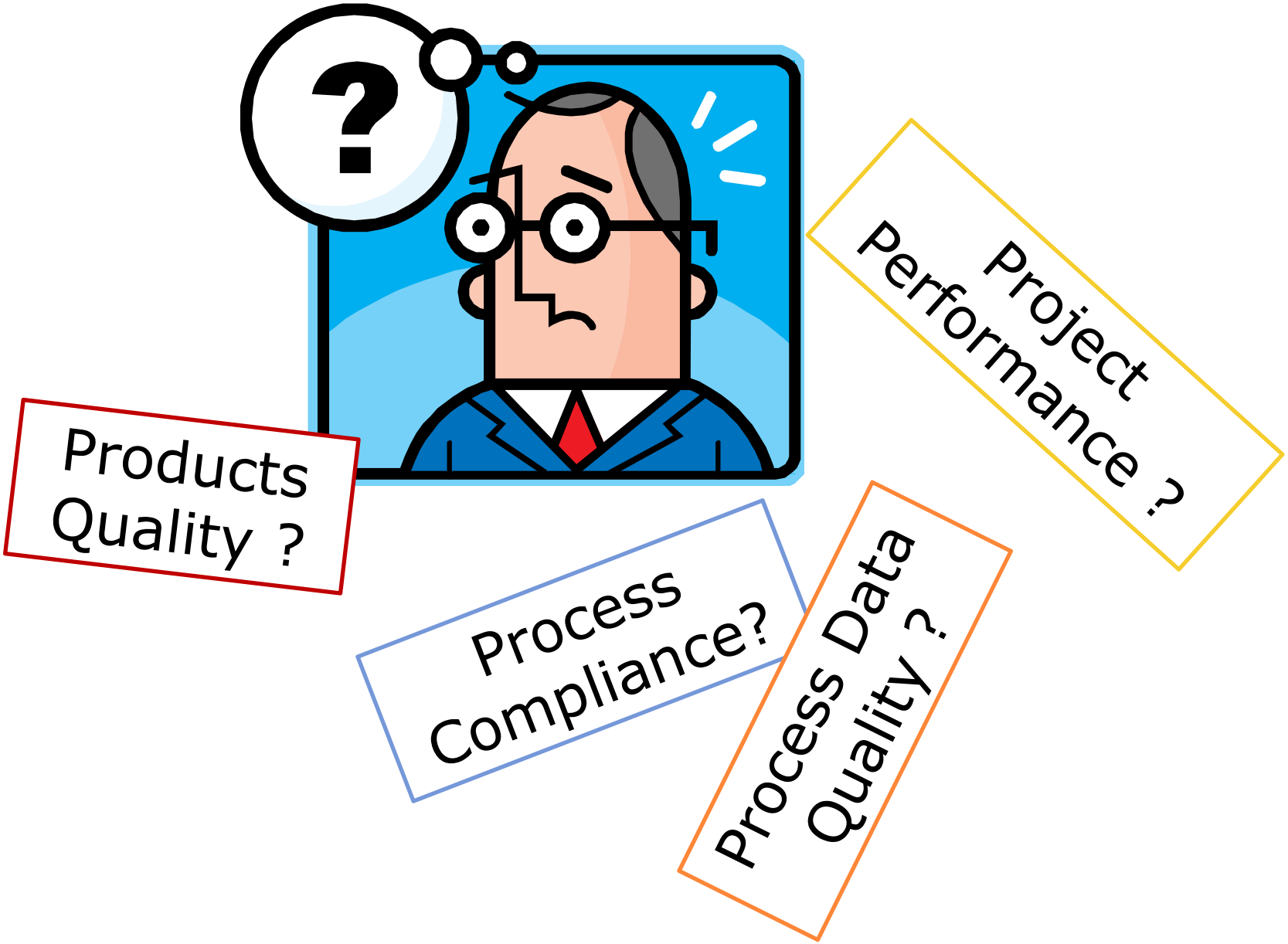

Products Quality ?
Process Compliance?
Process Data Quality ?
Project Performance ?

