Power Week

Université IBM i 2019

22 et 23 mai

IBM Client Center Paris

IBM

for Business

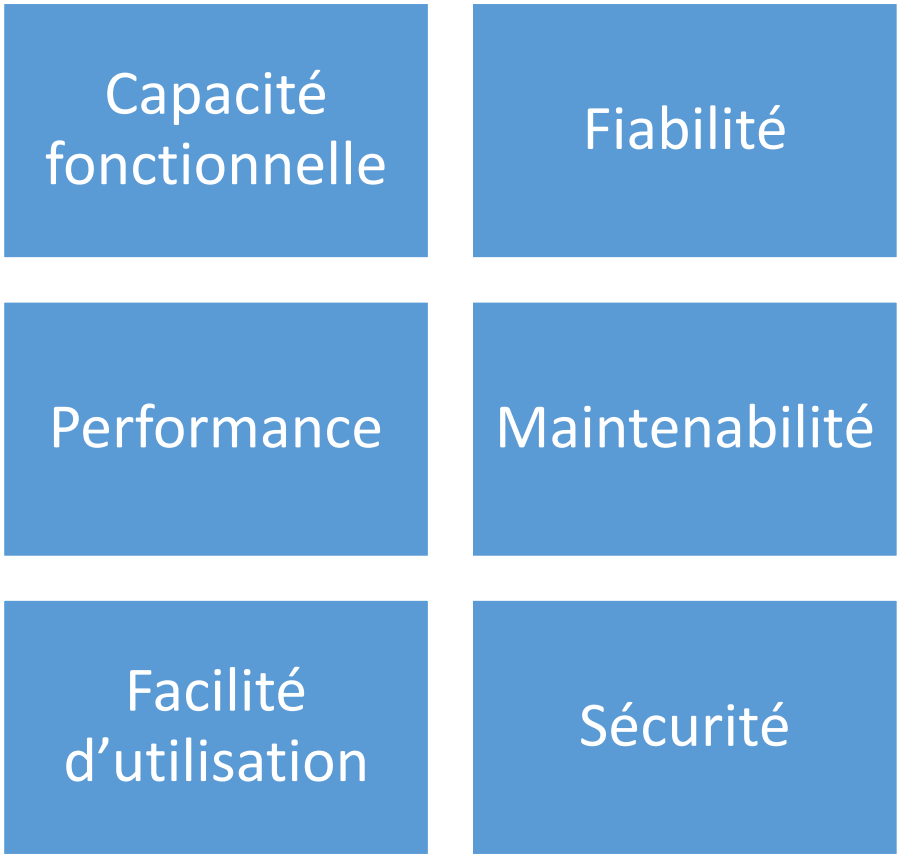
S43 – Mesurez la qualité du code de vos applications RPG

Véronique LEOTARDI – Olivier ARNAUD
IBM France
vleotardi@fr.ibm.com / olivier.arnaud@fr.ibm.com

Agenda

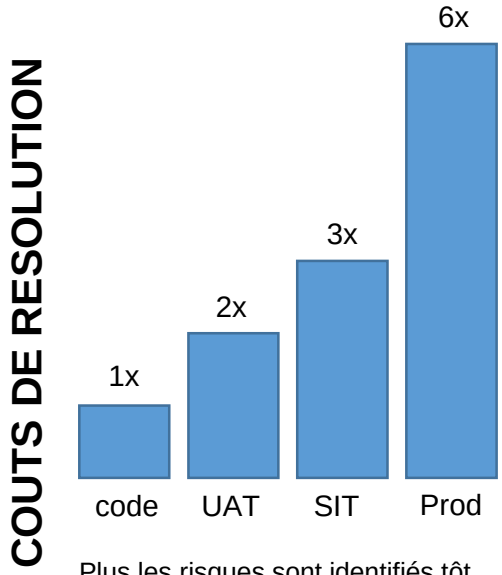
1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils

- La **qualité logicielle** est une appréciation globale d'un logiciel, basée sur de nombreux critères définis par la norme ISO/IEC 25010



- La non-qualité **coûte cher**

- Plus le problème est pris tardivement, **plus il coûte cher**:



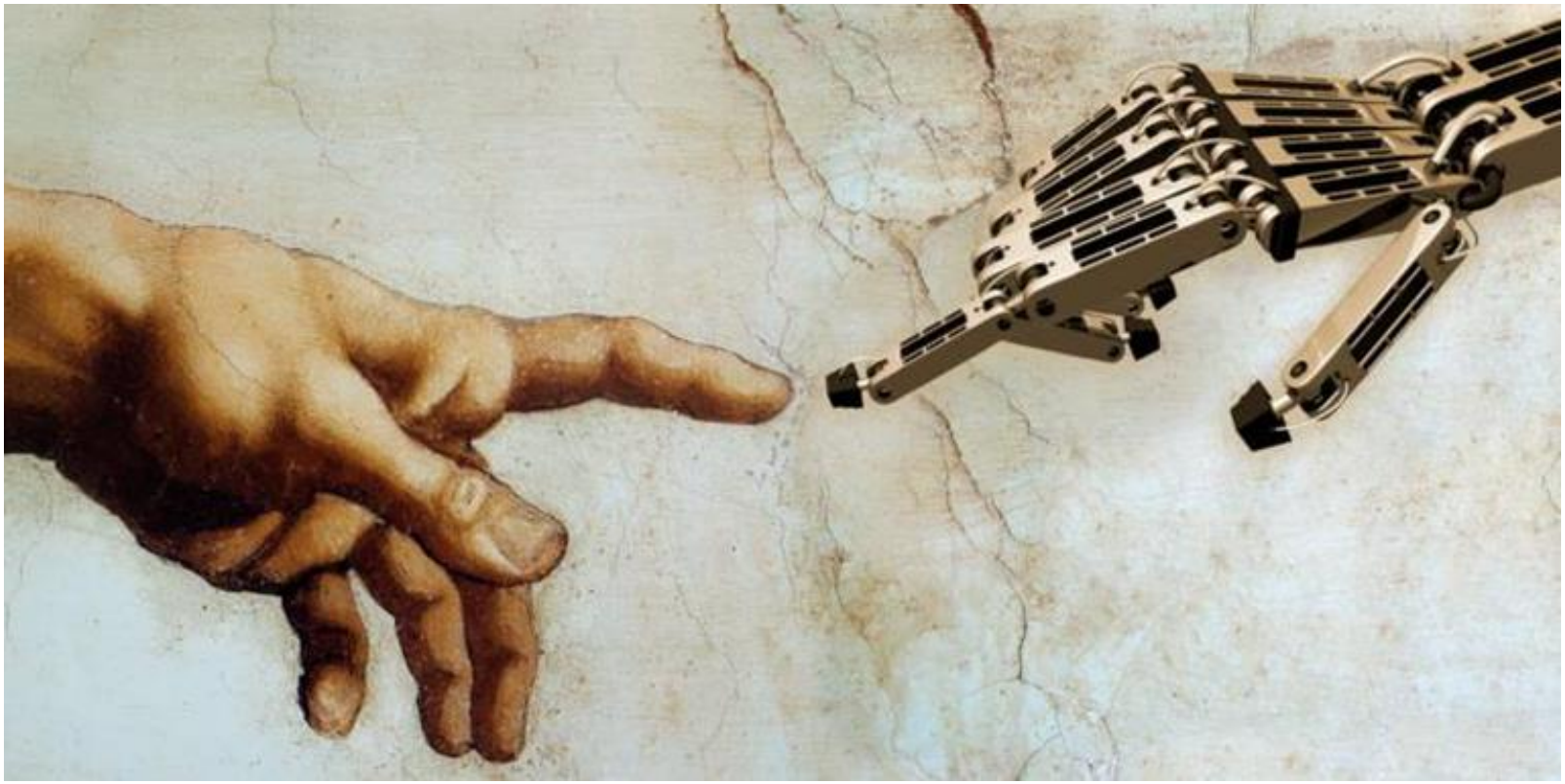
Plus les risques sont identifiés tôt
Moins le coût de résolution est élevé

Source: Economics of Software Quality – Jones & Bonsignour;
Software Defect Analysis: Anaempirical Study of Causes and Costs in the IT Industry;
CAST Report on Application Software Health (CRASH 2012) – Curis, Sappidi & Szynkarski

Le contrôle qualité – Comment ?

Manuellement
- Revue ponctuelle du code
- Onboarding
- Chiffrage de la reprise
- Problème ponctuel

Automatiquement
- Revue régulière
- A chaque livraison
- Analyse des écarts
- Plans d'action
- Intégré au cycle de développement



Le contrôle qualité – Pour qui ?

Développeur

- Montre les **faiblesses** ou **risques potentiels**
- Permet **d'optimiser** les opérations de correction
- Donne des **éléments techniques** pour faire des **estimations**

Architecte

- Montre la **qualité** et la **complétude** de **l'architecture**
- Obtient des détails sur les domaines à améliorer à moindre coût

Testeur

- Anticipe les efforts de recette des objets à tester
➔Eléments de complexité de l'application
➔Eléments appelés par un composant

Chef de projet

- Base sa communication sur **des faits et des mesures**
- Suit les **résultats** sur la base d'une **amélioration continue**
- Permet des **remédiations rentables**

Client

- Base sa **communication** sur des **faits** et des **mesures**
- Suit les **résultats** sur la base d'une **amélioration continue**
- **Challenge l'infogéreur** (Comité de pilotage, Contrat…)

Utilisateur final

- **Bénéficie de façon indirecte** de tous les précédents avantages en disposant d'une application **de bonne qualité au fil de ses évolutions**

1 Le contrôle qualité – Les principaux freins à sa mise en œuvre

DIFFICULTÉ TECHNIQUE :
difficulté à définir clairement le périmètre de mesure et les indicateurs à mettre en place.

Règles par défaut existantes

MANQUE DE TEMPS :
complexité à intégrer le suivi dans l'organisation de l'entreprise.

Automatisation des traitements d'analyse et de restitution

FACTEUR HUMAIN :
difficulté à impliquer les collaborateurs, notion taboue dans l'entreprise.

Tableaux de bord

Source:
AFNOR - ENQUETE NATIONALE
LES COUTS DE LA NON-QUALITE DANS L'INDUSTRIE
SEPTEMBRE 2017

Agenda

1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils

La qualité tout au long du cycle de vie

Initialisation

- Connaissance de l'application (T)
- Réduit la courbe d'apprentissage de l'équipe (T)

Conception

- Étude sur l'architecture (A)

Management

- Prendre des decisions

Analyses

- Analyse d'impact (A)
- Connaissance des applications (T / A)
- Optimisation (A)

Construction

- Faits pour construire des estimations (A)
- Aide pour la retro documentation (T)

Développement

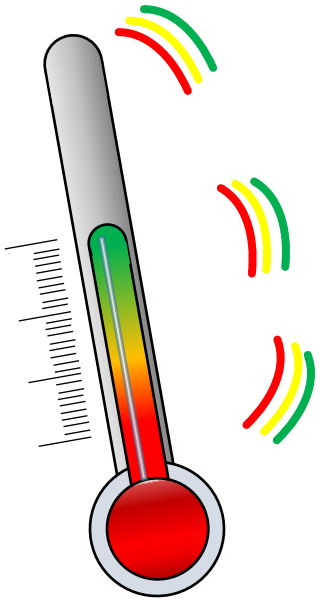
- Évalue la qualité du code (A)

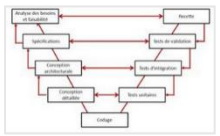
Tests

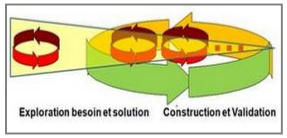
- Failles de sécurité (A)
- Intégrité architecturale (A)
- Organisation des tests (A)

Déploiement

- Système d'amélioration continue (A)







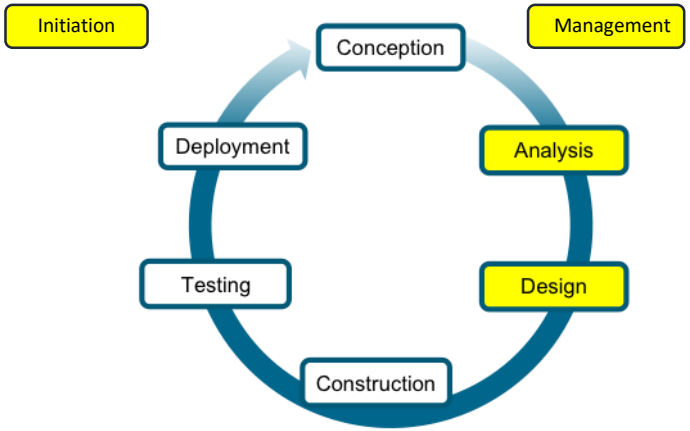
(T) Transition
(A) ADM

Principaux cas d'utilisation: Transition

Avant la transition :

Accélérer la reprise de l'application:

- Focus sur les principaux domaines à risque
- Aider à dimensionner le plan technique de transition basé sur des faits



Equipe

- Identifier les besoins de formation
- Réduire la courbe d'apprentissage de l'équipe
- Accélérer la connaissance de l'application

Documentation

- Voir l'état factuel de la documentation du code

Avant, pendant, après la transition :

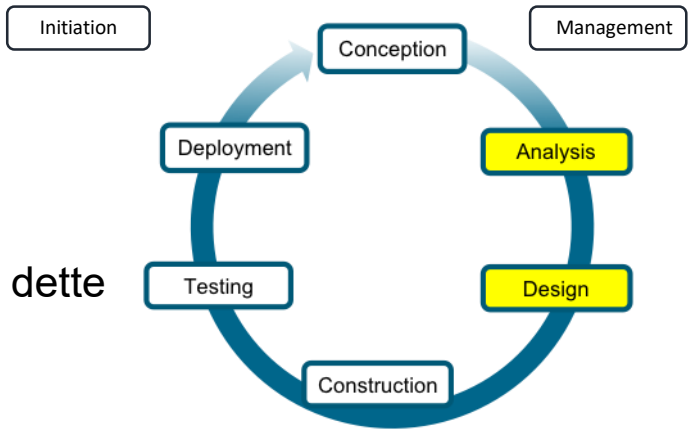
- Fonder la communication interne/avec le client sur des faits et des mesures
- Identifier les faiblesses de l'application (failles de sécurité, problèmes de performance…)
- Démontrer la qualité et la conformité à l'architecture et aux règles de codage

Principaux cas d'utilisation: Opérations ADM (1/2)

Design / Conception / Analyses d'impact

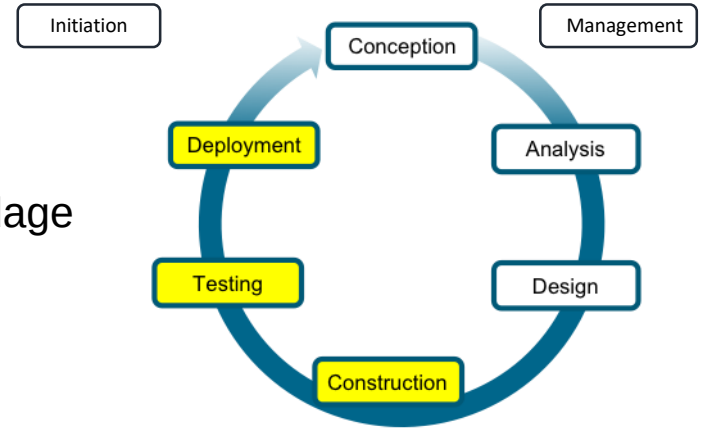
- Obtenir des faits pour bâtir des estimations
- Obtenir de l'aide pour gérer des composants complexes
- Identifier les risques et faiblesses de l'application
  - Sécurité, performance, …

- Mettre en œuvre des plans d'action pour la réduction de la dette technique et des risques



La qualité tout au long de la construction, du test, du déploiement…

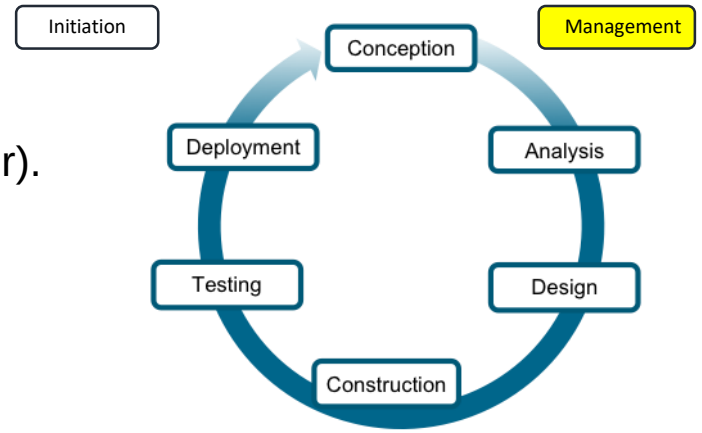
- Identifier les risques et faiblesses de l'application
  - Sécurité, performance, …
- Anticiper les efforts de test
- Montrer la qualité et la conformité à l'architecture et au codage des règles
- Contrôler au fil des livraisons l'amélioration de la qualité



Principaux cas d'utilisation: Opérations ADM (2/2)

Documentation

- Accroître la connaissance de l'application
- Mettre à jour la documentation technique à la disposition des nouveaux venus et des membres de l'équipe (turn-over).



