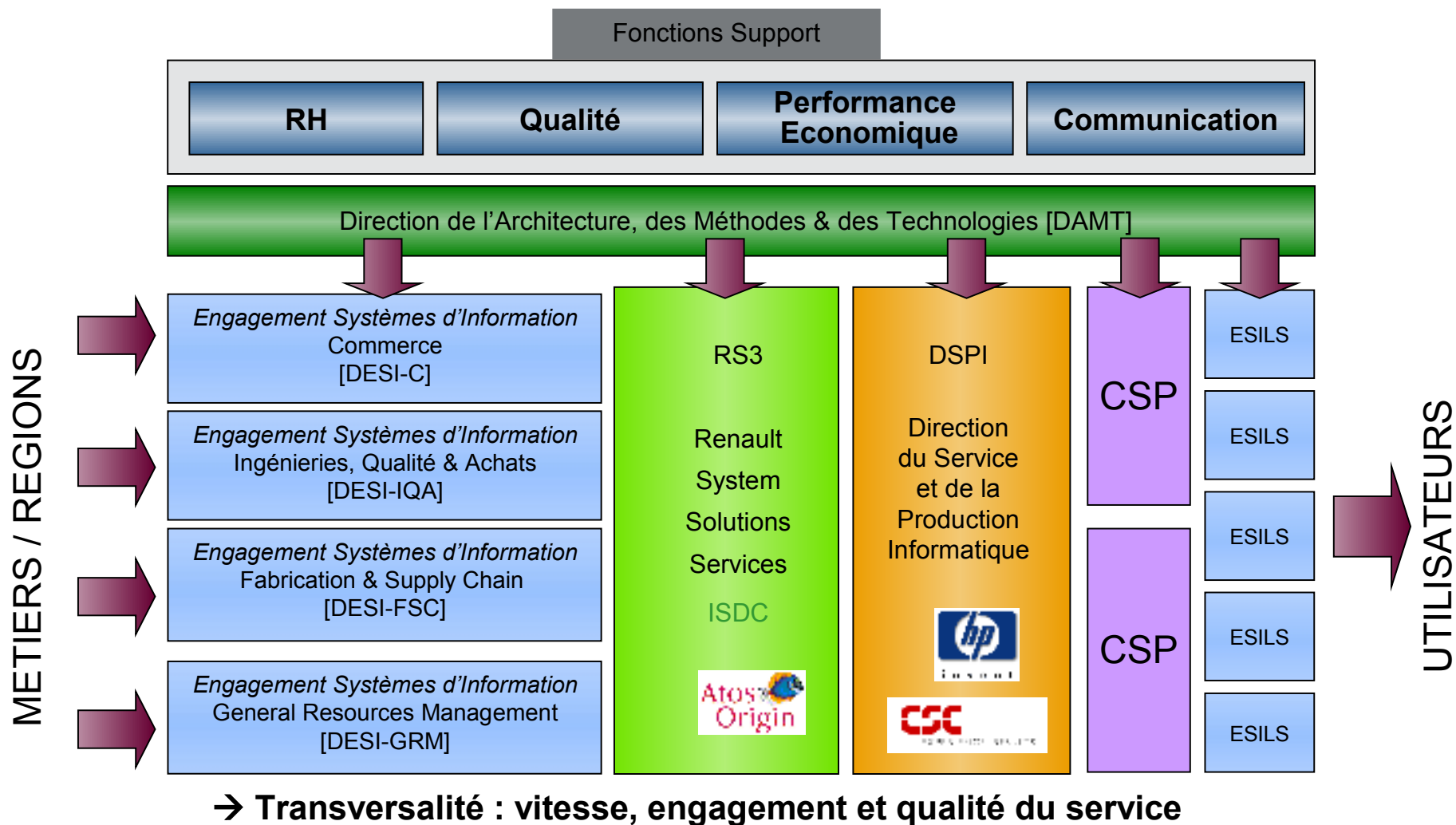
- Un partenaire stratégique AvtoVaz (LADA)



- Renault double champion du monde
En 2005 et 2006



FONCTIONNEMENT DE LA DSIR





**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

L'audit de code chez Renault

■ Situation 2006

- 100 audits manuels par an, réalisés par un forfait externalisé
- Délais d'obtention des résultats = 1 mois environ, rapport Word (30 à 200 pages)
- Faible exploitation des résultats
- Mais mise en place d'une culture qualité de code depuis 2004

■ Objectifs fixés en 2007

- **Maitriser la qualité du code du parc applicatif**
 - en rendant les **équipes responsables de la qualité de leur code**
 - en **mettant à disposition un outil d'analyse**
 - en **insérant la qualité de code dans les processus de production** des applications
 - en **systématisant les audits de code pour les projets sensibles et les jalons majeurs** de recette et mise en production

=> Projet ACA = Audits de Code Automatisés (Automated Code Analysis)