Product Quality and Process Compliance

Monitoring with metrics

Jean-François Cazemajou
Olivier Lafourcade
26/05/2014

Agenda

- Initiative Genesis
- Platform & Concepts Overview
- Quality Product model (C Model)
- Engineering Scorecard Model (CR/Escape defect analysis)
- End of Phase Review Models
- Feedbacks / Lessons Learned

Initiative Genesis

Top Goals of the Dashboard initiative / mid 2012

- Consolidate metrics and dashboard
  Standardize **project monitoring dashboards**

- Reduce cost of metric's
  Automate data collection, management & reporting

- Insure CMMI conformity
  Provide an integrated projects metrics **historical database**

DAR Result : DataDrill – SQuORE – Internal solution

The criteria of the DAR were (with their weight):

(5) Functional coverage
(5) Result demonstrator phase
(4) Recurrent cost
(4) Vender long tem robustness
(3) Attractiveness solution
(3) Commercial relation ship
(3) Evolution potential
(3) Non Recurrent cost
(3) Solution availability
(2) Infrastructure compliance

Solution	Evaluation (DAR tool)
DataDrill	443
SQuORE	629
Rockwell Collins France (Internal Developed Solution)	243

SQuORE – selected

SQuORE was selected as a **metrics management platform**.

The extra bonus:

SQuORE is sold as a **Decision-making Dashboard for System and Software Project Management**

Delivered prepackaged with:

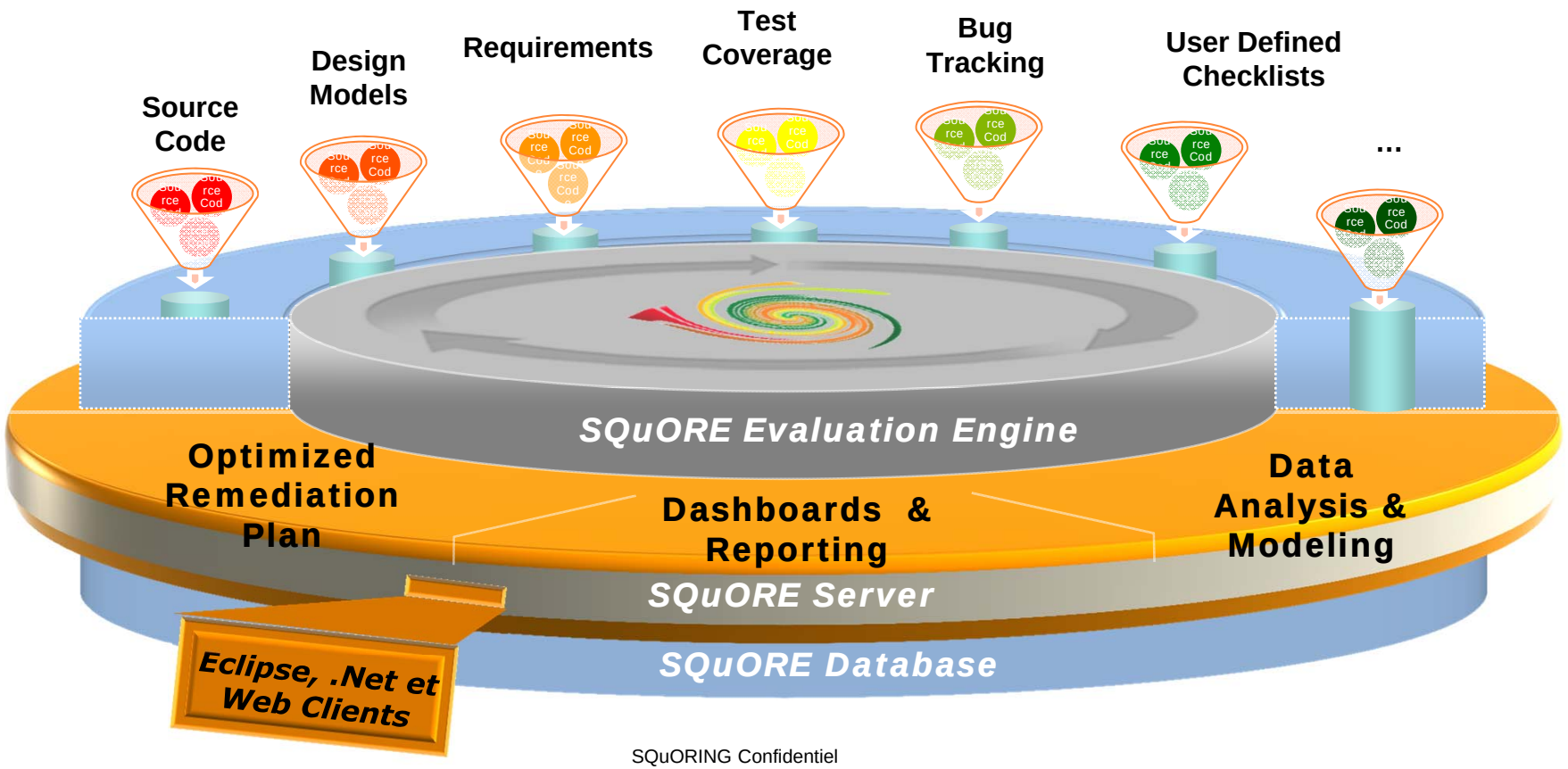
- Native Source code analyzers (C,C++, Java, Python, ADA,XML …)
  - Control flow analysis, Source Code Metrics, Rules Checking, Code Cloning detection
  - Open source detection capabilities
- Third-party source code analyzers
  - Logiscope, Coverity, PRQA /QAC, PC-Lint, PMD, Findbugs, CheckStyle, CPPCheck, VectorCast, Polyspace,…
- Assessment models (Maintainability ISO9126, Technical debt, Automotive HIS/MISRA compliance, System Engineering,….)
- Data providers for Requirement engineering
  - DOORS, Reqtify, Excel
- Data providers for Test monitoring
  - RTRT, Gcov, Junit, …
- Data providers for CR/BR management
  - Jira, Clearquest, Subversion, Git,…
- Non exhaustive list
  - See SQuORING website for full list