Communication interne / avec le client

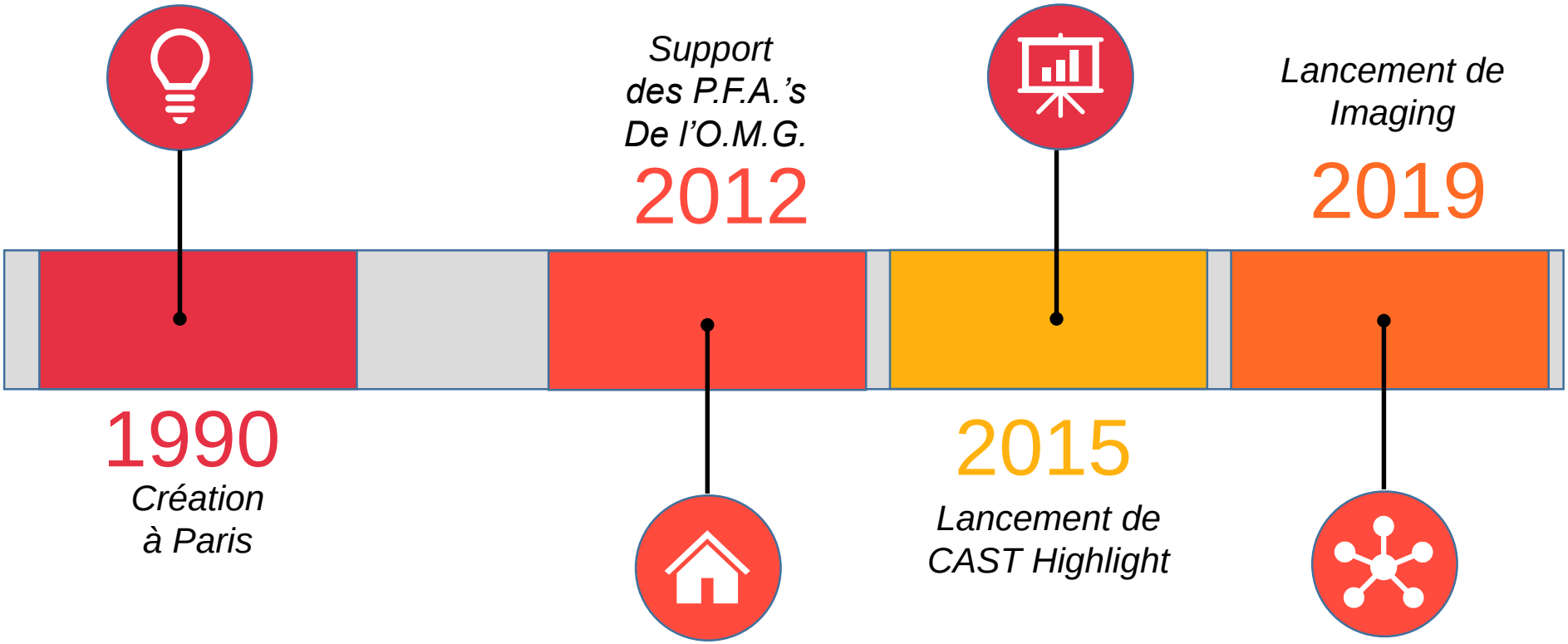
- Fonder la communication sur des faits et des mesures.

Management

- Aider à la prise de décision
- Fonder la communication sur des faits et des mesures
- Permettre de faire des remédiations à juste coût:
    - Identification des priorités associées au contexte client

Agenda

1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils



Principaux produits CAST pour la qualité logicielle :
- CAST Application Intelligence Platform
- CAST Highlight

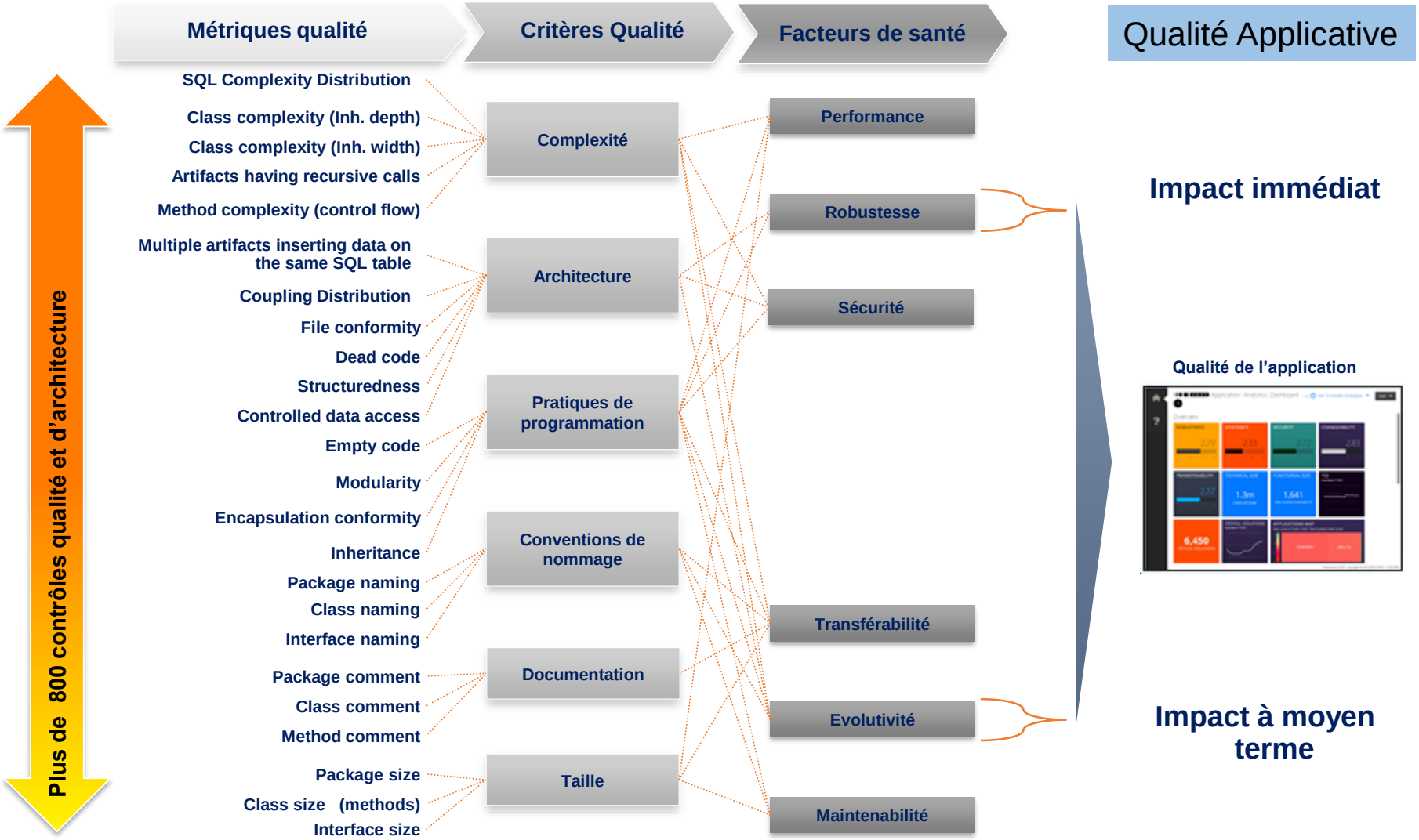
Localisation
Amérique du Nord - Europe - Inde - Chine

Les facteurs de santé dans le modèle qualimétrique CAST (Normalisé par le CISQ)

Transférabilité	Mesure de l'effort pour **transférer l'application vers une nouvelle équipe interne ou externe** ou vers **un nouveau membre au sein de l'équipe actuelle** *Exemple : Avoid unreferenced Sections and Paragraphs*
Evolutivité	Mesure de l'effort **pour implémenter une modification (évolution ou correction) au sein d'une application** *Exemple : Avoid using GOTO statement*
Robustesse	Mesure du **risque d'introduire un défaut lors d'une modification de l'application (stabilité) et de l'effort de test (testabilité)** *Exemple : Avoid empty catch blocks*
Performance	Mesure du risque de **non-performance actuelle ou future de l'application en fonction de sa conception et de son architecture** *Exemple : Avoid OPEN/CLOSE inside loops*
Sécurité	Mesure du **risque de faille potentielle de la sécurité d'une application** *Exemple : Close database resources ASAP*

Le TQI ou Total Quality Index est une agrégation des 5 facteurs de santé.

Le modèle qualimétrique CAST (1/2)



CAST est construit sur un modèle base sur les moyennes pondérées

Le modèle qualimétrique CAST (2/2)

Complet

Paramétrable

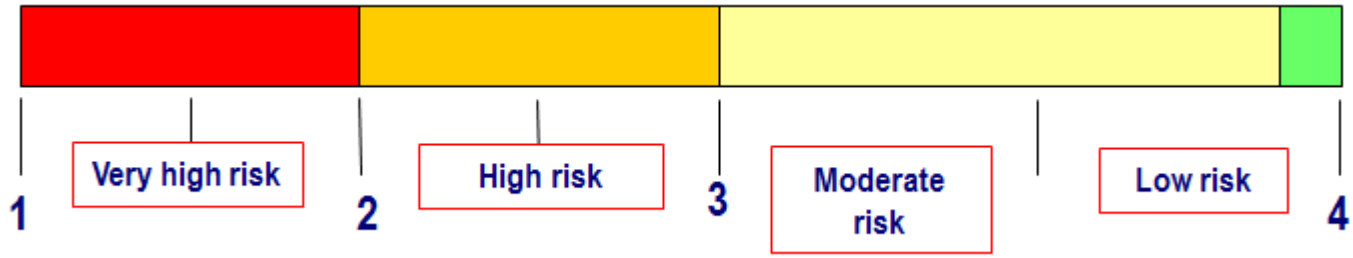
Evolutif

10 ans d'expérience dans la technologie RPG

1

Un nombre très élevé de violations dans le code source

Une application présentant des risques très élevés

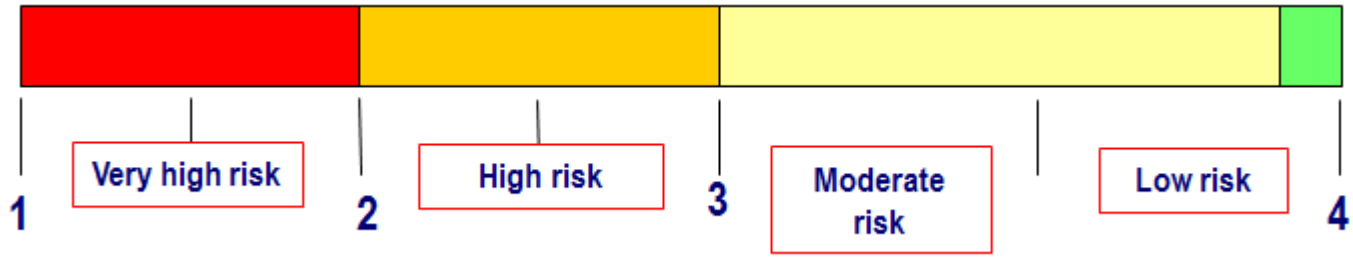


La recommandation de l'éditeur:
Des notes >3 pour un risque maîtrisé

2

Un nombre important de violations dans le code source

Une application présentant des risques élevés

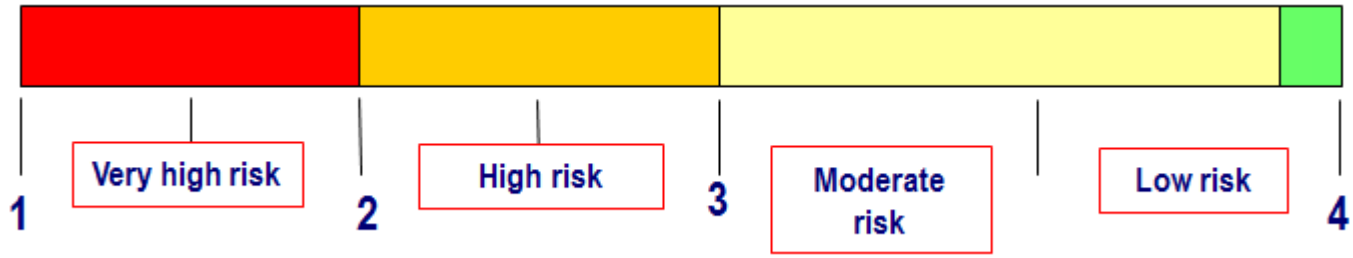


La recommandation de l'éditeur:
Des notes >3 pour un risque maîtrisé

3

Un nombre acceptable de violations dans le code source

Une application présentant des risques modérés

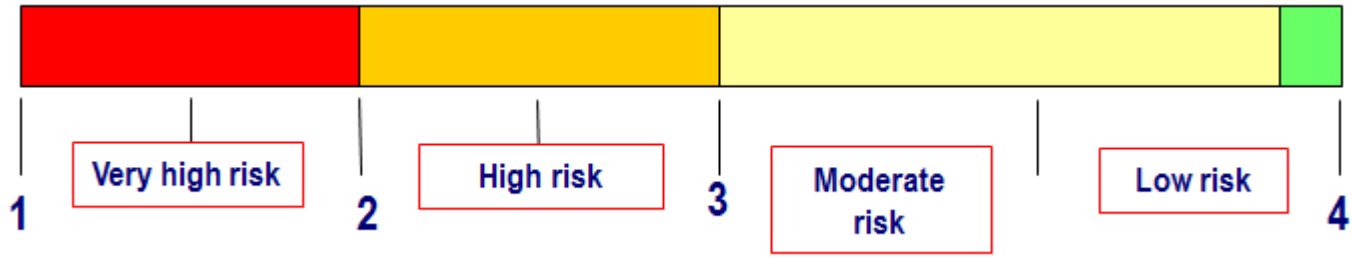


La recommandation de l'éditeur:
Des notes >3 pour un risque maîtrisé

3.5

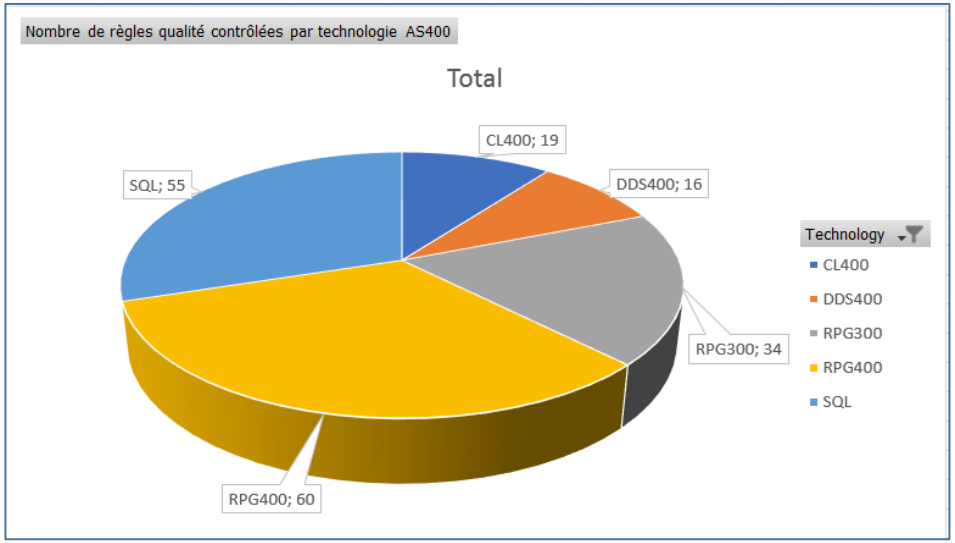
Un faible nombre de violations dans le code source