Historique du projet ACA

- **Novembre 2006 : démonstrateur**
- **Février 2007 : Ouverture du projet**
- **Avril 2007 : Note de centrage**
- **Mai 2007 : Accord investissement en Comité DSIR**
- **29 Juin 2007 : Commande fournisseur**
- **Août 2007 : mise à disposition infrastructures (Dévt, Re7, Oper)**
- **mi Octobre 2007 : début Pilote Java**
- **15 Novembre 2007 : Mise en Production**
- **18 Décembre 2007 : Accord Renault-Atos pour déploiement**

- **Janvier 2008 : Déploiement généralisé pour Java**



Volumétrie 2008 (JAVA uniquement disponible)

- **152 applications (SI) couvertes (+ 15 en pilote)**
 - Dont 91 des 95 SI de la cible (96%)
 - Dont 56% ont un audit de référence (75% sur les SI de la cible)
- **472 utilisateurs déclarés**
 - France, Espagne, Roumanie, Inde ;
 - Renault, Renault Offshore Inde, Atos Origin, Meconsa, Dacia, Satiam...
 - Dont 275 prestataires (58%), la plupart Atos Origin
 - >100 utilisateurs distincts par mois
 - ~30 sessions par jour ; pic de ~12 utilisateurs simultanés
- **> 611 audits en 2008**
 - ~70 audits par mois / 3-4 audits par jour
 - Audits disponibles sous 1 à 6 heures
- **> 36 Millions de lignes auditées**
 - Patrimoine couvert > 7,4 Millions de lignes
 - 74 000 lignes par audit en moyenne ; max = 400 000 lignes