Platform & Concepts Overview

Rockwell Collins

SQuORE: An Open and Scalable Architecture

A Collaborative Platform for Optimizing Software Project Management

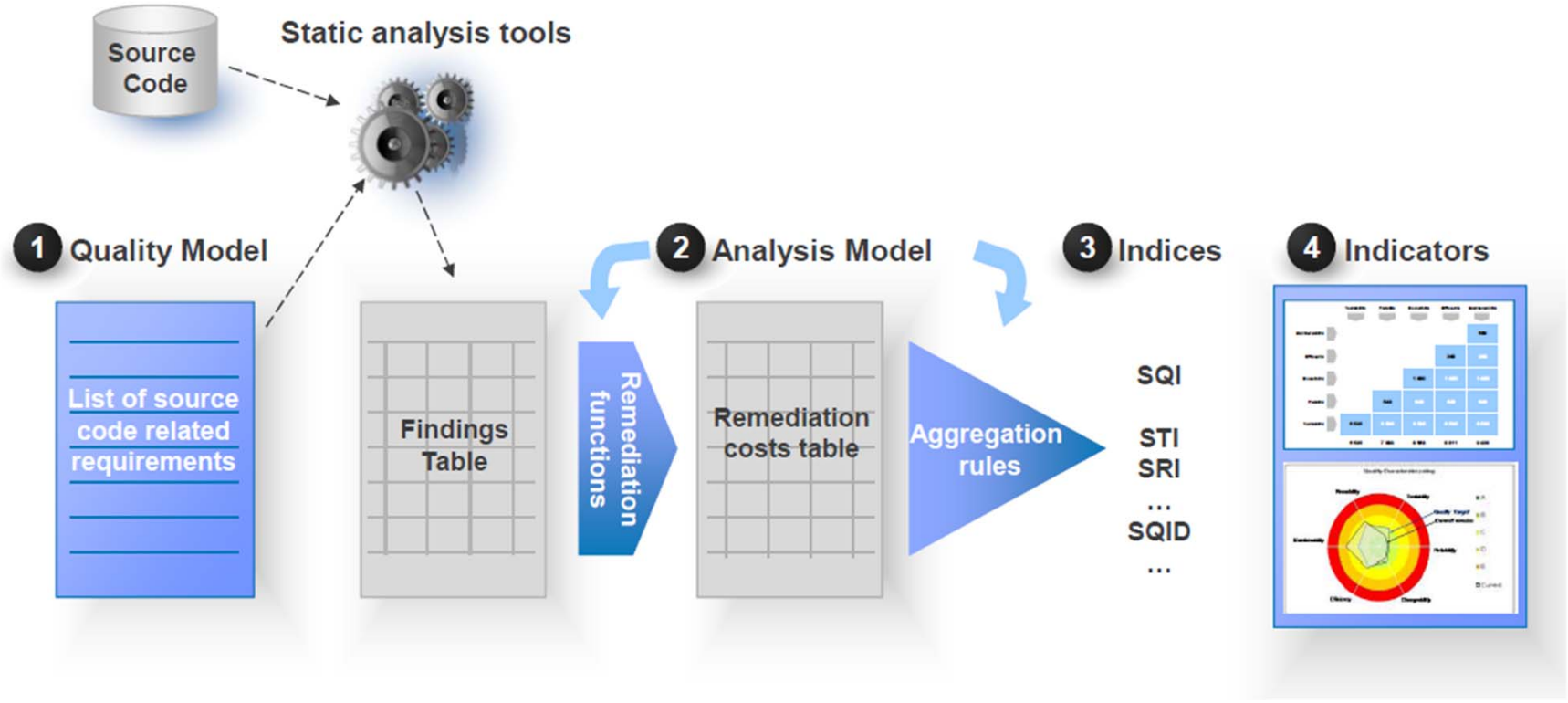


Concept : Technical Debt Management Method

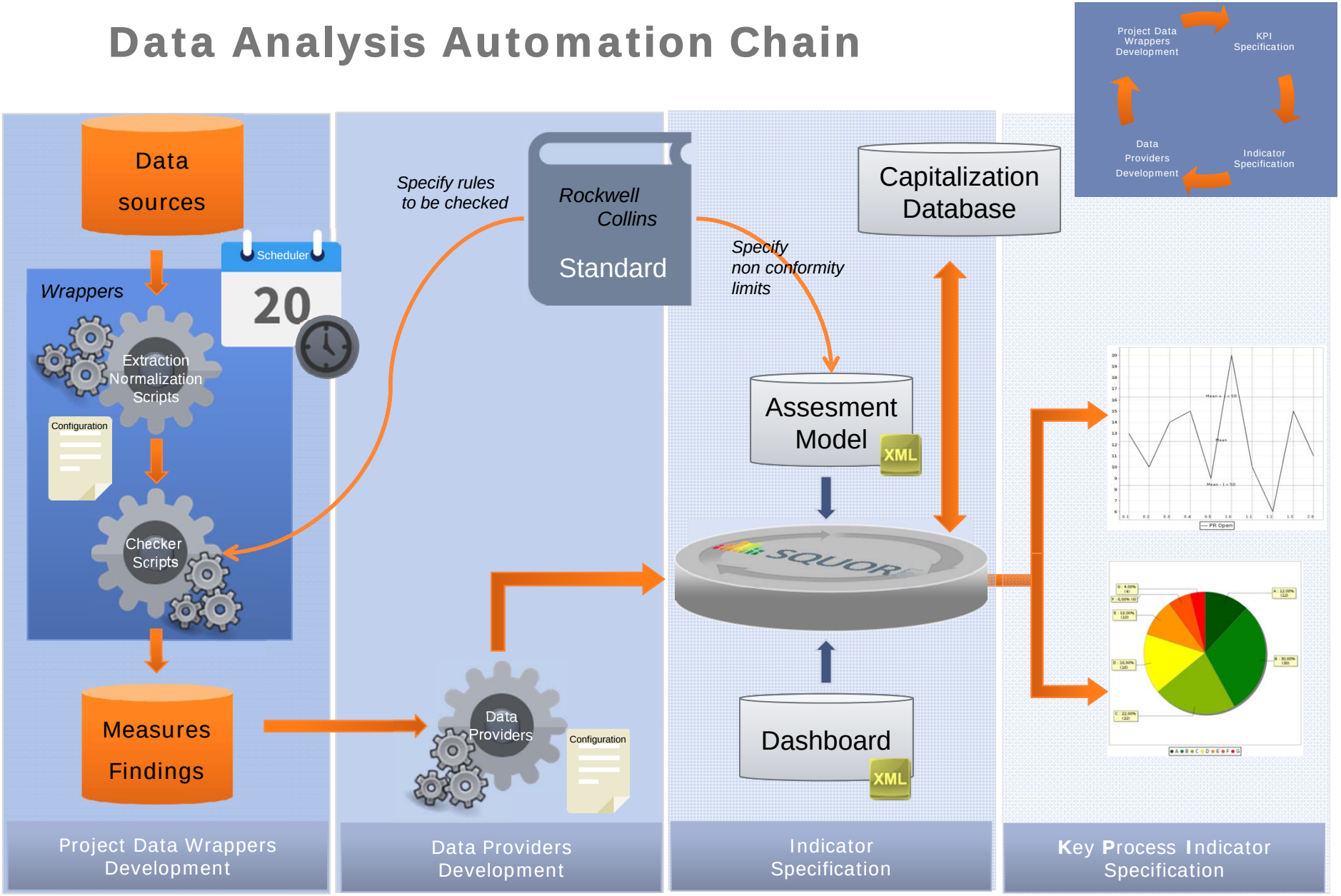
- The **Technical Debt** (or Design Debt) Management method applied to software developments allows:
  - To define clearly what creates the technical debt (violation of coding/design standards)
  - To estimate correctly this debt (severity, remediation cost)
  - To analyze this debt upon technical and business perspective
  - To propose remediation strategies and plans (action items, portfolio analysis).