Une application présentant peu de risques



La recommandation de l'éditeur:
Des notes >3 pour un risque maîtrisé

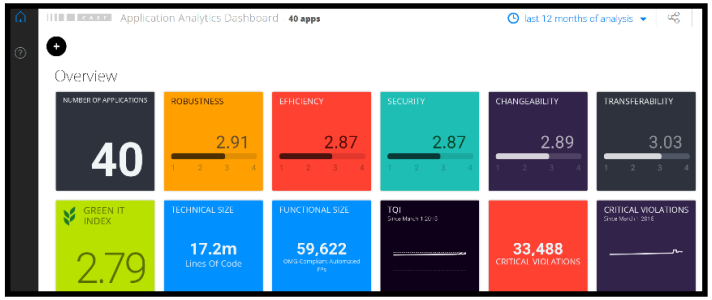
Exemples de règles RPG vérifiées



CAST vérifie 184 règles
(SQL, CL400, DDS400, RPG's)

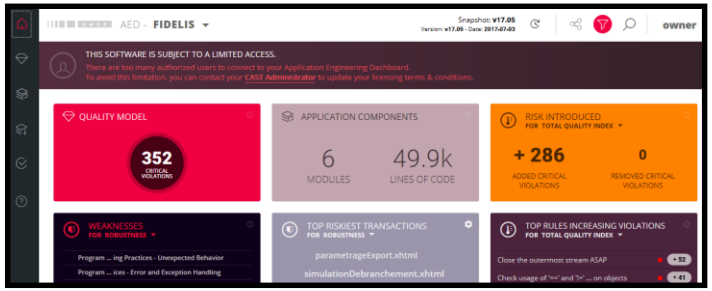
Technology	Metric name	
CL400	Avoid unreferenced CL Programs (CL400)	
CL400	Avoid use of *NOMAX (CL400)	Critique
DDS400	Avoid Logical Files using DYNSLT (DDS400)	
DDS400	Avoid Physical files with fields defined locally (DDS400)	
DDS400	Avoid unreferenced Logical file (DDS400)	
RPG300	Avoid Programs with High Cyclomatic Complexity (RPG300)	Critique
RPG300	Avoid undocumented RPG300 Programs (RPG300)	Critique
RPG300	Specify Error Subroutine for File Exception Handling (RPG300)	
RPG400	Avoid Procedures with more than X lines of code (RPG400)	
RPG400	Avoid using NOT EXISTS (RPG400)	Critique
RPG400	Avoid using plain END statement, use the explained END statement like ENDIF, ENDDO, etc (RPG400)	

Liste exhaustive des règles vérifiées



CAST HD (Health Factor Dashboard)

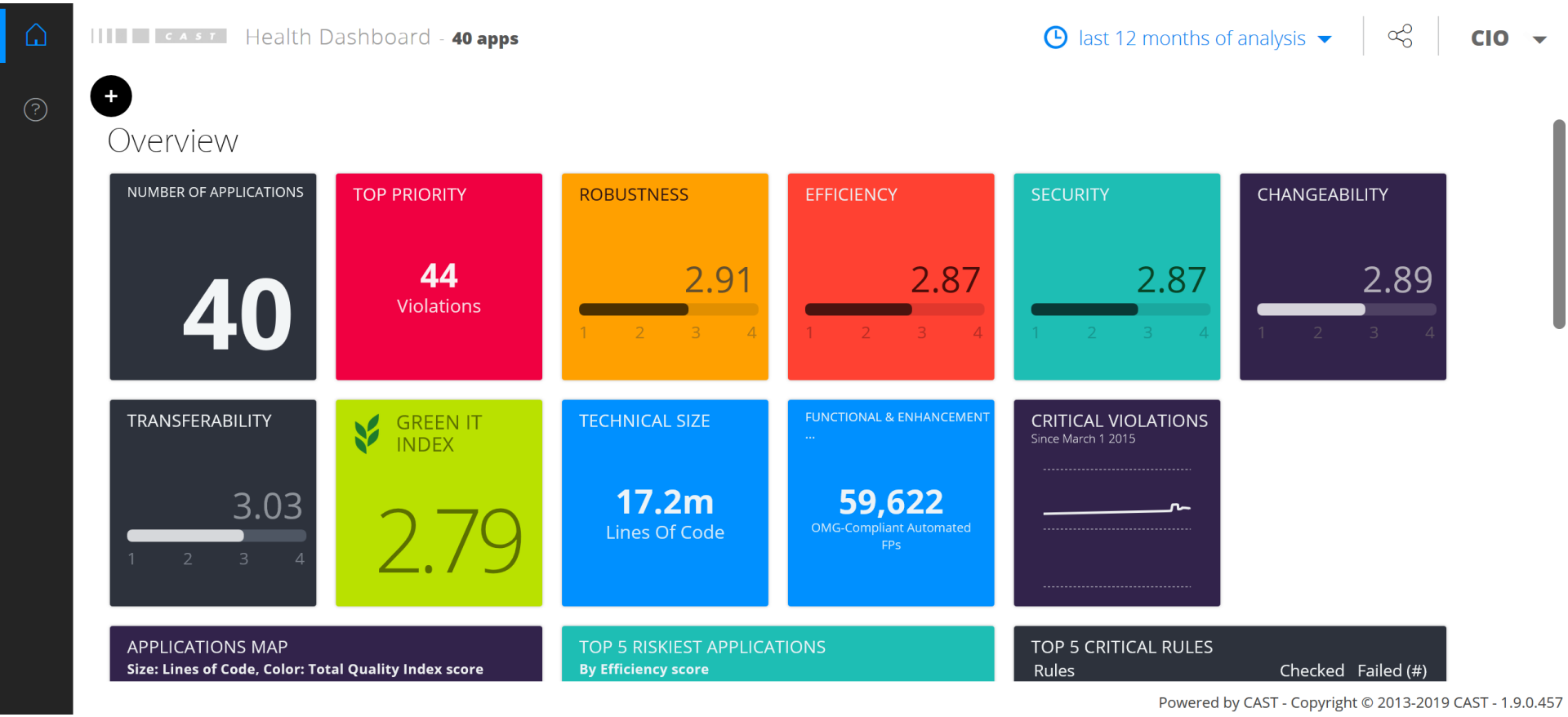
➔ Permet de donner des informations sur un ensemble d'applications (portfolio) aux chefs de projet et responsables d'applications.



CAST ED (Engineering Dashboard)

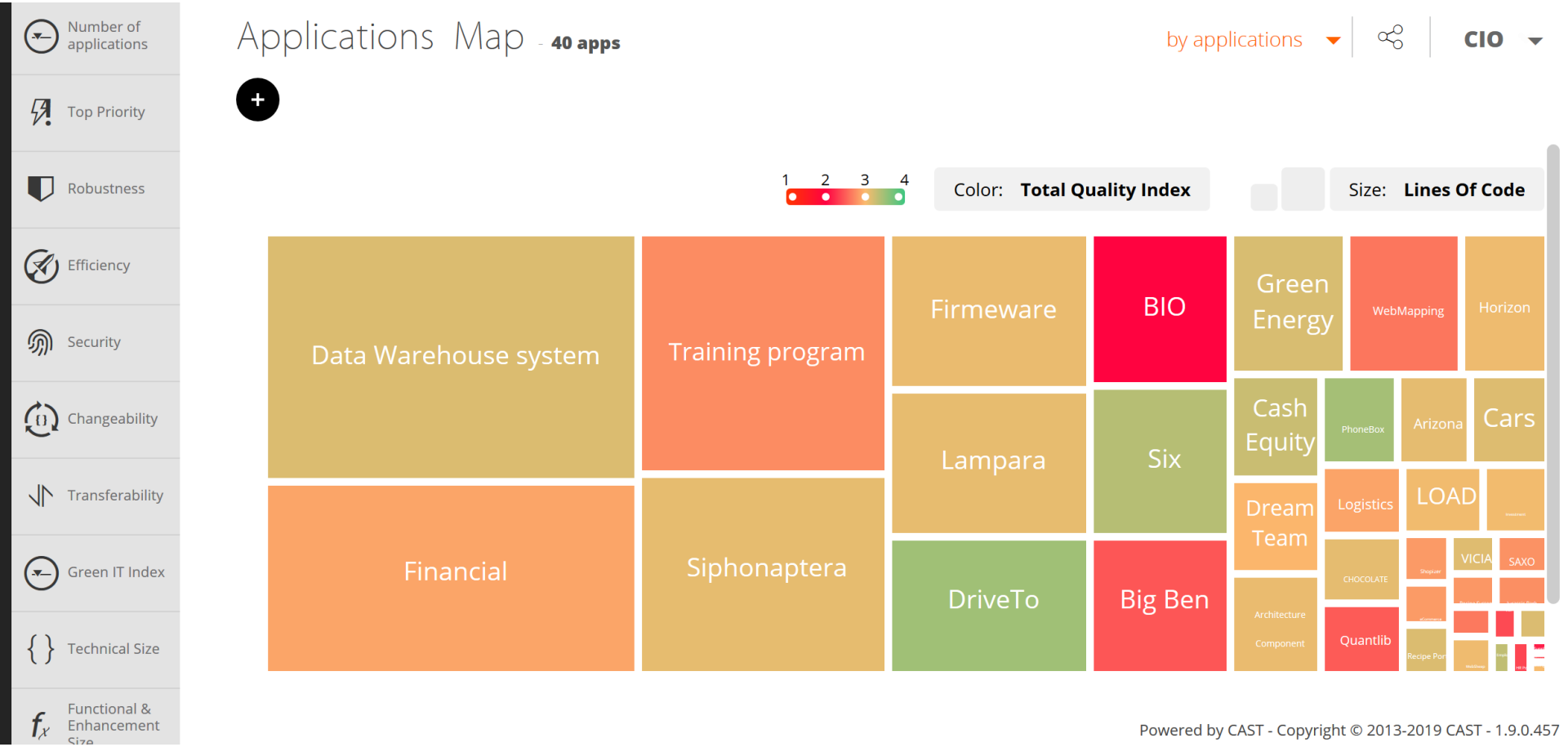
➔ Permet de naviguer dans la qualité d'une application: du facteur de santé à la ligne de code

3 CAST HD – Health Factor Dashboard - Accueil



- Permet de **restituer l'information aux gestionnaires d'applications**, chefs de projets sur la qualité de leurs applications
- Voici une synthèse d'un portfolio, ensemble d'applications
- Plusieurs niveaux d'agrégation sont possibles (exemple différentes entités)

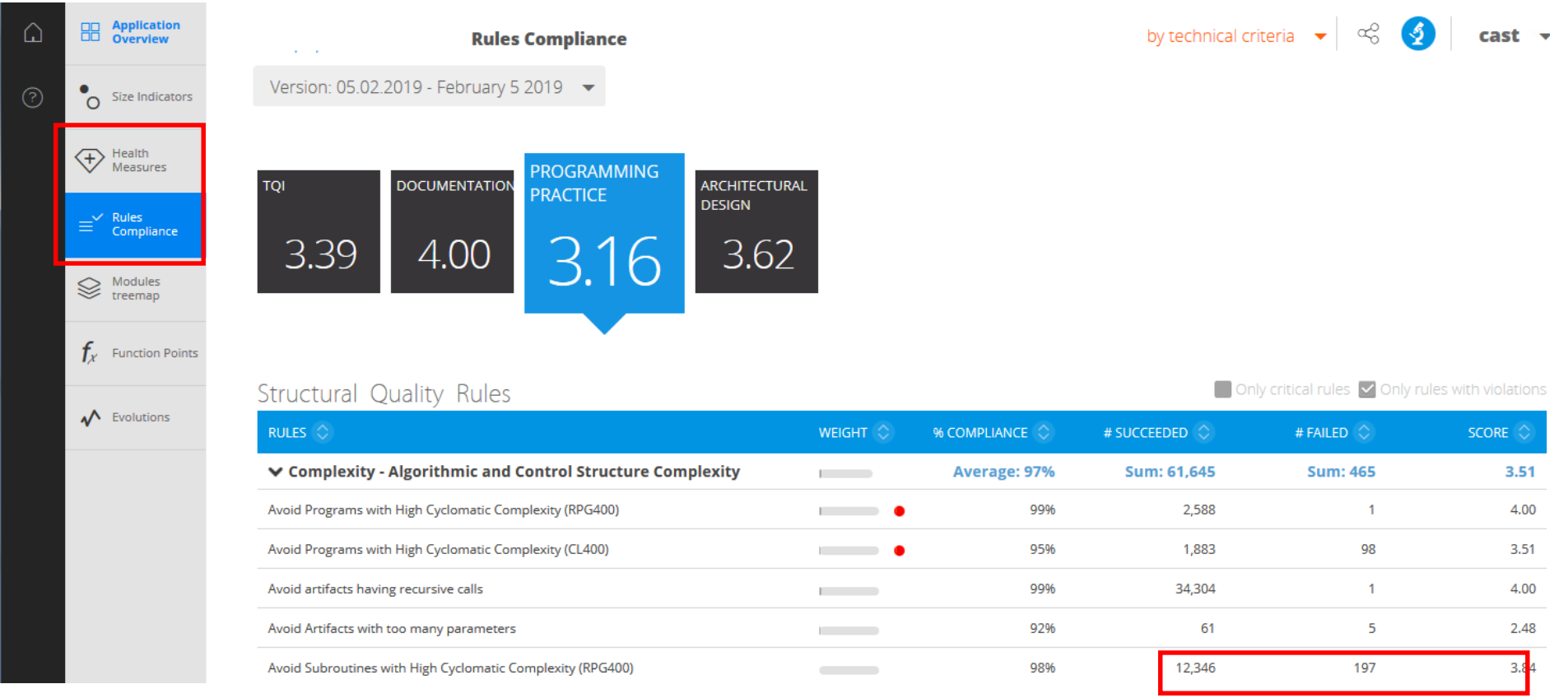
CAST HD – Health Factor Dashboard - Portfolio



QUANTITATIF
- Taille de l'application en lignes de code ou points de fonction

QUALITATIF
- Couleur indiquant le risque (fort à faible / rouge à vert)