Solution technique et mode opératoire

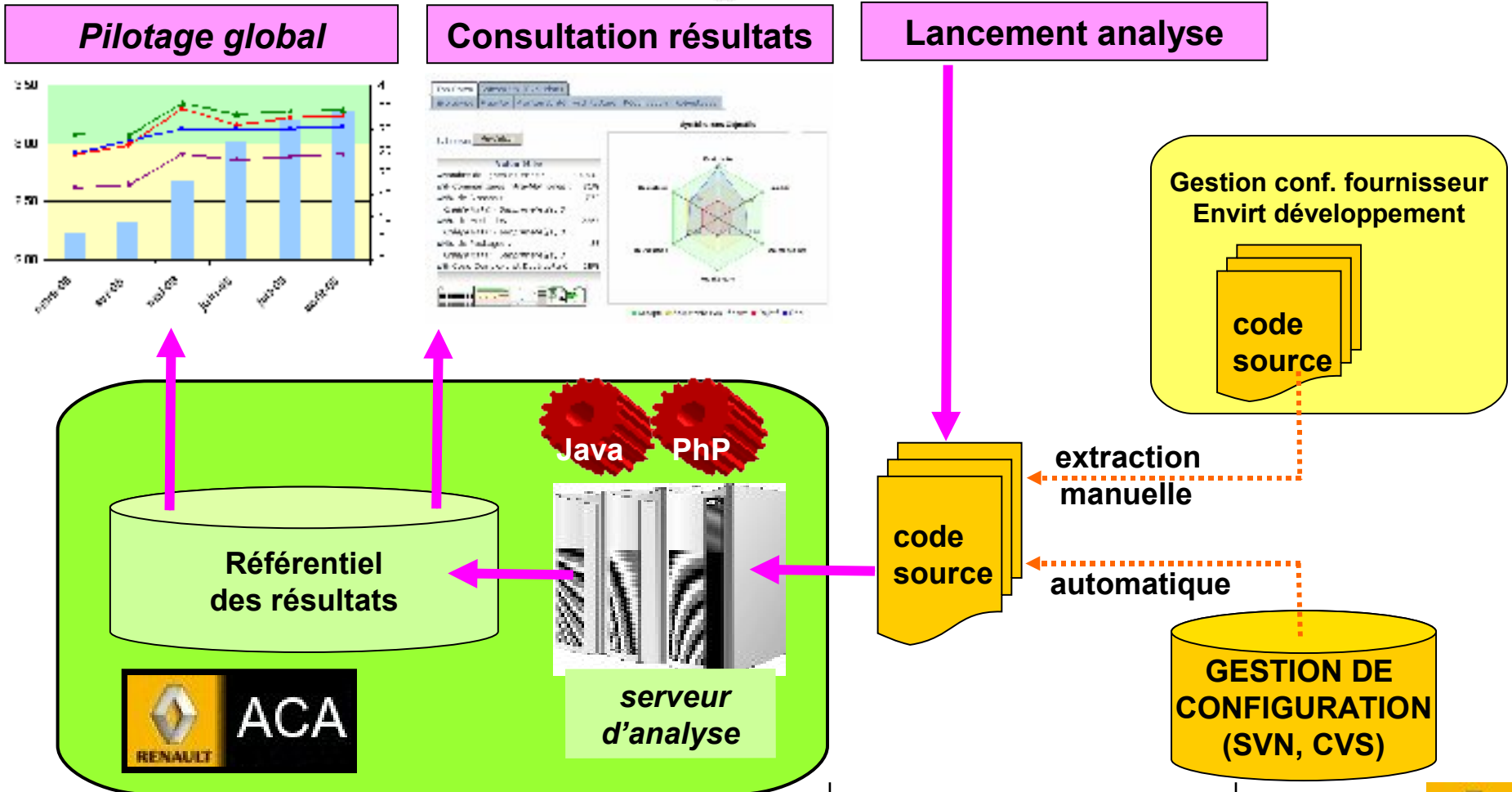


Principes de fonctionnement de la plate forme ACA

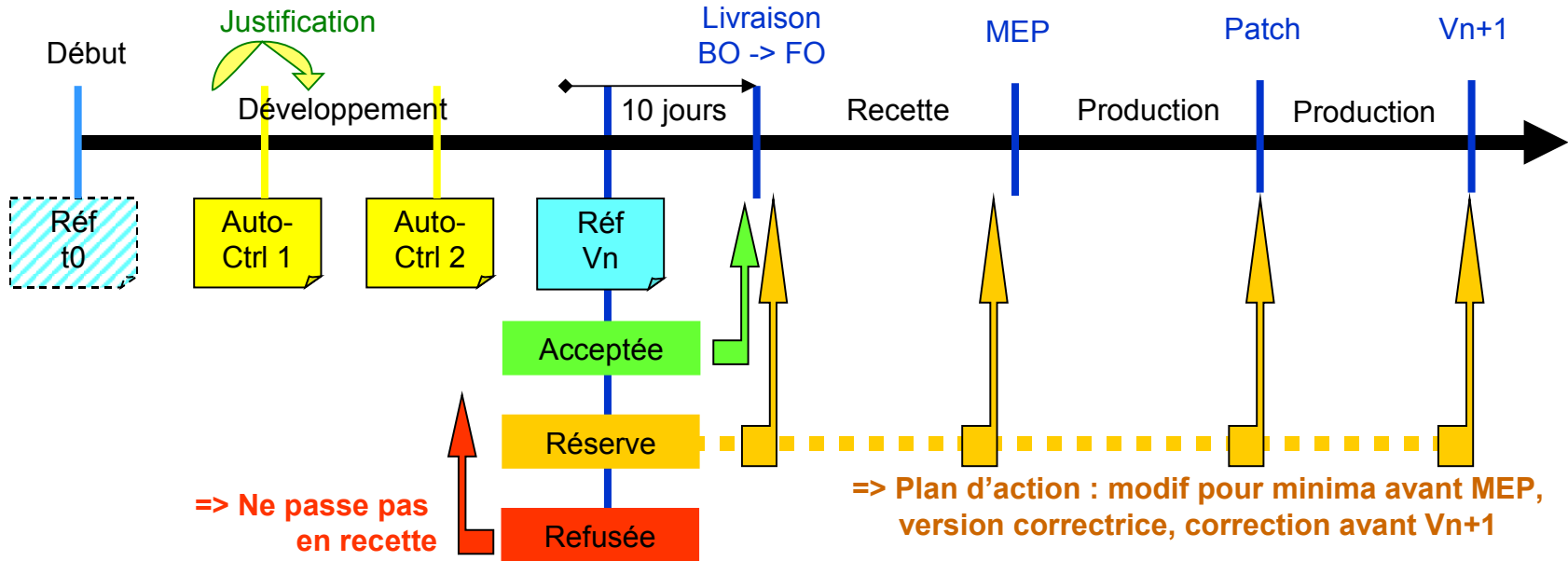
Utilisateurs :
Renault, prestataires...



Rôles : Développeur, Chef de Projet, Management, Ingénieur Qualité...



Audits de code dans le cycle de vie du projet



Audit de livraison avec objectif contractualisé (« douane applicative »)

- Projet **code neuf** : **note $\geq 3^{(*)}$** sur tous les objectifs
- TMA **Maintenance** : **ne pas dégrader $(*)$** les notes initiales

() la validation est en fait plus complexe, voir procédure d'audit ACA*

Procès-verbal d'acceptation résultat (synthèse)

NON AUTOMATISÉ À MAINTENANT

ACA Accès verbal d'acceptation d'aucun de code Nouveau code

Audit MonProjet v2.3 pour MEP Avril

Projet		Audit	
Nom	MonProjet	Exp. Final	SEP 2003
N°	1001	Typ. Audit	Mod. de Processus
Version	003	Exp. Final	MEP/MSH
		Exp. Final	MEP

Audit de code : NON CONFORME

Code de l'audit	Finalité	Intensité	Architecture	Qualité de code	Qualité de tests
-----------------	----------	-----------	--------------	-----------------	------------------

Synthétiser le niveau global et les modules :
 Niveau projet : **ACCEPTÉ AVEC RÉSERVE**
 MY_BATCHS : module **NON CONFORME**
 MY_WEB : module **NON CONFORME**
 Le projet comporte 1 module(s) conforme(s), dont le plus complexe atteint un taux de 4%.