- This method
  - is targeted for an automated implementation
  - is generic
  - is language and tool independent

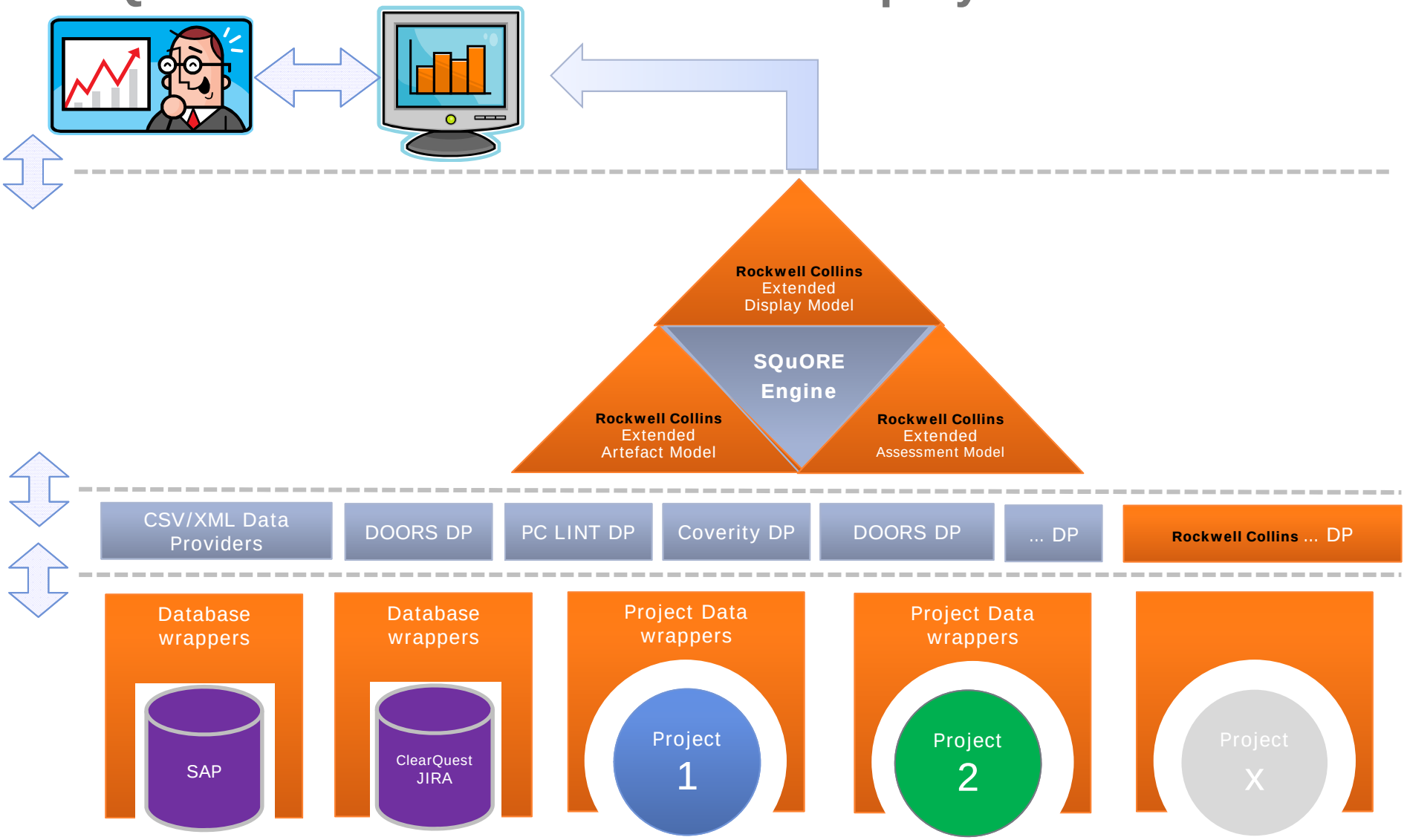
Quality Model Analysis Principle



Data Analysis Automation Chain



SQuORE Framework Platform Deployment



Quality Product Model

RCE C Model

RCE – Product Quality implementation

- **RCE standard for C langage** fully reworked according PC-Lint and SQuORE capabilities

	Before	After
Standard Specification 	188 rules sorted into: • Rule • Guideline	101 rules in the standard + 27 extra debugging themes, Breakdown into 5 severity levels
Verification 	➢ Manual ➢ Partly automated locally by projects	➢ **75 % of Blocking / Advisory / Required rules are automatically verified** ➢ 46 % of C coding standard rules are automatically verified ➢ The extras are automatically checked (220 PCLint & SQuORE checkers) => debugging purpose

	RCE C Standard	Debugging
Blocking	2	9
Required	26	
Advisory	25	4
Guideline	41	
Information	7	14
	101	**27**