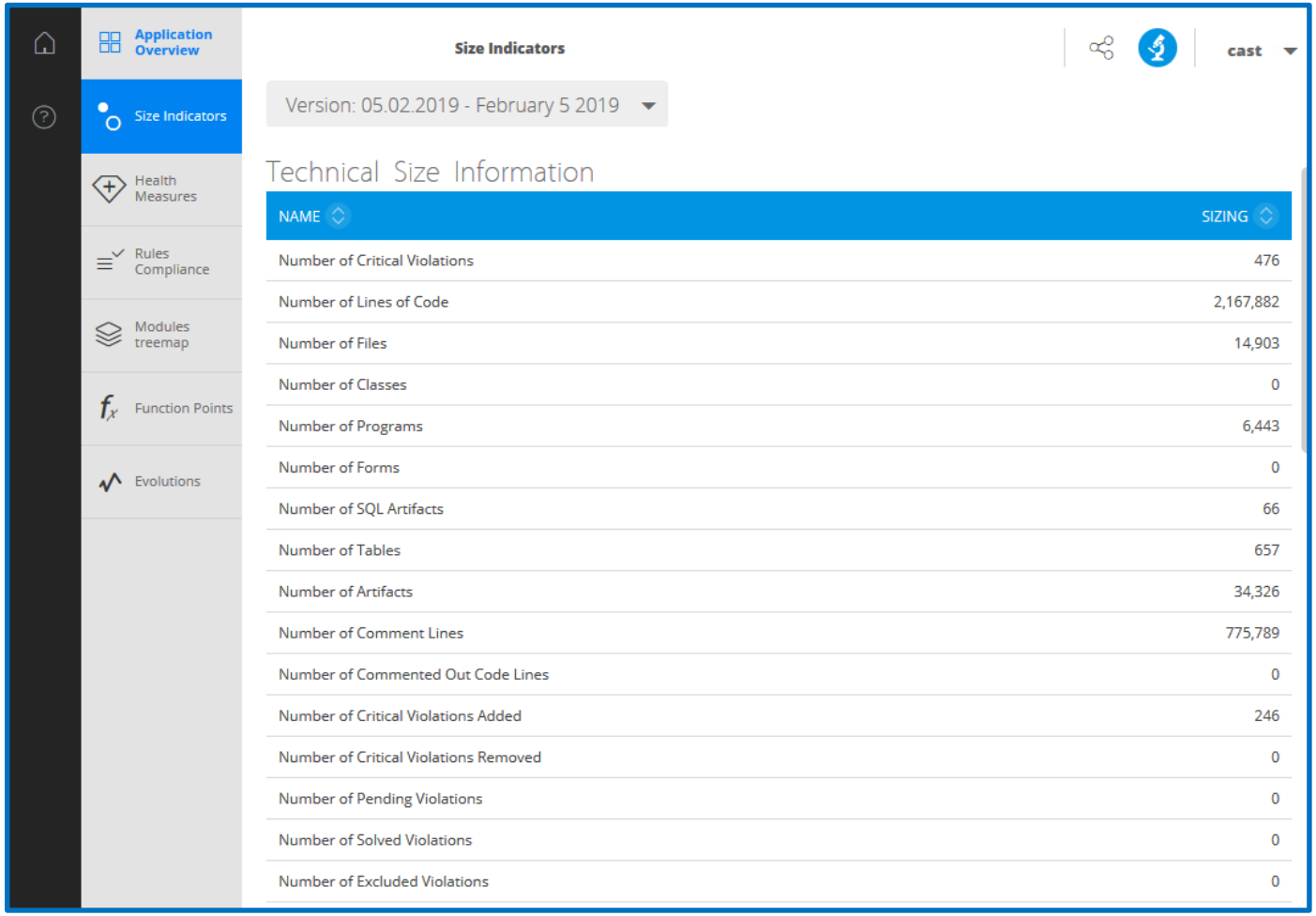
3 CAST HD – Health Factor Dashboard - En détail



ZOOM sur les Facteurs de santé, par Critère de qualité, Critère technique AFFICHAGE

- **du % de contrôles réussis,**
- **nombre de violations,**
- **Note CAST…**

CAST HD – Health Factor Dashboard - En détail



Des indicateurs permettent de suivre le dimensionnement de l'application
Exemples:

- **Nombre de lignes de code**
- **Nombre de classes, programmes**
- **Lignes commentées…**

➔AU NIVEAU GLOBAL

CAST HD – Health Factor Dashboard - En détail

Application Overview

Size Indicators

Health Measures

Rules Compliance

Modules treemap

Function Points

Evolutions

Size Indicators

cast

Version: 05.02.2019 - February 5 2019

Technology sizing by module

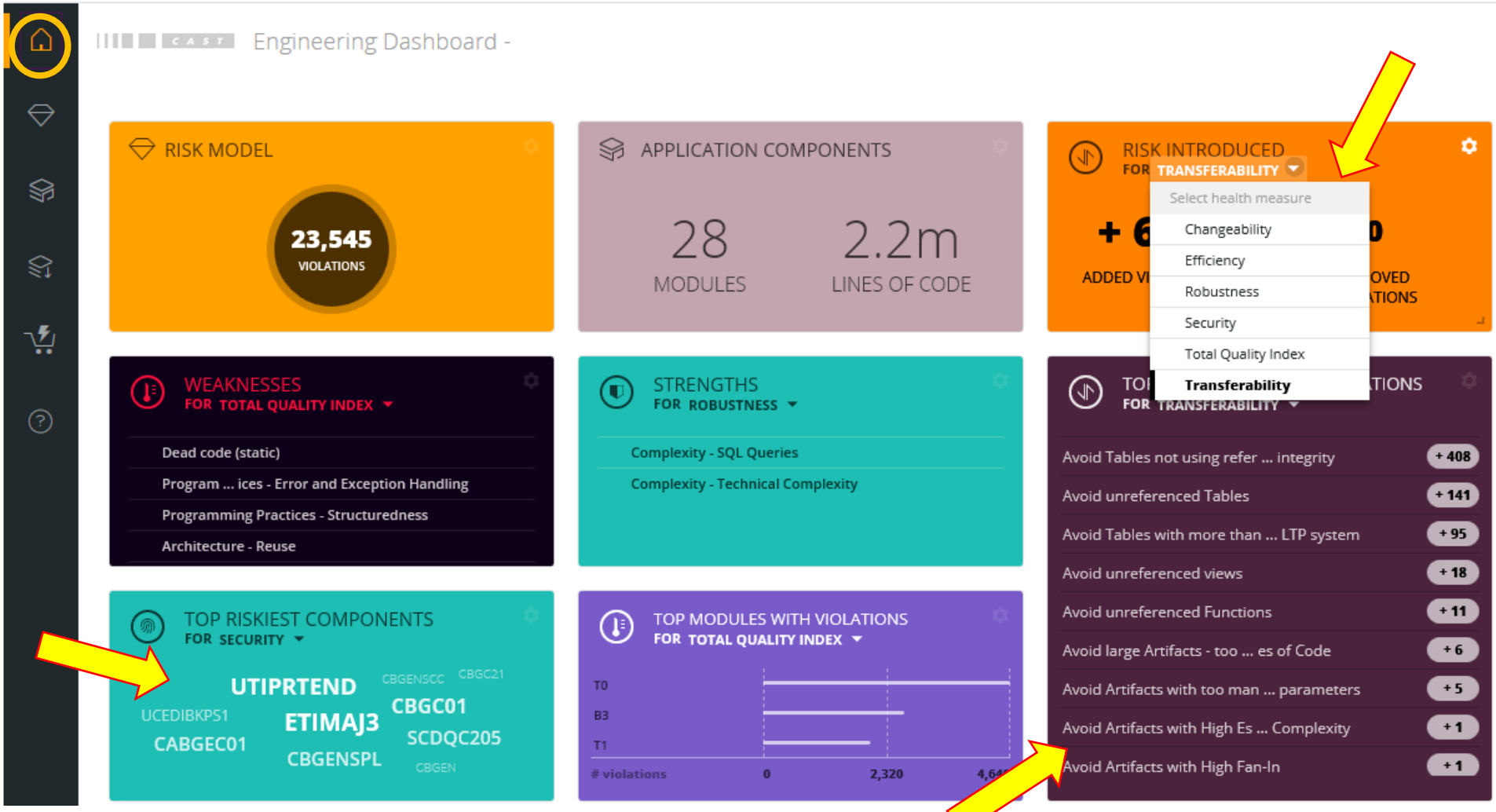
MODULE NAME	#ARTIFACTS	#LOCS	CL400	DDS400	MENU	RPG300	RPG400	SQL
A0	168	14k	503	3k	-	3k	8k	133
A1	170	15k	415	4k	-	6k	5k	180
A2	498	45k	300	10k	-	28	35k	109
A3	774	66k	3k	12k	-	16k	35k	1k
A4	2	150	-	-	-	-	150	-
A5	88	6k	430	628	-	-	5k	28
B1	519	29k	517	7k	-	6k	14k	483
B2	2k	173k	6k	48k	56	84k	32k	2k

Number of Pending Violations	0
Number of Solved Violations	0
Number of Excluded Violations	0

Des indicateurs permettent de suivre le dimensionnement de l'application
Exemples:

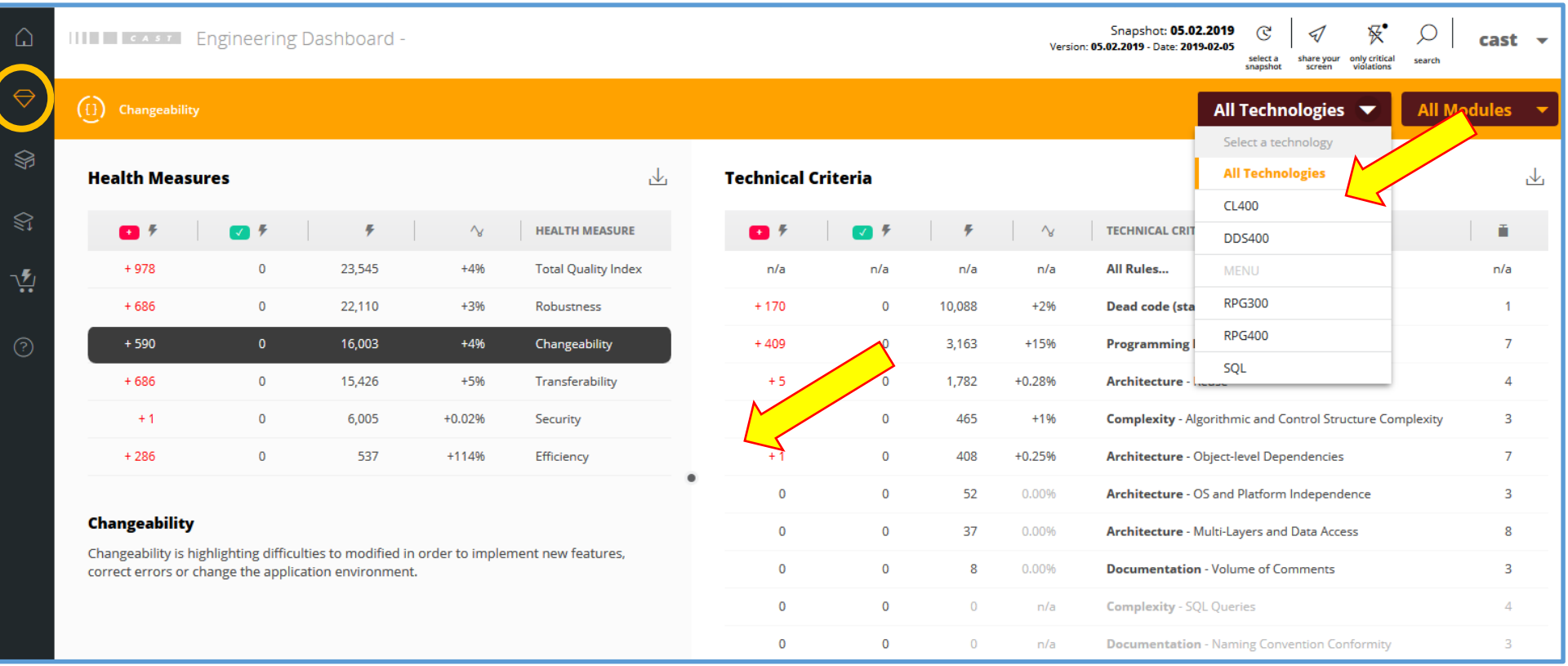
- **Nombre de lignes de code** ➔**AU NIVEAU GLOBAL**
- **Nombre de classes, programmes**
- **Lignes commentées…** ➔**AU NIVEAU MODULE**

CAST ED – Engineering Dashboard - Accueil

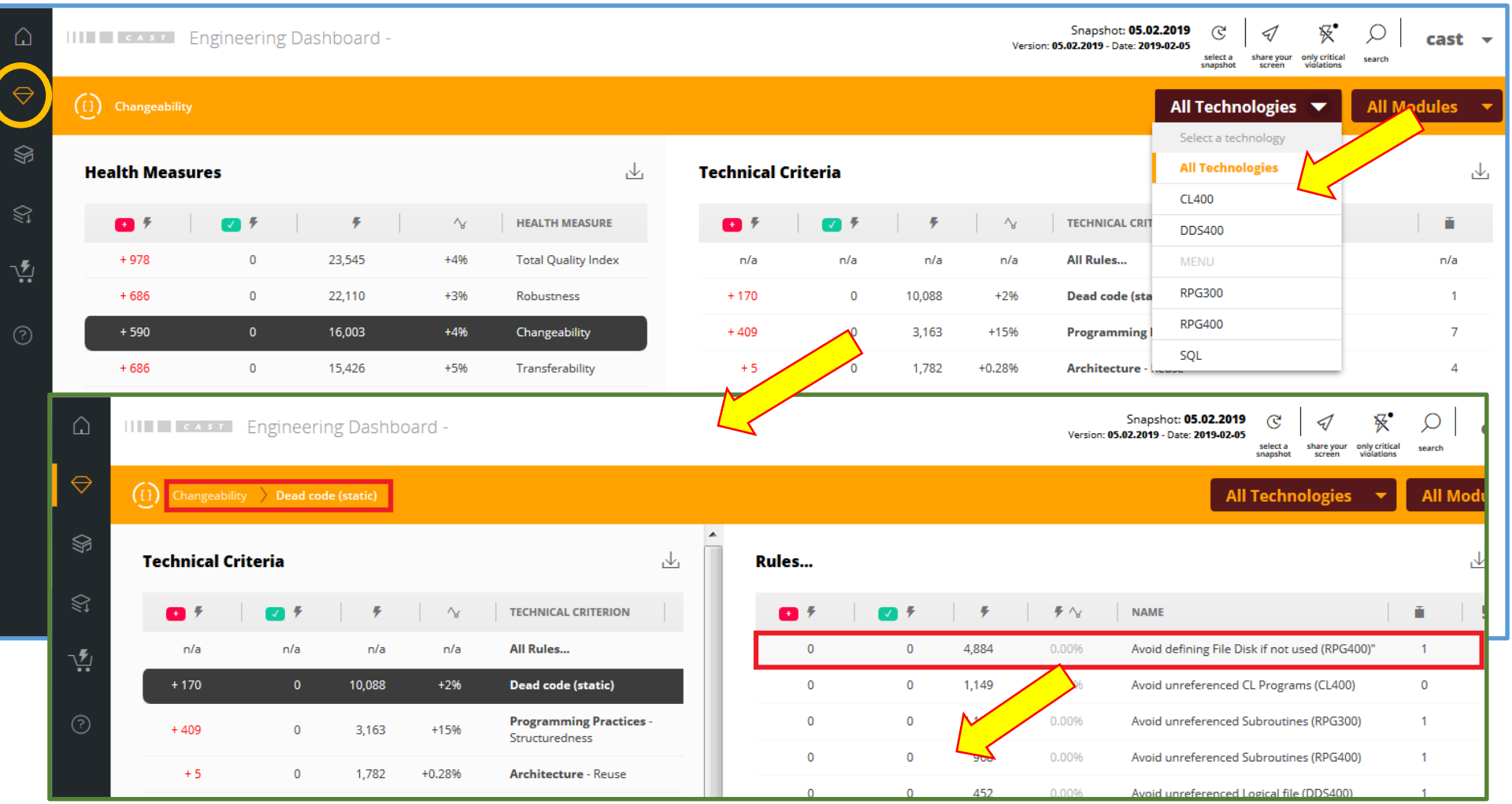


➔ Identification des composants les plus à risque
➔ Restitution des métriques par facteur de santé
➔ Customisation des écrans de synthèse
➔ Accès aux écrans de détail