Synthétiser les non conformités par objectif :
 Maintenanceabilité : il y a au moins un module qui n'est **PAS CONFORME** pour cet objectif.
 Architecture : il y a au moins un module qui n'est **PAS CONFORME** pour cet objectif.

Correcteurs à intégrer avant mise en production (MEP)

Remarque : les modules à intégrer doivent être en état de compilation, testés et validés, dans l'IMPÉRIATIVEMENT lors de leur intégration dans le code de production.

Plaisanceilles, Réutilisables : à intégrer à la question
 Architectures-Dépendances cycliques
 Architectures dépendant de classes internes
 Architectures d'implémentation de classes externes

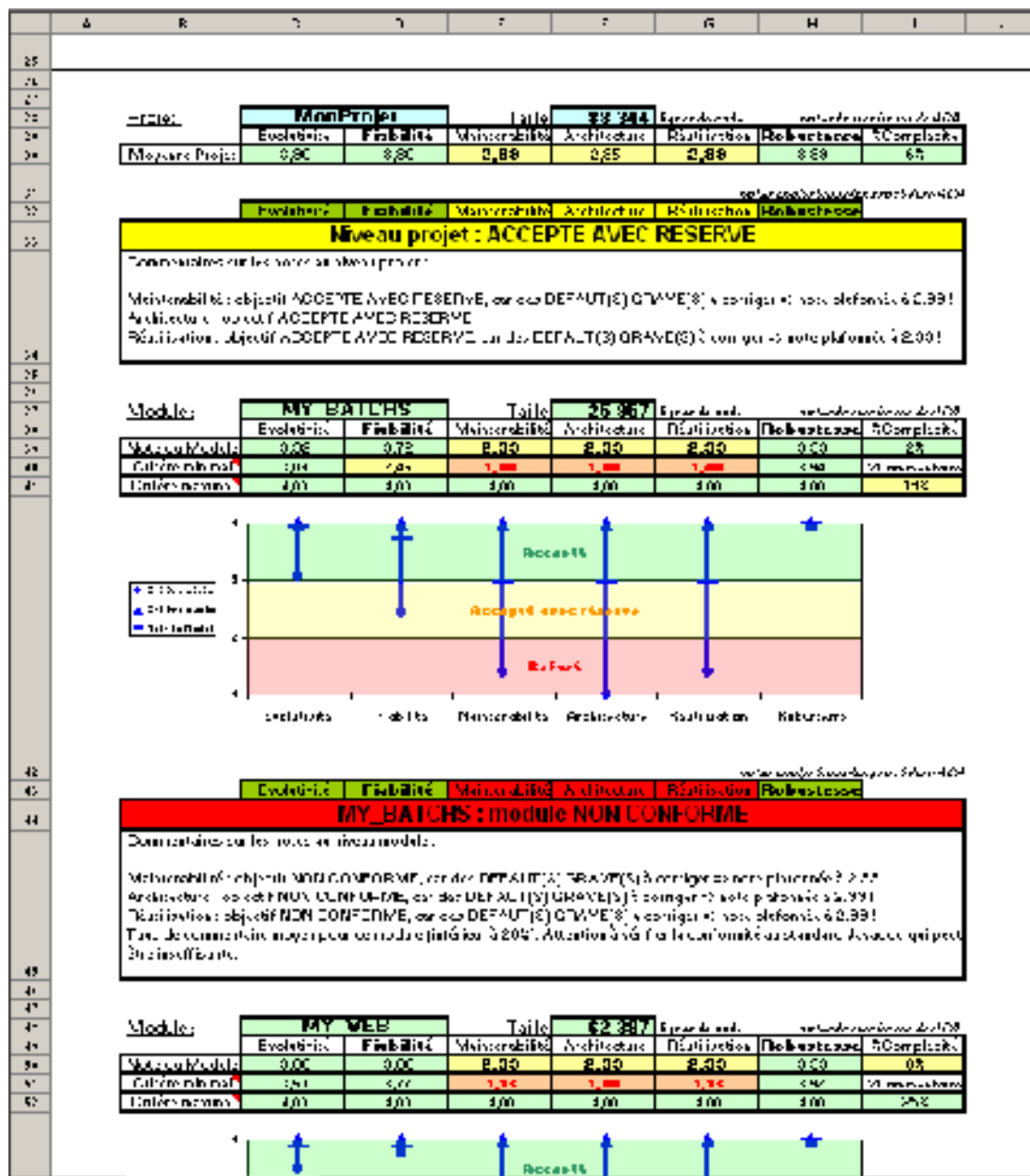
Recommandations pour la maintenance préventive

Identifier ou autoriser et évaluer les défauts à corriger sur le long terme
 mais également les défauts à prévenir qui ont des impacts sur la MEP.
 Éventuellement corriger les erreurs de langage existant en support fonctionnel pour prévenir les actions.