Engineering Scorecard Model

Process Quality – Escape defects

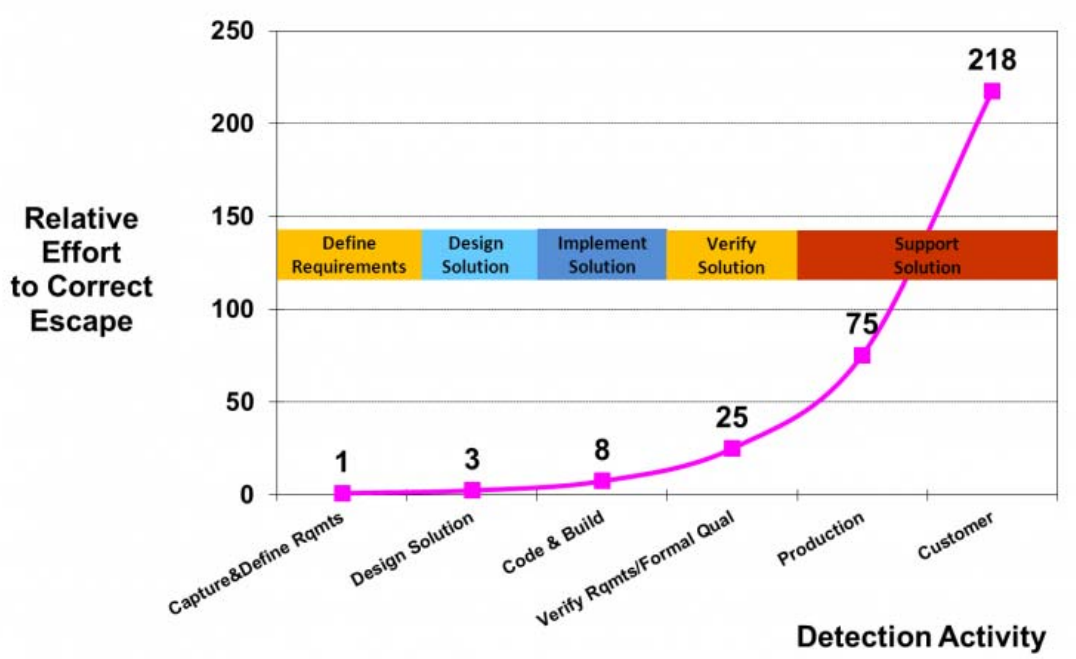
RCF Engineering Scorecard Model Genesis

- Previously Quality measure (TGNQLS/H) was based on the number of bugs delivered to customers on a Black Label unit.
  - Good start but:
    - Lag indicator
    - No direct corrective actions on the processes
    - Highly dependent on the customer use of the delivered product
    - For HW, the indicator was dependent of the volume of units sold
    - ….
- Decision to start to analyze Escape Defects rate
  - Firstly on Change Request data from ClearQuest (coarse grained data)
  - Secondly on Peer Review Data

Engineering scorecard – Escape Defect Concepts

- Escape Defect : Error which is not detected in the phase where it has been injected
- Saved Defect : Error which was detected in the phase where it has been injected
- Escape Rate = Escapes /(Saved +Escapes)

- The later the detection occurs the more it costs:



Engineering Scorecard Model Analysis

- First analysis based on Escape Rate metrics per discipline showed heterogeneous bad results from project to project (30 to 85%).
  - Main root causes:
    - “Saved defects” are rarely captured (only on specific projects)
    - ClearQuest is so heavy that it’s only used after a 1st delivery to customer (e.g: Excel is used for the first dev. phases.)

- Decision to focus on **Escape Distances** in order to:
  - Be independent of project context
  - To not force people to do extra work
    (e.g: to record non valuable “save defects” just for the sake of a “good escape rate”)
  - Easy to understand

- **Average Escape Distances** is used to monitor process quality.