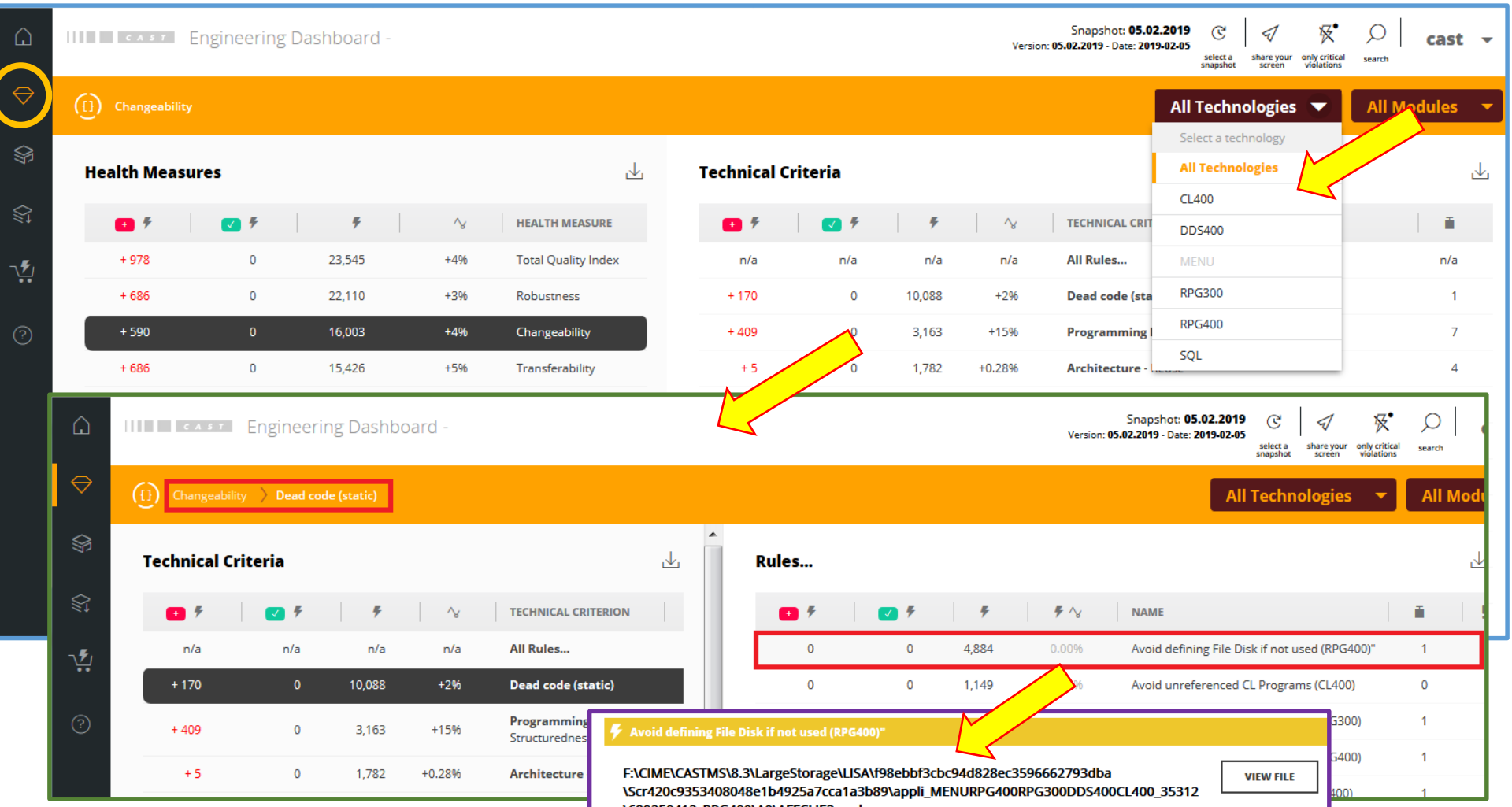
CAST ED – De la mesure à la ligne de code



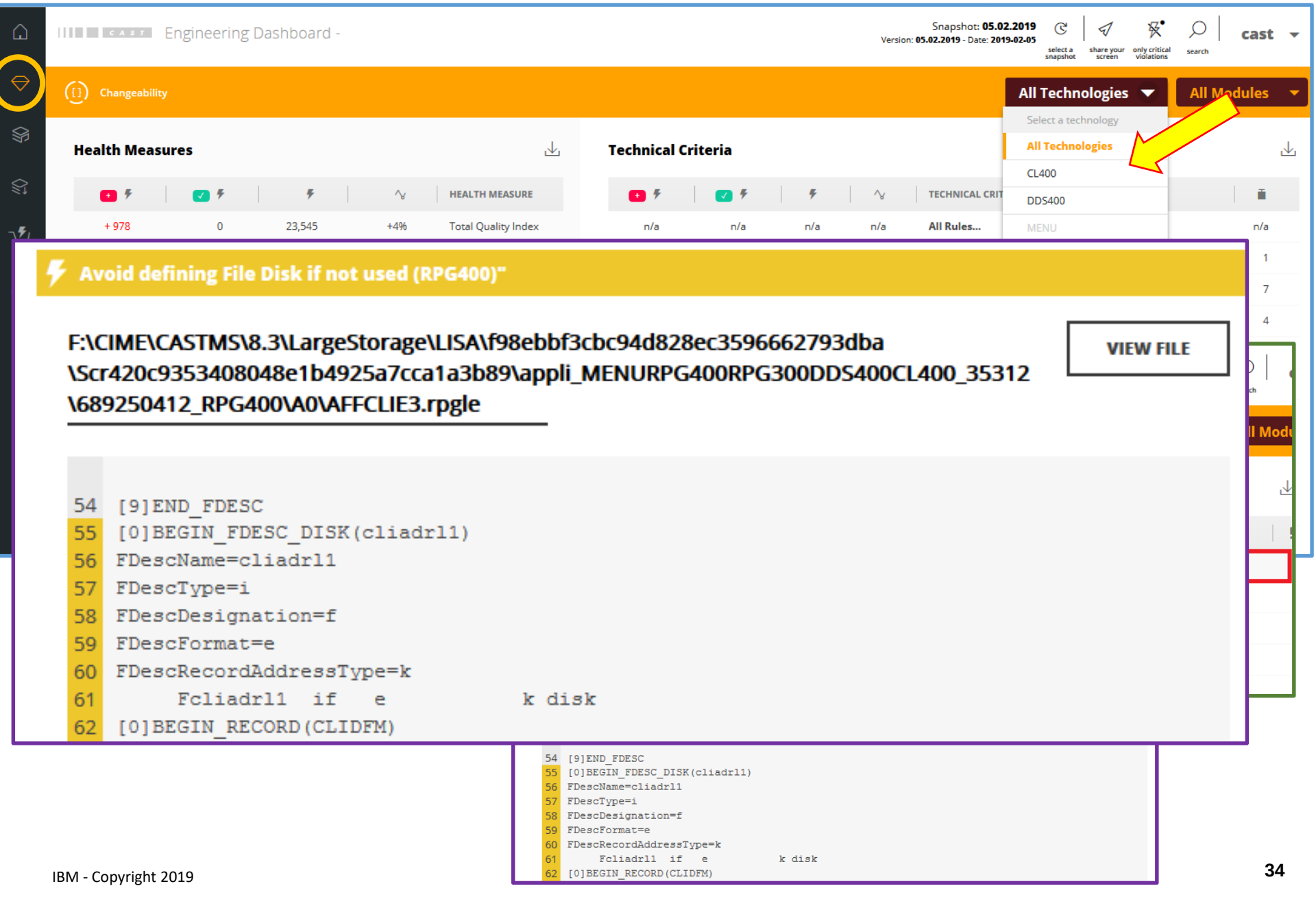
CAST ED – De la mesure à la ligne de code



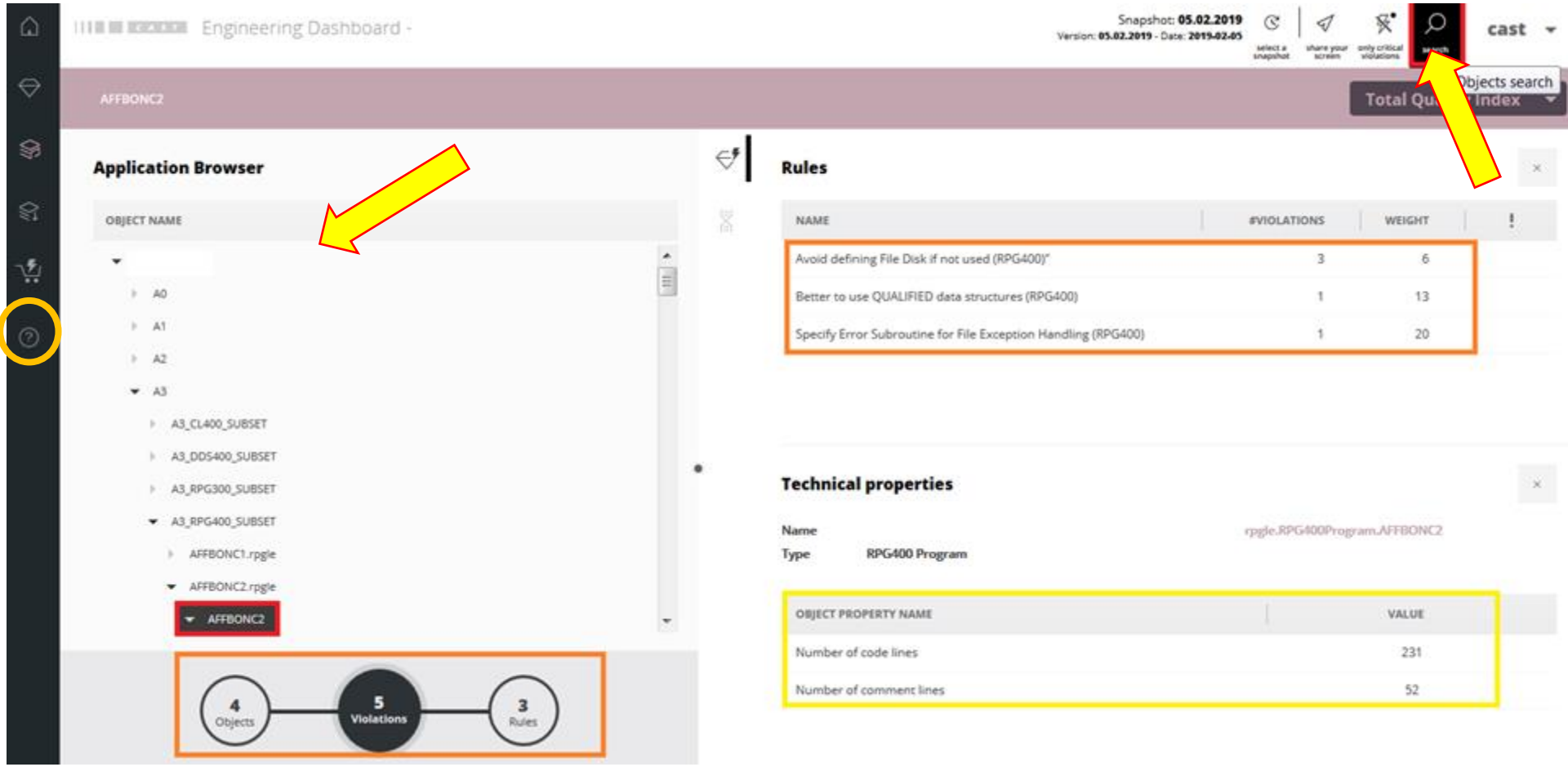
CAST ED – De la mesure à la ligne de code



3 CAST ED – De la mesure à la ligne de code



CAST ED – Engineering Dashboard - Synthèse par objet

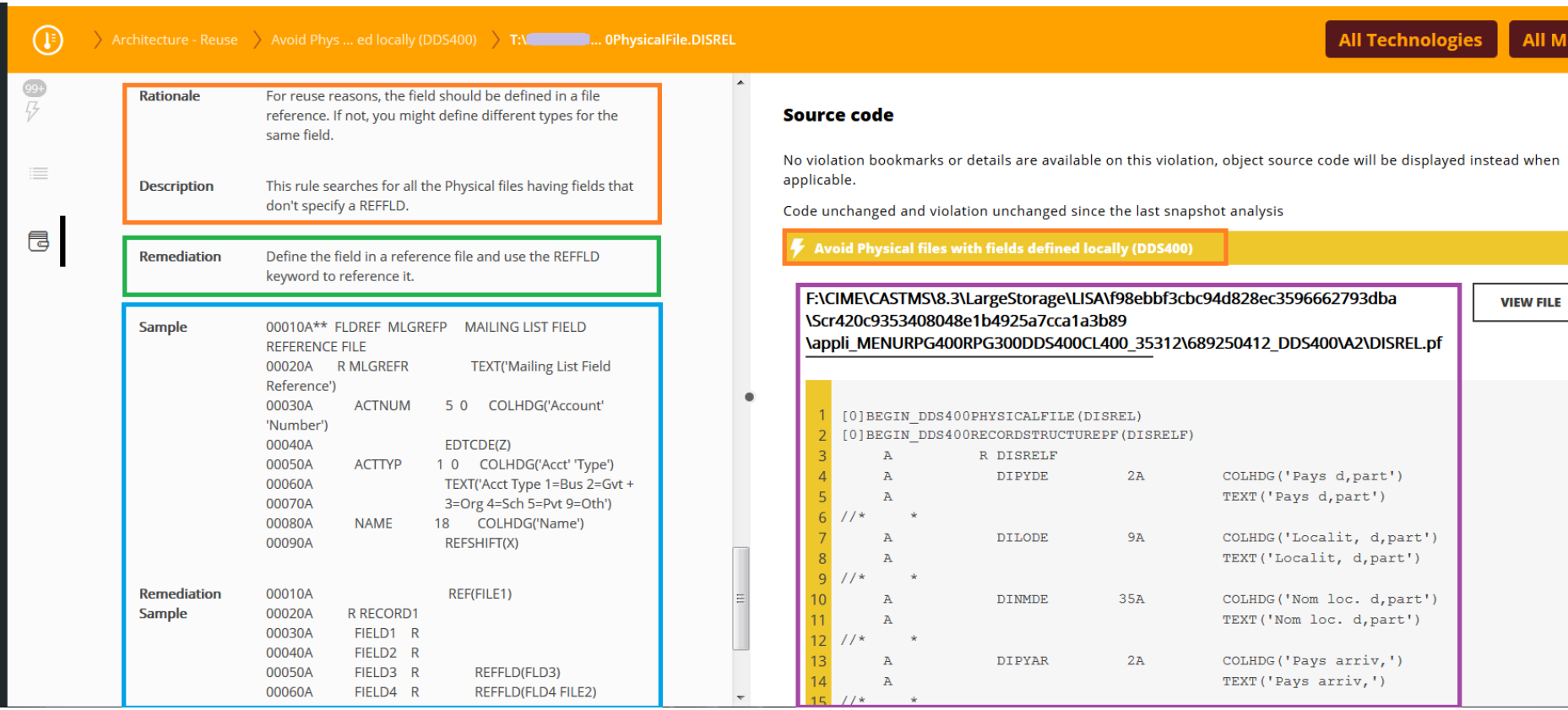


➔ Saisie de l'objet ou de la règle dans la partie recherche

➔ Navigation dans l'arborescence des objets (Programmes, ….)

➔ Affichage des règles en violation

CAST ED – Engineering Dashboard - Documentation



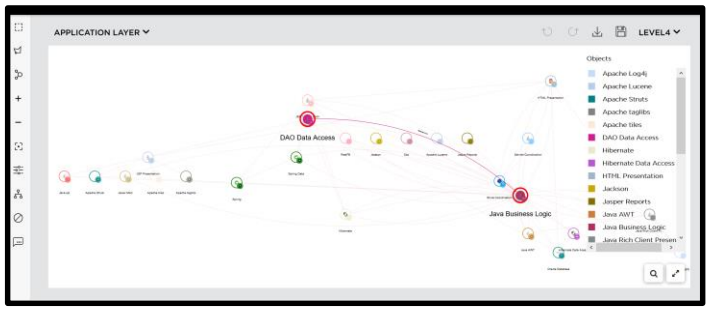
Chaque règle est consultable via les tableaux de bord CAST:

- ➔Description de la règle contrôlée
- ➔Proposition de correction
- ➔Exemple
- ➔Artifact en violation et lignes de code concernées

Les autres outils CAST (1/2)

CAST Highlight

➔ Permet d'analyser tout un ensemble d'applications (portfolio) dans une période de temps raisonnable. Cela permet de fournir une bonne estimation de la qualité d'ensemble de ce portfolio quitte à utiliser CAST AIP pour compléter sur un sous-ensemble d'applications.