Procès-verbal d'acceptation

Résultats

- au niveau projet, et,
- et par module



Exemple de mise en œuvre par un projet



ACA : Synthèse des résultats

Compuware CAQS Portal - Microsoft Internet Explorer fourni par RENAULT

CAQS ACA

Gest. Qualité Administration Documentation Justification Filtrage Admin. Vues

Nombre de messages : 0

Détails du projet

19/08/08 11:45

Robustesse

Audit du 19 aout

Norme ISO 9126

Synthèse des Objectifs

Métrique	Valeur
Nombre de lignes de code	1 748
% Commentaires Intra-Méthodes	27%
Nombre de classes	41
Créé(a)(s) / Supprimé(a)(s) /	
Nombre de Modules	1%
Créé(a)(s) / Révisé(a)(s) /	
Nombre de Packages	13
Créé(a)(s) / Révisé(a)(s) /	
% Code Complexe et Déstructuré	7%
Nombre de défauts	
En attente de validation	0
Validées	0
Refusées	0

Gestion des rapports

■ Accepté ■ Acceptable avec réserve ■ Rejeté ■ Féé

ACA : travail sur l'axe « Robustesse »

The screenshot shows the ACA (Automated Code Analysis) interface. The main window displays a table of analysis results. A red box highlights the first row, which has a score of 1.00. A red text box above the table states: "Erreur rédhibitoire, forte pondération mais une seule occurrence" (Redhibitory error, high weighting but only one occurrence). A blue text box on the left says: "Explication, Exemple, Bonne pratique..." (Explanation, Example, Good practice...). A blue text box at the bottom says: "Autres critères / défauts impactant la robustesse" (Other criteria / defects impacting robustness). At the bottom right, a legend titled "Répartition" shows four categories: "Note de 1" (red), "Note de 2" (orange), "Note de 3" (green), and "Note de 4" (light green).

Intensité	Note	Unité	Statut	Répartition
1.00	1.00	0.00	0.00	15%
3.43	2.00	0.00	0.00	
3.76	2.00	0.00	0.00	
3.06	1.00	0.00	0.00	
1.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	
4.00	0.00	0.00	0.00	



ACA : Localisation des défauts « Robustesse »

The screenshot displays the CAQS ACA tool interface. The top navigation bar includes 'CAQS' and 'ACA' logos, along with a 'Context Layout' dropdown. The main window shows a code review of a Java class, 'SajTilesRequestProcessor.java', located at 'c:\ppl\CAQS\src\IRA-52508_SAJ\client_WRR\src\saj\com\client\struts\SajTilesRequestProcessor.java'. The code is displayed in a text editor with a yellow highlight on line 37, which is the end of a try-catch block. The code is as follows:

```
26 public class SajTilesRequestProcessor extends StrutsLayoutRequestProcessor {
27     protected boolean processRequest(
28         HttpServletRequest request,
29         HttpServletResponse response) {
30         try {
31             request.getRequestDispatcher(getClass().getServletContext());
32             response.setContentType("text/html");
33             return true;
34         } catch (Exception e) {
35             Logger.getLogger(getClass()).debug(
36                 "Exception in tiles request processor");
37         } catch (Exception e) {
38         }
39     }
40     return true;
}
```

ACA : Contrôle de l'amélioration sur la synthèse

Comparteur CAQS Portal - Microsoft Internet Explorer fourni par RENAULT

ACA

Nombre de messages : 0

Détails du projet

22/00/CD 12 25

Audit du 22 aout

ValeurMétric

Nombre de lignes de code:	1707
% Commentaires intra-Méthodes:	25%
Nb de Classes:	20
<i>(Créé(s)/) - Supprimé(s)/)</i>	
Nb de Méthodes:	156
<i>(Créé(s)/) - Supprimé(s)/)</i>	
Nb de Packages:	17
<i>(Créé(s)/) - Supprimé(s)/)</i>	
% Docs Complets et Structurés:	7%
Nombre de Validations:	
En attente de validation:	0
Validées:	0
Refusées:	0

Gestion des rapports

Synthèse des Objectifs

Architecture

Maintenance

Modélisation

Performance

Portabilité

Testabilité

● Accepté ● Accepté avec réserve ● Rejeté ● Réé

ACA : Contrôle de l'amélioration par comparaison

Computer CAQS Portal - Microsoft Internet Explorer fourni par RENAULT

CAQS ACA

Gérer qualité | Administrateur | Configuration | Justification | Paramètre | Contrôle qualité

Nombre de messages : 0

Détails du projet

008-RTK-2200-2A7

008-RTK-2200

008-RTK-2200_Sw

008-RTK-2200

008-RTK-2200

008-RTK-2200

Top Down | Bottom Up | Evolutions

Synthèse | Nouveau en anomalie | Dégradés | Améliorés | Arrêtés et dégradés | Clés en anomalie

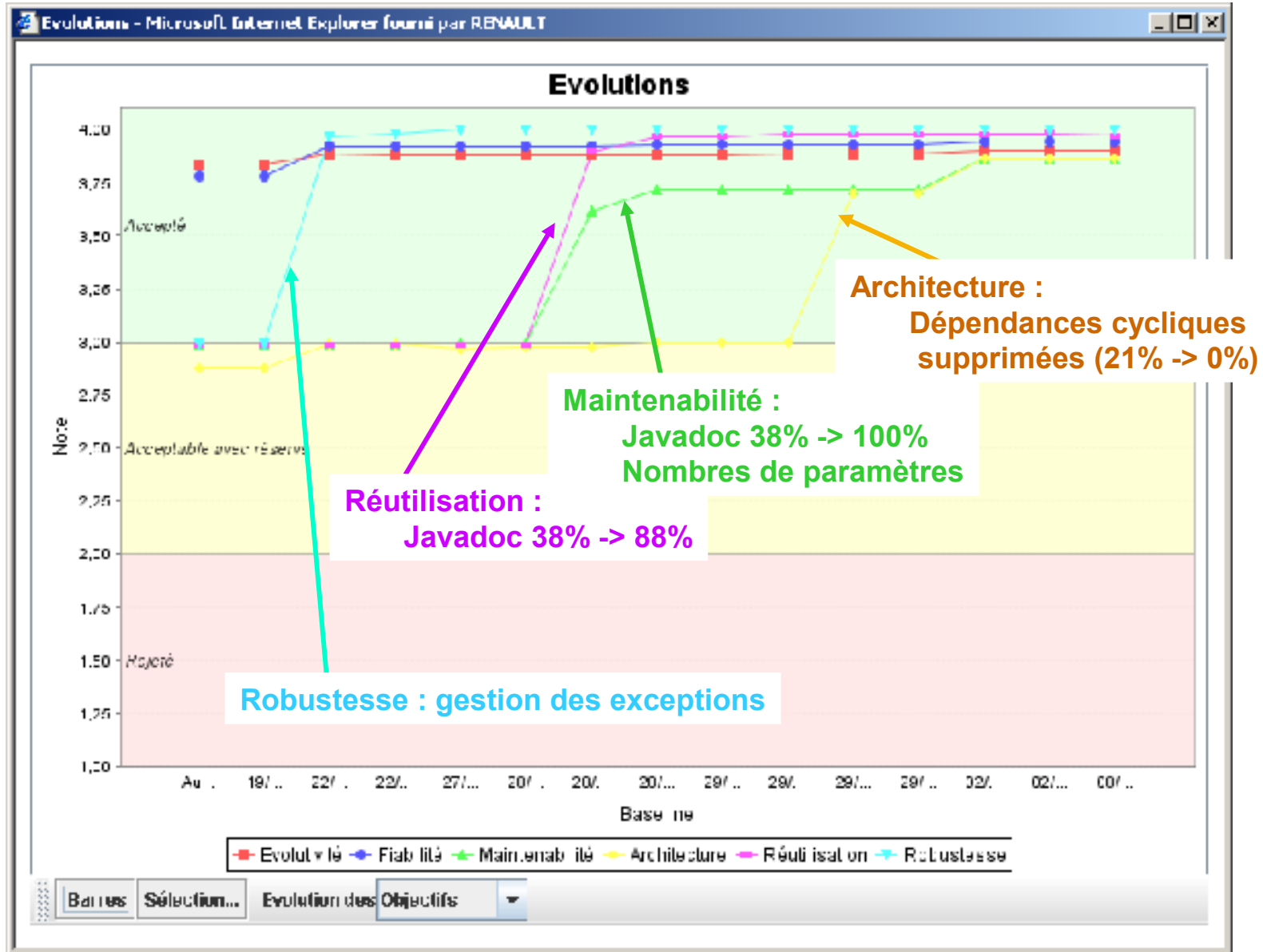
Valeur cible		Objectif	Actuel	Evolution
Hélements nouveaux	27	0,70	0,70	→
Hélements améliorés en anomalie	2	0,98	0,98	→
Éléments dégradés	4	0,50	0,50	→
Éléments améliorés	10	2,95	2,95	→
Éléments arrêtés ou dégradés	2	0,97	0,97	→
Éléments stables	741	2,95	2,95	→
Hélements arrêtés en anomalie	48			
Hélements supprimés	20			