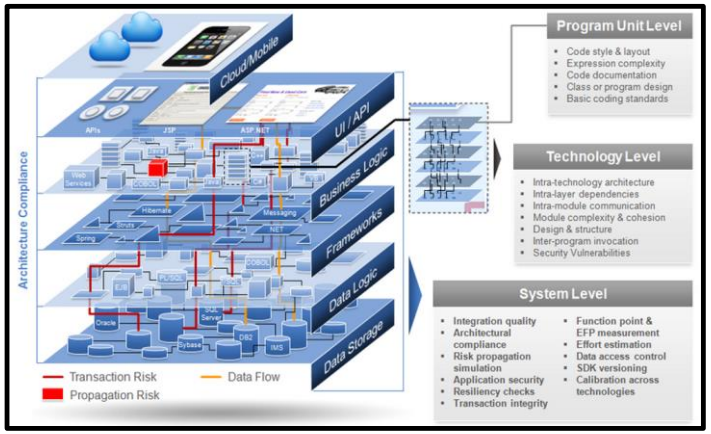
CAST Imaging

➔ Permet de naviguer au cœur de l'application. Cet outil permet d'explorer de façon dynamique les différentes couches de l'architecture de l'application. C'est le dernier né de la famille;)

Les autres outils CAST (2/2)

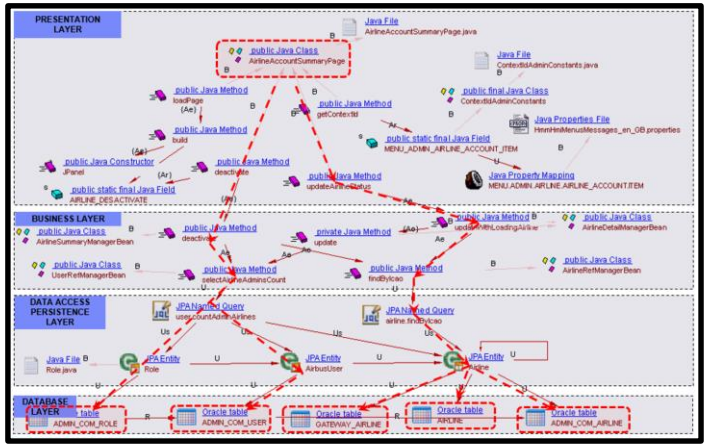
Action Plan Optimizer

➔ Permet de faire une simulation des notes obtenues dans le cadre d'un plan d'action.



CAST Architecture Checker

➔ Permet d'afficher les transactions CAST d'un écran à un élément de persistance.



CAST Enlighten

➔ Permet une **navigation interactive pour simuler les impacts** de la modification d'un objet et fournir une **documentation automatisée** sous la forme d'un rapport HTML permettant d'alimenter une documentation technique.

Agenda

1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils

Qui sommes-nous? Quelles sont nos références?

LES DATES CLES

2003 Premiers travaux chez IBM France sur la mesure de la qualité logicielle.
2008 Création du centre SQ / AFP CoE avec un focus initial sur la Qualité Logicielle.
2012 Ajout de l'approche de mesure de la taille logicielle (AFP).
2013 Premiers calculs AFP sur un large périmètre
2014 Extension des ressources avec l'Inde
2015 Devient un centre mondial
2016 Extension avec des ressources au Maroc
2017 Extension avec des ressources Lilloises
2019 Création d'une extension CAST permettant d'évaluer les applications Datastage.

PROJETS EXTERNES

- Consulting / Gouvernance
- Audit unique
- Centre de test / Mise en place de "Quality Gate"
- CoE set up and operations pour des comptes externes

PROJETS INTERNES

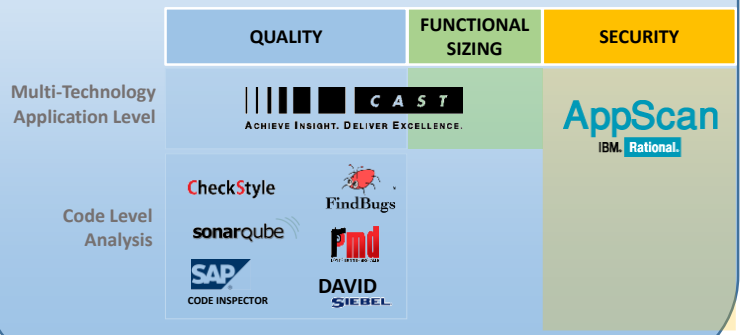
- Audit / Due Diligence
- Usine Logicielle / (Build, Transition,…)
- Test Center

SERVICES

Un catalogue de services avec des unités d'oeuvre:
- Mise en œuvre d'une approche holistique
- Production de rapports à partir de l'exécution d'analyses
- Résultats de recherche et d'interprétation
- Plan de remédiation
- Conseil / Coaching

PRINCIPAUX OUTILS

Notre approche a besoin et se repose sur des outils qui réalisent de l'analyse de code:



LOCALISATION

Qui sommes-nous? Quelles sont nos références?

Qualité Logicielle

Client : Entité BNPP dédiée aux prêts à la consommation

Réalisation : Implémentation interne (Programme AD/M) d'un plan d'amélioration de la qualité en utilisant CAST

Points de fonction automatisés

Client: Leader aéronautique européen

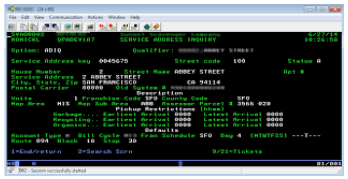
Réalisation: Mesures de Qualité et de Productivité pour réaliser des gains avec un système d'incitation et pénalités.

Agenda

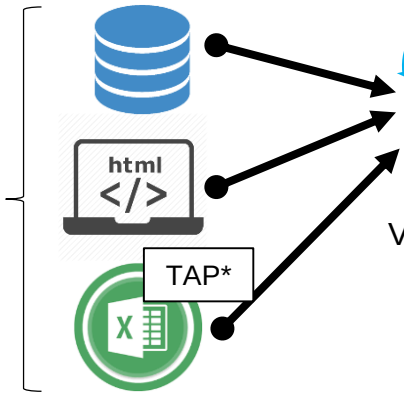
1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils

La procédure d'analyse

APPLICATION



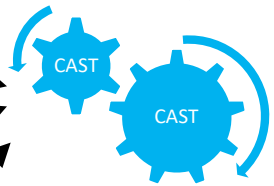
ENTRANTS DE L'ANALYSE



Mise en œuvre

Exécution

Production
De rapports



Vérification des pré-requis

Rapport technique
(SQuID)

Livrable IBM

Vue transition

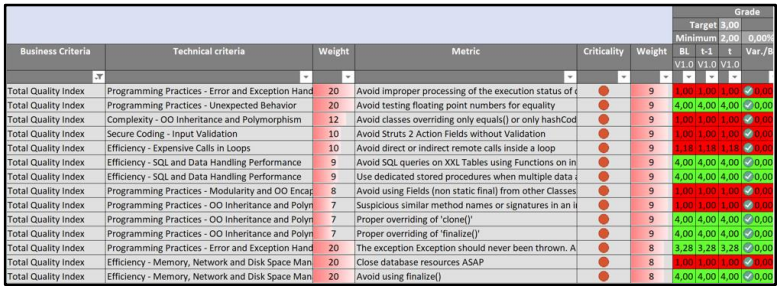
Rapport générique
PPT

Livrable IBM

Livrable IBM

* TAP: Technical Application Profile

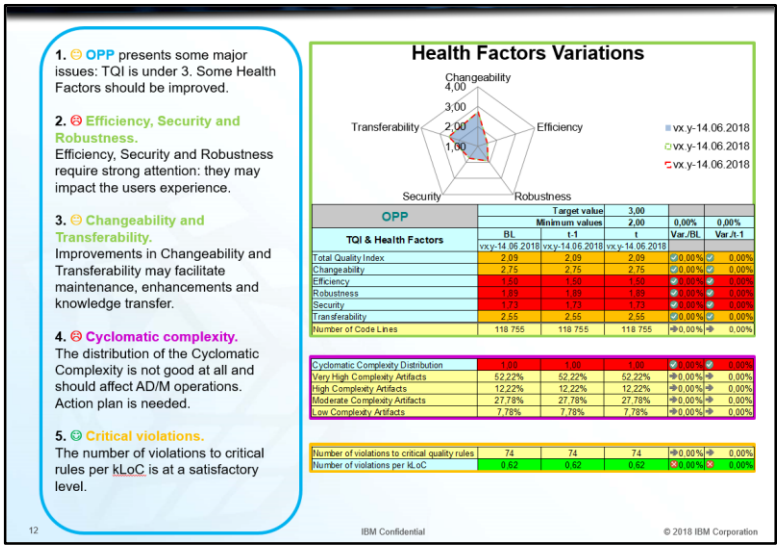
Les outils de notre centre SQ/AFP CoE (1/2)



Business Criteria	Technical criteria	Weight	Metric	Criticality	Weight	Grade BL V1.0	Grade t-1 V1.0	Grade t V1.0	Var./B
						Target 3,00			
						Minimum 2,00			0,00%
Total Quality Index	Programming Practices - Error and Exception Hand	20	Avoid improper processing of the execution status of		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - Unexpected Behavior	20	Avoid testing floating point numbers for equality		9	4,00	4,00	4,00	0,00
Total Quality Index	Complexity - OO Inheritance and Polymorphism	12	Avoid classes overriding only equals() or only hashCod		9	1,00	1,00	1,00	0,00
Total Quality Index	Secure Coding - Input Validation	10	Avoid Struts 2 Action Fields without Validation		9	1,00	1,00	1,00	0,00
Total Quality Index	Efficiency - Expensive Calls in Loops	10	Avoid direct or indirect remote calls inside a loop		9	1,18	1,18	1,18	0,00
Total Quality Index	Efficiency - SQL and Data Handling Performance	9	Avoid SQL queries on XXL Tables using Functions on in		9	4,00	4,00	4,00	0,00
Total Quality Index	Efficiency - SQL and Data Handling Performance	9	Use dedicated stored procedures when multiple data		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - Modularity and OO Encap	8	Avoid using Fields (non static final) from other Classes		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Suspicious similar method names or signatures in an i		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Proper overriding of 'clone()'		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Proper overriding of 'finalize()'		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - Error and Exception Hand	20	The exception Exception should never been thrown. A		8	3,28	3,28	3,28	0,00
Total Quality Index	Efficiency - Memory, Network and Disk Space Man	20	Close database resources ASAP		8	1,00	1,00	1,00	0,00
Total Quality Index	Efficiency - Memory, Network and Disk Space Man	20	Avoid using finalize()		8	4,00	4,00	4,00	0,00

SQuID (Software Quality Insight Dashboard)

➔Un livrable sous Excel qui permet de synthétiser l'ensemble de l'information sur la qualité logicielle. C'est un rapport technique généré par l'équipe d'analyse et utilise par l'équipe d'interprétation.

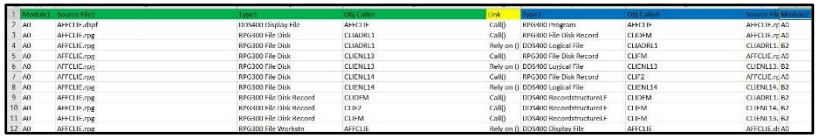


Generic Report

➔ Un livrable au format PowerPoint base sur le SQuID, plus lisible et compréhensible que le SQuID.

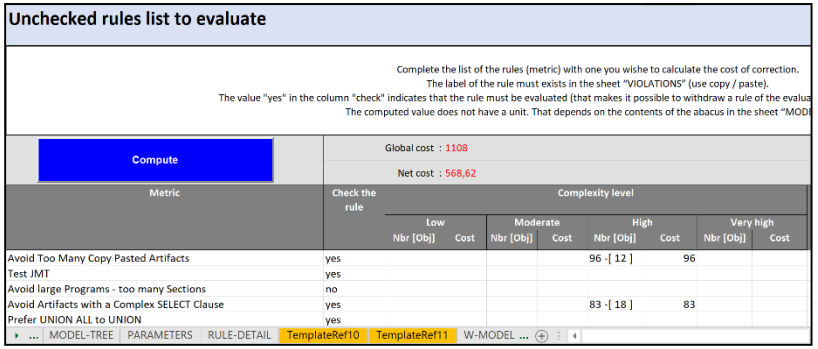
Ces outils, basés sur notre expérience, viennent en complément des outils CAST standards et font partie de la panoplie d'outils que nous proposons dans le cadre de nos offres.

Les outils de notre centre SQ/AFP CoE (2/2)



Object Link Generator

➔Livrable Excel permettant de synthétiser toutes les informations des objets en utilisant les liens stockés dans CAST. Au moyen des filtres il est possible de voir tous les appelants / appelés en partant d'un objet spécifique.



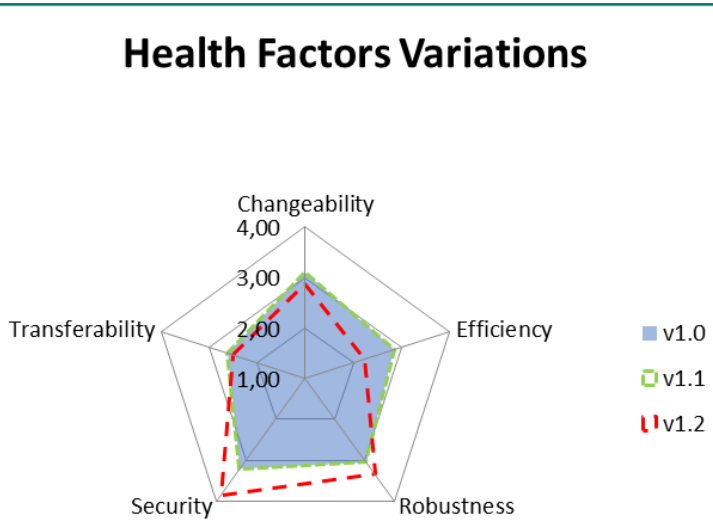
Complexity / Estimation asset

➔Livrable Excel permettant d'accéder à la complexité des composants macro.
Cet outil permet de faire des estimations des coûts de correction sur les règles en violation.

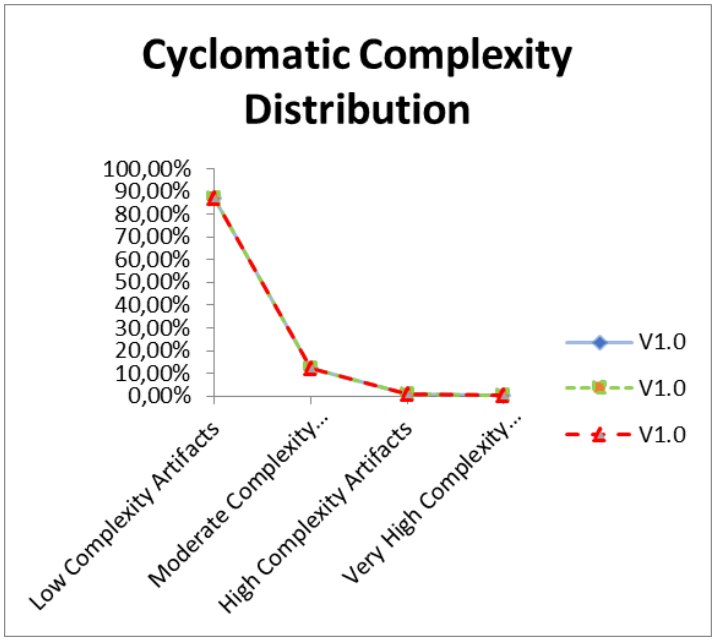
Ces outils, basés sur notre expérience, viennent en complément des outils CAST standards et font partie de la panoplie d'outils que nous proposons dans le cadre de nos offres.