Synthèse version 19/08/08 11:15

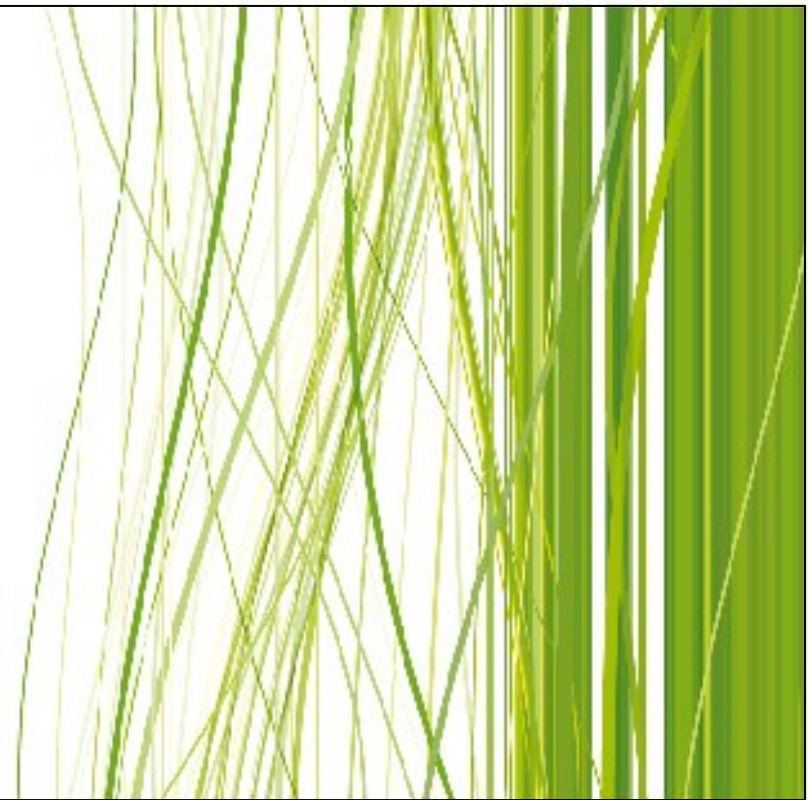
Synthèse version 22/08/08 12:21

● Accepté ● Acceptable avec réserve ● Rejeté
● Rév ● Rév

ACA : Pilotage de la qualité au cours du temps



Bilan de l'utilisation



Gains qualitatifs directement liés

- **Amélioration de la qualité des projets**
 - Sur les projets « neufs » (ex : dépendances cycliques, fiabilité, robustesse)
 - comme sur la maintenance (pas de dégradation, plan d'action immédiat sur les « quick win »)
 - ... et les équipes qui se prennent au jeu (« super debugger »)
- **Gain sur le « rework » et recette**
 - Prise en compte dès le back office grâce aux audits d'autocontrôle en libre service et « en toute impunité »
 - Prise en compte avant recette grâce à l'automatisation, la rapidité d'analyse, la navigation dans le code et le suivi comparatif
 - Sorte de correcteur orthographique : les erreurs de base sont corrigées
- **Factualiser la qualité des livraisons de code**
 - Front office accepte livraison « en conscience » de sa qualité
 - Possibilité de cibler un plan d'action immédiat



Effets de bord

- **Meilleure connaissance du patrimoine**
 - Qualité « en l'état » du parc
 - Qualité initiale après migration RPL -> Cobol
- **Aide à la réversibilité**
 - Changement de prestataire, Transfert en Inde
- **Aide au plan de fiabilisation, performance...**
 - Même si les notes sont bonnes, recherche de critères / défauts spécifiques
- **Pour l'automatisation des audits, les projets se mettent en gestion de configuration conforme**
- **Auto-formation des développeurs, revue par les pairs, tutorat**
 - Remonte un besoin de seniors, architectes...

Quelques points durs

- **Mise au point du modèle qualimétrique**
 - Nombreuses variables de réglage (pondération, criticité, moyenne)
 - Règles automatisables versus règles souhaitées
 - Attention aux notes moyennées et pondérées
-> rédaction d'une procédure plus inquisitrice et PV associé
- **Négociation avec les partenaires de développement**
 - Accord sur le modèle qualimétrique : bataille d'experts
 - Quel engagement sur les défauts découverts
 - Passer de l'autocontrôle sans fin à l'audit officiel et en temps avant une livraison
- **Faire appliquer la douane applicative**
 - Renégocier le contrat de prestation
 - Mise en visibilité des PV d'acceptation
 - « oser » geler une mise en production « pour de la qualité »



Prochaines étapes



ACA en 2009 : agrandir la couverture de langages

■ Déploiement Java



- Objectif 2009 : audits sur les jalons/projets majeurs (« N1 » et « N2 »)
 - 52 projets cibles (que Java) + audit « à temps » (à la livraison du code)
- Mise en place de CAQS v4.2 avec aide au plan d'action

■ Mise en place de nouveaux langages



- PhP

- forte demande et croissance pour les « développements rapides » => audits rapides !



- VB.NET & C# :

- Notamment pour valider les composants sur Sharepoint



- Cobol mainframe :

- pour l'après migration RPL (Renault Programming Language) vers Cobol



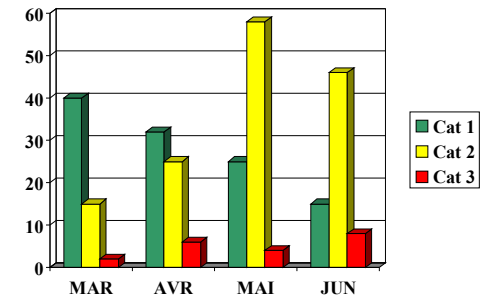
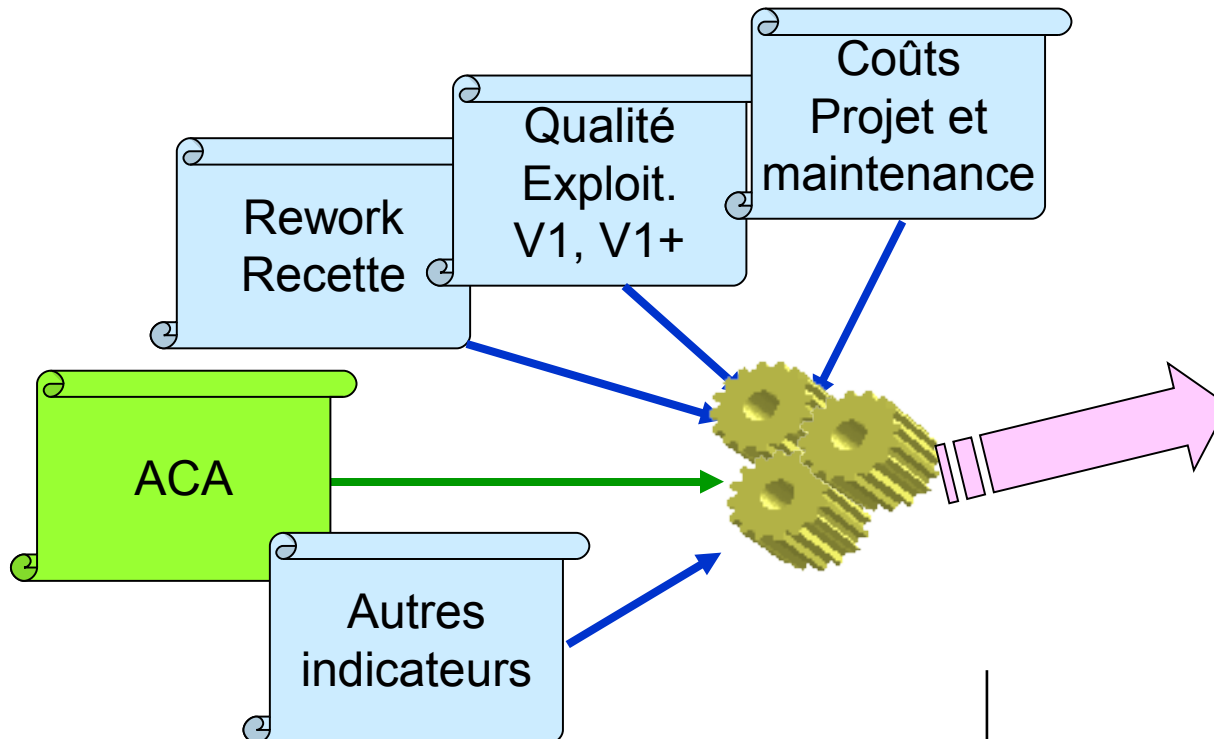
- SAP / ABAP

- Pour développer l'audit de code automatisé dans le monde ABAP



ACA en 2009 : consolider l'analyse qualitative

- **Officialiser la « douane applicative »**
 - Geler une mise en production pour cause de non qualité
- **Savoir corréler qualité de code et :**
 - Rework Recette (Outil de gestion de tests)
 - Incidents d'Exploitation (Outil de gestion des incidents, comité incidentologie)
 - Coûts de maintenance, de Point Fonction



Conclusion

- **Choix gagnant d'une plate-forme à disposition des développeurs**
 - Appropriation par ceux qui codent
 - Equipe support ACA minimale = 1 chef de projet + 1 administrateur
- **Choix gagnant d'audits rapides, automatisés et faciles à exploiter**
 - Mieux vaut quelques règles systématiquement testées que beaucoup non appliquées
 - Les correctifs sont appliqués avant livraison
- **Apport « prouvé » de l'audit de code sur le parc Java**
 - Impatience des projets à élargir le spectre des langages couverts
 - C'est devenu « culturel »