5 SQuID (Software Quality Insight Dashboard) 1/2

IBM L&K	Target value		3,00		
	Minimum values		2,00	0,00%	0,00%
TQI & Health Factors	BL	t-1	t	Var./BL	Var./t-1
	v1.0	v1.1	v1.2		
Total Quality Index	2,99	2,99	2,97	-0,53%	-0,51%
Changeability	3,10	3,10	2,87	-7,48%	-7,43%
Efficiency	2,85	2,85	2,22	-21,97%	-22,00%
Robustness	3,04	3,04	3,34	9,85%	9,90%
Security	3,21	3,21	3,84	19,66%	19,64%
Transferability	2,62	2,62	2,51	-4,28%	-4,26%



Synthèse des principales informations



Number of Code Lines	3 540	3 540	3 540	0,00%	0,00%
Automated IFPUG FP Estimation	29	29	29	0,00%	0,00%
Number of LoC per Function Point	122,07	122,07	122,07	0,00%	0,00%
Number of violations to critical quality rules	75	75	75	0,00%	0,00%
Number of violations per kLoC	21,19	21,19	21,19	0,00%	0,00%
Cyclomatic Complexity Distribution	3,81	3,81	3,81	0,00%	0,00%
Very High Complexity Artifacts	0,38%	0,38%	0,38%	0,00%	0,00%
High Complexity Artifacts	0,75%	0,75%	0,75%	0,00%	0,00%
Moderate Complexity Artifacts	12,08%	12,08%	12,08%	0,00%	0,00%
Low Complexity Artifacts	86,79%	86,79%	86,79%	0,00%	0,00%

- **Agrège** la plupart des **informations relatives à la qualité logicielle** présente dans CAST
- Permet la **comparaison de 3 versions** d'une application

5 SQuID (Software Quality Insight Dashboard) 2/2

						Grade			
						Target 3,00			
						Minimum 2,00			0,00%
Business Criteria	Technical criteria	Weight	Metric	Criticality	Weight	BL V1.0	t-1 V1.0	t V1.0	Var./B
Total Quality Index	Programming Practices - Error and Exception Hand	20	Avoid improper processing of the execution status of		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - Unexpected Behavior	20	Avoid testing floating point numbers for equality		9	4,00	4,00	4,00	0,00
Total Quality Index	Complexity - OO Inheritance and Polymorphism	12	Avoid classes overriding only equals() or only hashCod		9	1,00	1,00	1,00	0,00
Total Quality Index	Secure Coding - Input Validation	10	Avoid Struts 2 Action Fields without Validation		9	1,00	1,00	1,00	0,00
Total Quality Index	Efficiency - Expensive Calls in Loops	10	Avoid direct or indirect remote calls inside a loop		9	1,18	1,18	1,18	0,00
Total Quality Index	Efficiency - SQL and Data Handling Performance	9	Avoid SQL queries on XXL Tables using Functions on in		9	4,00	4,00	4,00	0,00
Total Quality Index	Efficiency - SQL and Data Handling Performance	9	Use dedicated stored procedures when multiple data a		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - Modularity and OO Encap	8	Avoid using Fields (non static final) from other Classes		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Suspicious similar method names or signatures in an i		9	1,00	1,00	1,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Proper overriding of 'clone()'		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - OO Inheritance and Polyn	7	Proper overriding of 'finalize()'		9	4,00	4,00	4,00	0,00
Total Quality Index	Programming Practices - Error and Exception Hand	20	The exception Exception should never been thrown. A		8	3,28	3,28	3,28	0,00
Total Quality Index	Efficiency - Memory, Network and Disk Space Man	20	Close database resources ASAP		8	1,00	1,00	1,00	0,00
Total Quality Index	Efficiency - Memory, Network and Disk Space Man	20	Avoid using finalize()		8	4,00	4,00	4,00	0,00

LIVRABLE « OFFLINE »

AGGREGATEUR

FORMAT « CONNU »

ACCESSIBLE

- Décrit toutes les métriques de l'application

- Classées par criticité, poids, Facteur de santé

Generic Report (par Slide Generator)

1. 😐 **BlueMainFrame** presents some major issues: TQI is under 3. Some Health Factors should be improved.

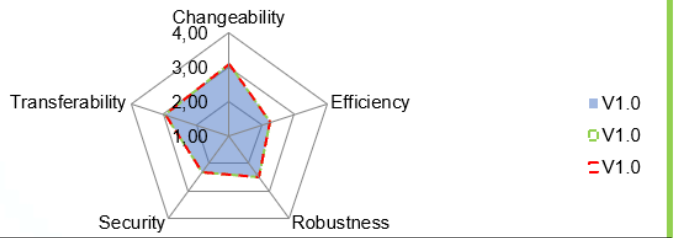
2. 😐 **Efficiency, Security and Robustness.**
Efficiency, Security and Robustness should be improved: they may impact the users experience.

3. 😊 **Changeability and Transferability.**
Improvements in Transferability may facilitate knowledge transfer. Changeability has reached a satisfactory level.

4. 😐 **Cyclomatic complexity.**
The distribution of the Cyclomatic Complexity is average and should affect AD/M operations, some actions should be taken.

5. 😊 **Critical violations.**
The number of violations to critical rules per kLoC is at a satisfactory level.

Health Factors Variations



BlueMainFrame	Target value		3,00		
	linim um values		2,00	0,00%	0,00%
TQI & Health Factors	BL	t-1	t	Var./BL	Var./t-1
	V1.0	V1.0	V1.0		
Total Quality Index	2,62	2,62	2,62	0,00%	0,00%
Changeability	3,07	3,07	3,07	0,00%	0,00%
Efficiency	2,27	2,27	2,27	0,00%	0,00%
Robustness	2,52	2,52	2,52	0,00%	0,00%
Security	2,34	2,34	2,34	0,00%	0,00%
Transferability	2,94	2,94	2,94	0,00%	0,00%
Number of Code Lines	24 930	24 930	24 930	0,00%	0,00%
Cyclomatic Complexity Distribution	2,25	2,25	2,25	0,00%	0,00%
Very High Complexity Artifacts	3,51%	3,51%	3,51%	0,00%	0,00%
High Complexity Artifacts	3,51%	3,51%	3,51%	0,00%	0,00%
Moderate Complexity Artifacts	39,18%	39,18%	39,18%	0,00%	0,00%
Low Complexity Artifacts	53,80%	53,80%	53,80%	0,00%	0,00%
Number of violations to critical quality rules	33	33	33	0,00%	0,00%
Number of violations per kLoC	1,32	1,32	1,32	0,00%	0,00%

7 IBM Confidential

LIVRABLE « OFFLINE »

COMMUNICATION

FORMAT « CONNU »

VALEUR AJOUTEE

Deux exemples de mode de restitution:
-« **Transition** » pour mettre en évidence la qualité **avant la reprise d'actif**

-« **Run** » en outil quotidien (mensuel) au sein d'une maintenance

Object Link Generator

	Module1	Source File1	Type1	Obj Caller	Link	Type2	Obj Called	Source File	Module2
2	A0	AFFCLIE.dspf	DDS400 Display File	AFFCLIE	Call()	RPG300 Program	AFFCLIE	AFFCLIE.rp	A0
3	A0	AFFCLIE.rpg	RPG300 File Disk	CLIADRL1	Call()	RPG300 File Disk Record	CLIDFM	AFFCLIE.rp	A0
4	A0	AFFCLIE.rpg	RPG300 File Disk	CLIADRL1	Rely on ()	DDS400 Logical File	CLIADRL1	CLIADRL1.	B2
5	A0	AFFCLIE.rpg	RPG300 File Disk	CLIENL13	Call()	RPG300 File Disk Record	CLIFM	AFFCLIE.rp	A0
6	A0	AFFCLIE.rpg	RPG300 File Disk	CLIENL13	Rely on ()	DDS400 Logical File	CLIENL13	CLIENL13.	B2
7	A0	AFFCLIE.rpg	RPG300 File Disk	CLIENL14	Call()	RPG300 File Disk Record	CLIF2	AFFCLIE.rp	A0
8	A0	AFFCLIE.rpg	RPG300 File Disk	CLIENL14	Rely on ()	DDS400 Logical File	CLIENL14	CLIENL14.	B2
9	A0	AFFCLIE.rpg	RPG300 File Disk Record	CLIDFM	Call()	DDS400 RecordstructureLF	CLIDFM	CLIADRL1.	B2
10	A0	AFFCLIE.rpg	RPG300 File Disk Record	CLIF2	Call()	DDS400 RecordstructureLF	CLIFM	CLIENL14.	B2
11	A0	AFFCLIE.rpg	RPG300 File Disk Record	CLIFM	Call()	DDS400 RecordstructureLF	CLIFM	CLIENL13.	B2
12	A0	AFFCLIE.rpg	RPG300 File Workstn	AFFCLIE	Rely on ()	DDS400 Display File	AFFCLIE	AFFCLIE.ds	A0

Objets Appelants

Nature
des liens entre l'appelant et l'appelé

Objets Appelés

LIVRABLE « OFFLINE »

APPELS INTER MODULES

FORMAT « CONNU »

VALEUR AJOUTEE

L'utilisation des filtres permet de voir tous les appelants / appelés à partir d'un objet spécifique

Complexity / Estimation asset

Add CAST action plan selected rules

Filed the list with all violated rules

Filed the list with critical violated rules

Global Cost (Day): 207,15 Global Cost (Currency): 105952,5 $

Net Cost (Day): 207,15 Global Cost (Currency): 105 952,50 $

Evaluation bases in team metrics

Quick evaluation based on CAST metrics

Metric	Technology	Check the rule	Complexity level								Cost Brut	Sharing Ratio (%)		Comments
			Low		Moderate		High		V High					
			Nbr [Obj]	Cost	Nbr [Obj]	Cost	Nbr [Obj]	Cost	Nbr [Obj]	Cost				
Avoid Fields in Se…let Classes that are not …al static	JEE	yes	7 -[7]	0,21							0,21	100	21	Hourly cost: 500 $
Avoid Log forging …lnerabilities (CWE-117)	JEE	yes			2 -[2]	0,14	1 -[1]	0,35			0,49	100	19	Hourly cost: 500 $
Avoid Procedures …ing an Insert, Update, Del… Create Table or S	Microsoft …	…s	3 -[3]	0,15	9 -[9]	0,9			4 -[4]	4,8	5,85	100	6	Hourly cost: 650 $
Avoid Tables with… Primary Key	Microsoft T…	…	46 -[46]	2,3							2,3	100		Hourly cost: 650 $
Avoid classes over…g only equals() or only has…de()	JEE	ye…	11 -[11]	0,33							0,33	100		Hourly cost: 500 $
Avoid cyclical calls …inheritances between pack…	JEE	yes	140 -[140	4,2							4,2	100		…lourly cost: 500 $
Avoid declaring Pub…stance Variables	JEE	yes	574 -[574	17,22							17,22	100	17	…ourly cost: 500 $
Avoid direct or indir…emote calls inside a loop	JEE	yes	…-[12]	0,36	33 -[33]	2,31	16 -[16]	5,6	1 -[1]	1	9,27	100	9,…	…urly cost: 500 $
Avoid directly instan…g a Class used as a managed	JEE	yes	…[507	15,21	140 -[140]	9,8	16 -[16]	5,6	2 -[2]	2	32,61	100	32,6	…rly cost: 500 $
Avoid double checke…ing	JEE	yes	5…	0,15	1 -[1]	0,07					0,22	100	0,22	…ly cost: 500 $
Avoid empty catch bl…	JEE	yes	76…	2,25	149 -[127]	8,89	66 -[21]	7,35			18,49	100	18,49	…y cost: 500 $

ALIMENTATION

Plan d'action CAST

Toutes les règles violées

Toutes les critiques

EVALUATION

Métriques de l'équipe du CoE

Métriques issues de CAST

Macro-estimation du **coût de correction** pour un **ensemble de règles en violation.**

Notre offre et nos outils

Offres de services	Infra client	Cloud IBM
Installation et mise en service de la plateforme CAST AIP	X	
Assistance à la prise en main des outils administration de la plateforme CAST AIP	X	
Accompagnement au changement (process, formation utilisateurs)	X	X
Intégration dans le cycle de développement	X	X
Paramétrage et passage de la première analyse	X	X
Passage des analyses itératives	X	X
Accompagnement à l'interprétation des résultats	X	X
Accompagnement à mise en œuvre de plans d'action	X	X
Mise en œuvre des outils spécifiques du CoE	X	X
Urbanisation, interaction entre applications	X	X

Nous disposons d'une **infrastructure** CAST déployée sur un **Cloud Softlayer basée à Paris**

Pour aller plus loin…et rester en contact!

[Démonstration du CAST Health Dashboard (CIO/cast)]
[Démonstration du CAST Engineering Dashboard (CIO/cast)]
[Démonstration de Imaging (cast/cast)]
[Démonstration du CAST Security Dashboard (CSO/cast)]

[Communauté IBM]

[Documentation CAST]

Frédéric SCHMITT
SQ/AFP CoE Leader
frederic.schmitt@fr.ibm.com / 06.70.76.91.99

Véronique LEOTARDI
SQ/AFP CoE Team Leader
vleotardi@fr.ibm.com / 06.70.21.79.37

5 Questions

Agenda

1. Le contrôle qualité
2. La qualité tout au long du cycle de vie
3. La plateforme CAST AIP
4. SQ/AFP CoE: Qui sommes-nous ? Quelles sont nos références?
5. Notre offre et nos outils

A. Annexes

A Liste des règles vérifiées (Contexte RPG – CL400)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
CL400	Efficiency - Memory, Network and Disk Space Manageme	20	Avoid use of *NOMAX (CL400)		7
CL400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Programs with High Cyclomatic Complexity (CL400)		7
CL400	Efficiency - Expensive Calls in Loops	10	Avoid using SQL queries inside a loop		7
CL400	Efficiency - SQL and Data Handling Performance	9	Avoid SQL queries using functions on indexed columns in the WHER		9
CL400	Architecture - Reuse	4	Copy Pasted Code (% of LOC)		9
CL400	Architecture - Reuse	4	Avoid Too Many Copy Pasted Artifacts		9
CL400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid artifacts having recursive calls		8
CL400	Efficiency - SQL and Data Handling Performance	9	Avoid "SELECT *" queries		8
CL400	Documentation - Bad Comments	8	Avoid Artifacts with high Commented-out Code Lines/Code Lines ra		8
CL400	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with High Depth of Nested Subqueries		7
CL400	Efficiency - Memory, Network and Disk Space Manageme	20	Avoid use of RCLRSC (CL400)		6
CL400	Complexity - SQL Queries	18	SQL Complexity Distribution		5
CL400	Complexity - Technical Complexity	5	SQL Complexity Distribution		2
CL400	Programming Practices - Error and Exception Handling	20	Do not use MONMSG CPF0000 without EXEC (CL400)		1
CL400	Efficiency - Memory, Network and Disk Space Manageme	20	Avoid use of DLYJOB (CL400)		1
CL400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid locking object by ALCOBJ command (CL400)		1
CL400	Architecture - OS and Platform Independence	10	MONMSG command must be formated (CL400)		1
CL400	Efficiency - SQL and Data Handling Performance	9	Avoid use of OPNQRYF (CL400)		1
CL400	Dead code (static)	6	Avoid unreferenced CL Programs (CL400)		0

A Liste des règles vérifiées (Contexte RPG – DDS400)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
DDS400	Efficiency - Memory, Network and Disk Space Manageme	20	Avoid Logical Files using DYNSLT (DDS400)		9
DDS400	Architecture - Reuse	4	Avoid Too Many Copy Pasted Artifacts		9
DDS400	Architecture - Reuse	4	Copy Pasted Code (% of LOC)		9
DDS400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid artifacts having recursive calls		8
DDS400	Documentation - Bad Comments	8	Avoid Artifacts with high Commented-out Code Lines/Code Lines ra		8
DDS400	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with High Depth of Nested Subqueries		7
DDS400	Complexity - SQL Queries	18	SQL Complexity Distribution		5
DDS400	Volume - Number of Components	4	Avoid Physical files with more than X fields (DDS400)		5
DDS400	Architecture - Reuse	4	Avoid Physical files with fields defined locally (DDS400)		4
DDS400	Complexity - Technical Complexity	5	SQL Complexity Distribution		2
DDS400	Architecture - Multi-Layers and Data Access	22	Avoid Logical File without associated Physical File (DDS400)		1
DDS400	Dead code (static)	6	Avoid unreferenced Display Files (DDS400)		1
DDS400	Dead code (static)	6	Avoid unreferenced Printer Files (DDS400)		1
DDS400	Dead code (static)	6	Avoid unreferenced Physical file (DDS400)		1
DDS400	Dead code (static)	6	Avoid unreferenced Logical file (DDS400)		1
DDS400	Volume - Number of Components	4	Avoid Physical files with more than X associated LF (DDS400)		1

A Liste des règles vérifiées (Contexte RPG – RPG300)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
RPG300	Architecture - Object-level Dependencies	17	Avoid Programs with High Fan-Out (RPG300)		9
RPG300	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Programs with High Cyclomatic Complexity (RPG300)		8
RPG300	Documentation - Volume of Comments	11	Avoid undocumented RPG300 Programs (RPG300)		8
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid using NOT IN (RPG300)		7
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid using NOT EXISTS (RPG300)		6
RPG300	Architecture - Reuse	4	Copy Pasted Code (% of LOC)		9
RPG300	Architecture - Reuse	4	Avoid Too Many Copy Pasted Artifacts		9
RPG300	Complexity - Algorithmic and Control Structure Complexit	19	Avoid artifacts having recursive calls		8
RPG300	Documentation - Bad Comments	8	Avoid Artifacts with high Commented-out Code Lines/Code Lines rat		8
RPG300	Documentation - Volume of Comments	11	Avoid Programs with a very low comment/code ratio (RPG300)		6
RPG300	Architecture - Object-level Dependencies	17	Avoid Programs with High Fan-In (RPG300)		5
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with SELECT * statement (RPG300)		5
RPG300	Volume - Number of Components	4	Avoid Programs with more than X Subroutines (RPG300)		5
RPG300	Volume - Number of LOC	3	Avoid Programs with more than X lines of code (RPG300)		5
RPG300	Volume - Number of LOC	3	Avoid Subroutines with more than X lines of code (RPG300)		5
RPG300	Architecture - Object-level Dependencies	17	Avoid Subroutines with High Fan-Out (RPG300)		4
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with too many FROM Clauses (RPG300)		4
RPG300	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Subroutines with nested if Statements (RPG300)		3
RPG300	Architecture - Object-level Dependencies	17	Avoid Subroutines with High Fan-In (RPG300)		3
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid using dynamic SQL (EXECUTE IMMEDIATE) (RPG300)		3
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with COUNT(*) statement (RPG300)		3
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with too many WHERE Clauses (RPG300)		3
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs With Queries using GROUP BY (RPG300)		3
RPG300	Volume - Number of Components	4	Avoid Programs with more than X LoC and no Subroutines (RPG300		3
RPG300	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Subroutines with High Cyclomatic Complexity (RPG300)		2
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs With Queries on more than 4 Tables (RPG300)		2
RPG300	Programming Practices - Error and Exception Handling	20	Specify Error Subroutine for File Exception Handling (RPG300)		1
RPG300	Programming Practices - Structuredness	13	Avoid using obsolete "Bit Operations" statements in RPG Programs		1
RPG300	Programming Practices - Structuredness	13	Avoid using plain END statement, use the explained END statement		1
RPG300	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with Subqueries (RPG300)		1
RPG300	Dead code (static)	6	Avoid defining File Disk if not used (RPG300)		1
RPG300	Dead code (static)	6	Avoid unreferenced Subroutines (RPG300)		1
RPG300	Dead code (static)	6	Avoid unreferenced Programs (RPG300)		1
RPG300	Architecture - Reuse	4	Avoid using internally described files (RPG300)		1

A Liste des règles vérifiées (Contexte RPG – RPG400)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
RPG400	Architecture - Object-level Dependencies	17	Avoid Programs with High Fan-Out (RPG400)		9
RPG400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Programs with High Cyclomatic Complexity (RPG400)		8
RPG400	Documentation - Volume of Comments	11	Avoid undocumented RPG400 Programs (RPG400)		8
RPG400	Efficiency - Expensive Calls in Loops	10	Avoid using SQL queries inside a loop		7
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid using NOT IN (RPG400)		7
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid using NOT EXISTS (RPG400)		6
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid SQL queries using functions on indexed columns in the WHER		9
RPG400	Architecture - Reuse	4	Avoid Too Many Copy Pasted Artifacts		9
RPG400	Architecture - Reuse	4	Copy Pasted Code (% of LOC)		9
RPG400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid artifacts having recursive calls		8
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid "SELECT *" queries		8
RPG400	Documentation - Bad Comments	8	Avoid Artifacts with high Commented-out Code Lines/Code Lines rat		8
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with High Depth of Nested Subqueries		7
RPG400	Documentation - Volume of Comments	11	Avoid Programs with a very low comment/code ratio (RPG400)		6
RPG400	Complexity - SQL Queries	18	SQL Complexity Distribution		5
RPG400	Architecture - Object-level Dependencies	17	Avoid Procedures with High Fan-Out (RPG400)		5
RPG400	Architecture - Object-level Dependencies	17	Avoid Programs with High Fan-In (RPG400)		5
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with SELECT * statement (RPG400)		5
RPG400	Volume - Number of Components	4	Avoid Programs with more than X Subroutines (RPG400)		5
RPG400	Volume - Number of LOC	3	Avoid Procedures with more than X lines of code (RPG400)		5
RPG400	Volume - Number of LOC	3	Avoid Subroutines with more than X lines of code (RPG400)		5
RPG400	Volume - Number of LOC	3	Avoid Programs with more than X lines of code (RPG400)		5
RPG400	Architecture - Object-level Dependencies	17	Avoid Subroutines with High Fan-Out (RPG400)		4
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with too many FROM Clauses (RPG400)		4
RPG400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Subroutines with nested IF statements (RPG400)		3
RPG400	Architecture - Object-level Dependencies	17	Avoid Subroutines with High Fan-In (RPG400)		3
RPG400	Architecture - Object-level Dependencies	17	Avoid Procedures with High Fan-In (RPG400)		3
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with COUNT(*) statement (RPG400)		3
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs With Queries using GROUP BY (RPG400)		3
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with too many WHERE Clauses (RPG400)		3
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid using dynamic SQL (EXECUTE IMMEDIATE) (RPG400)		3

A Liste des règles vérifiées (Contexte RPG – RPG400)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
RPG400	Volume - Number of Components	4	Avoid Programs with more than X LoC and no Subroutines (RPG400		3
RPG400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Procedures with High Cyclomatic Complexity (RPG400)		2
RPG400	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Subroutines with High Cyclomatic Complexity (RPG400)		2
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs With Queries on more than 4 Tables (RPG400)		2
RPG400	Complexity - Technical Complexity	5	SQL Complexity Distribution		2
RPG400	Programming Practices - Error and Exception Handling	20	Specify Error Subroutine for File Exception Handling (RPG400)		1
RPG400	Programming Practices - Error and Exception Handling	20	Specify Program Status Data Area (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Assignment Operations" statements in RPG P		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "String Operations" statements in RPG Progra		1
RPG400	Programming Practices - Structuredness	13	Avoid obsolete E-spec (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Definition/Allocation/Array Operations" state		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Call Operations" statements in RPG Programs		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Conditional Operations" statements in RPG S		1
RPG400	Programming Practices - Structuredness	13	Better to use QUALIFIED data structures (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Bit Operations" statements in RPG Programs		1
RPG400	Programming Practices - Structuredness	13	Avoid obsolete L-spec (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using I-spec (RPG-IV only) (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Date Operations" statements in RPG Program		1
RPG400	Programming Practices - Structuredness	13	Avoid using O-spec; use Printer Files instead (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using obsolete "Arithmetic Operations" Statement in RPG Pro		1
RPG400	Programming Practices - Structuredness	13	Avoid using plain END statement, use the explained END statement		1
RPG400	Efficiency - SQL and Data Handling Performance	9	Avoid Programs with Subqueries (RPG400)		1
RPG400	Dead code (static)	6	Avoid unreferenced Subroutines (RPG400)		1
RPG400	Dead code (static)	6	Avoid defining File Disk if not used (RPG400)"		1
RPG400	Dead code (static)	6	Avoid unreferenced Programs (RPG400)		1
RPG400	Dead code (static)	6	Avoid unreferenced Copy Members (RPG400)		1
RPG400	Architecture - Reuse	4	Avoid using internally described files (RPG400)		1
RPG400	Volume - Number of Components	4	Avoid Programs with more than X Data Structures (RPG400)		1
RPG400	Programming Practices - Structuredness	13	Avoid using GOTO statement in RPG Programs (RPG400)		0

A Liste des règles vérifiées (Contexte RPG – SQL)

Technology	Technical criteria	Weight	Metric	Criticality	Weight
SQL	Programming Practices - Error and Exception Handling	20	Avoid empty catch blocks		7
SQL	Efficiency - Expensive Calls in Loops	10	Avoid using SQL queries inside a loop		7
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid exists and not exists independent clauses		6
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid Tables without Primary Key		5
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid non-indexed SQL queries		9
SQL	Architecture - Reuse	4	Copy Pasted Code (% of LOC)		9
SQL	Architecture - Reuse	4	Avoid Too Many Copy Pasted Artifacts		9
SQL	Complexity - Algorithmic and Control Structure Complexit	19	Avoid artifacts having recursive calls		8
SQL	Programming Practices - Structuredness	13	Avoid Tables not using referential integrity		8
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid too many Indexes on one Table		8
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid "SELECT *" queries		8
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid redundant indexes		8
SQL	Architecture - Multi-Layers and Data Access	22	Avoid having multiple artifacts deleting data on the same SQL table		7
SQL	Architecture - Multi-Layers and Data Access	22	Avoid having multiple Artifacts inserting data on the same SQL Tabl		7
SQL	Architecture - Multi-Layers and Data Access	22	Avoid having multiple Artifacts updating data on the same SQL Tabl		7
SQL	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Artifacts with High Cyclomatic Complexity		7
SQL	Architecture - Object-level Dependencies	17	Avoid Artifacts with High Integration Complexity		7
SQL	Programming Practices - Structuredness	13	Avoid using GOTO statement		7
SQL	Documentation - Volume of Comments	11	Avoid undocumented Triggers, Functions and Procedures		7
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with SQL statement including subqueries		7
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with queries on too many Tables and or Views		7
SQL	Efficiency - SQL and Data Handling Performance	9	Never use SQL queries with a cartesian product		7
SQL	Efficiency - SQL and Data Handling Performance	9	Avoid Artifacts with High Depth of Nested Subqueries		7
SQL	Volume - Number of Components	4	Avoid Tables with more than 20 columns on an OLTP system		7
SQL	Programming Practices - Structuredness	13	Avoid Artifacts with High Essential Complexity		6
SQL	Efficiency - SQL and Data Handling Performance	9	Prefer UNION ALL to UNION		6
SQL	Volume - Number of LOC	3	Avoid large Artifacts - too many Lines of Code		6
SQL	Programming Practices - Unexpected Behavior	20	Always define column names when inserting values		5
SQL	Complexity - Algorithmic and Control Structure Complexit	19	Avoid Artifacts with High Depth of Code		5
SQL	Complexity - SQL Queries	18	SQL Complexity Distribution		5
SQL	Complexity - SQL Queries	18	Avoid Artifacts with High RAW SQL Complexity		5

A Liste des règles vérifiées (Contexte RPG – SQL)

Technical criteria	Weight	Metric	Criticality	Weight
Efficiency - SQL and Data Handling Performance	9	Avoid using dynamic SQL in SQL Artifacts		5
Efficiency - SQL and Data Handling Performance	9	Use MINUS or EXCEPT operator instead of NOT EXISTS and NOT IN s		5
Efficiency - SQL and Data Handling Performance	9	Avoid using the GROUP BY clause		5
Efficiency - SQL and Data Handling Performance	9	DISTINCT should not be used in SQL SELECT statements		5
Efficiency - SQL and Data Handling Performance	9	Avoid NATURAL JOIN queries		5
Efficiency - SQL and Data Handling Performance	9	Replace OR conditions testing equality on the same identifier in SQL		5
Complexity - Algorithmic and Control Structure Complexit	19	Avoid Artifacts with too many parameters		4
Architecture - Object-level Dependencies	17	Avoid Artifacts with High Fan-Out		4
Efficiency - SQL and Data Handling Performance	9	Avoid queries using old style join convention instead of ANSI-Stand		4
Dead code (static)	6	Avoid unreferenced Functions		4
Complexity - Technical Complexity	5	Cyclomatic Complexity Distribution		4
Programming Practices - Unexpected Behavior	20	Avoid using quoted identifiers		3
Architecture - Object-level Dependencies	17	Avoid Artifacts with High Fan-In		3
Documentation - Volume of Comments	11	Avoid triggers, functions and procedures with a very low comment/		3
Efficiency - SQL and Data Handling Performance	9	Specify column names instead of column numbers in ORDER BY clau		3
Efficiency - SQL and Data Handling Performance	9	Avoid non-SARGable queries		3
Documentation - Style Conformity	6	Avoid Artifacts with lines longer than X characters		2
Complexity - Technical Complexity	5	SQL Complexity Distribution		2
Complexity - Algorithmic and Control Structure Complexit	19	Cyclomatic Complexity Distribution		1
Architecture - Object-level Dependencies	17	Coupling Distribution		1
Dead code (static)	6	Avoid unreferenced views		1
Dead code (static)	6	Avoid unreferenced Tables		1
Architecture - Reuse	4	Reuse by Call Distribution		1
Volume - Number of LOC	3	Size Distribution		1

A Liste des technologies couvertes

Global Technology	Sub-technology / framework / language
Microsoft (not .NET)	ASP, Visual Basic
C / C++	C99, C++03, C++11 standards, Pro*C, Pro*C++, IBM DB2 SQC, IBM DB2 SQC++
Adobe	Flex
Fortran	Fortran
Java/JEE technologies	Java JDK, Java Server Faces (JSF),Java Server Page (JSP), Apache Struts, Struts Validator, STXX, Hibernate, JPA, EJB, Spring loC, WSDL, CDI, Spring Batch, Spring Data, Spring WebFlow, Spring Security, Spring MVC, MyBatis (Java), JAX-RS, JAX-WS
Message Queues	ActiveMQ (Java, Python), IBM MQ (Java, Python), RabbitMQ (Java, Python), Spring JMS, Spring APMQ
Mobile	Android, iOS/Objective-C, iOS/Swift
IBM Mainframe zOS	Cobol, JCL z/OS, IMS/DB, CICS, EGL
Microsoft .NET	ADO.NET, ASP.NET, ASP.NET Core, ASP.NET MVC Razor, Visual Studio / .NET Framework, .NET Core, .NET Standard, NoSQL for .NET, LINQ to Objects, LINQ to DataSets, LINQ to SQL, iFrame, Interop, WinForms, C#, VB.NET, Entity Framework, Silverlight, WCF, WPF, MyBatis
Oracle Enterprise Solutions	Oracle Forms/Reports, PeopleSoft, Siebel
PHP	PHP Core, Symfony
IBM PL/1	Enterprise PL/I for z/OS
Python	Python Core, Flask
IBM System i	**RPG, COBOL AS400**
Cobol (not IBM)	Microfocus Cobol, GCOS Cobol, Cobol VMS
SAP ABAP	ABAP, GCOS Cobol, Web Dynpro for ABAP, ABAP Script
Reporting Enterprise Solutions	SAP BusinessObjects
	SAP Business Intelligence BusinessObjects
Scripting Languages	Korn shell, Bourne shell, C shell
SQL	**IBM DB2 UDB, IBM DB2 z/OS, Informix, MariaDB, Microsoft SQL Server T-SQL, MySQL, Oracle Server PL/SQL, PostgreSQL, SQLite, Support for DDL and DML *.sql files using an over language of ANSI SQL-92, Sybase ASE T-SQL, Teradata**
NoSQL	MongoDB (Java, ,NET, Node.js), Couchbase (Java, Node.js), Marklogic (Java, Node.js)
Misc. Enterprise Solutions	TIBCO BusinessWorks, TIBCO ActiveMatrix BusinessWorks, SAP PowerBuilder, Oracle BPEL
Web	Angular, Angular.js, CSS, HTML, JavaScript, JavaScript ECMA, Jscript, jQuery, ReactJS, SAPUI5, TypeScript, VBScript, XHTML
Node.js	Node.js, Express.js, Hapi.js, Sails.js, LoopBack