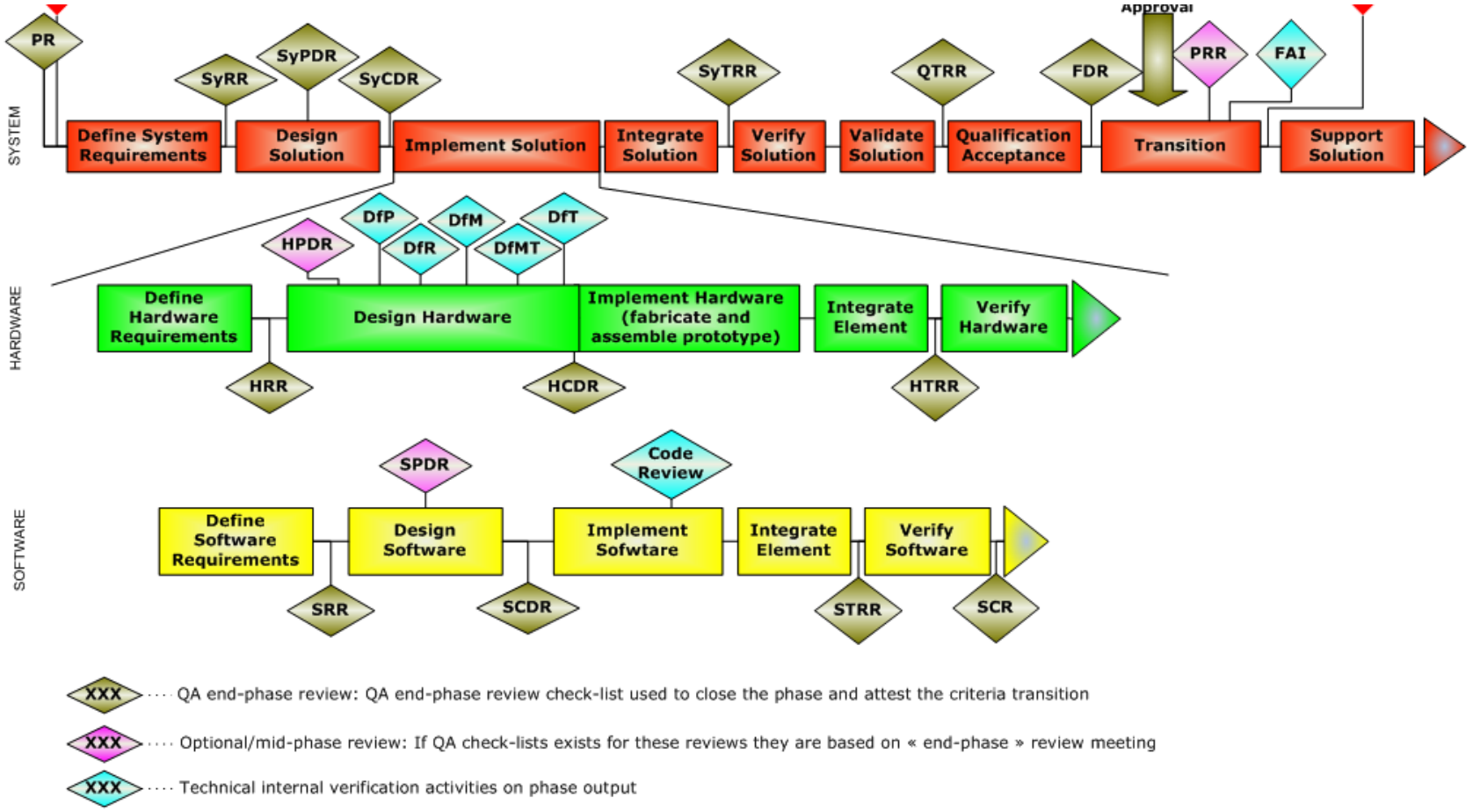
Engineering Scorecard Model / Issues

- 3 year floating window time frame was selected
- Data analysis showed that data is not reliable
  - Usable PR CR rate is between 40% to 95%
  - Manual informal sampling showed invalid detected data
    - E.g people set generally “implementation” as injection discipline
- Decision for FY14 was to set an Objective on the data quality only :
  - 95 % of the PR defects shall be properly filled-in (discipline; injection; detection) **(Usable PR CR Rate)**
  - Escape Distance goals will be set for FY15
  - Actions:
    - Massive training performed in October/November 2014
    - Improvement of the Analyzer to cope with specific use cases

End of Phase Review Models

Process Quality – End of Phase Review Checklist

End of phase Review Model: Life Cycle overview



End of phase Review Model: Checklist content

- End of phase Checklist are Quality Gate to verify process performed in the ending phase.

- Possible Outcome of the review: Passed / Conditional Passed / Rejected
  - A rejected outcome will iterate in another version of the Check-List
- All stakeholders are identified (QA, PM, Engineers….)
- Possible outcome of a question: OK, KO, N/A
- Questions are :
  - targeting generic checks (based on DO standards/phase/discipline)
  - also implementing past root cause analysis
- A comment can be attached to any question.
- A list of identified actions
- Approval date/signature
- Stored in InfoPath/XML format

End of phase Review Model: Information needs

The very same data are analyzed to address 2 different information needs:

- Monitoring project progress and Quality
- Analyzing organizational process performance
  - Global Quality indicators
  - Identify process improvement opportunities
    - Root Cause Analysis on recurrent KO or N/A
    - Analyze correlation between Questions
- 2 models very close with a different artefact tree

Feedbacks / Lessons learned

Global feedbacks

- Visibility provided to all stakeholders / based on figures
  (very appreciated from team leaders)
- Help to standardize / reinforce best practices
  (monitoring of acquired/lost practices)
- Improve the ROI of “manual” review
- With the finding capability, data quality can be evaluated
  (metrics on metrics, Confidence Indicator => Can i trust my data ?)
- Improve maintenance efficiency
  (e.g: I have 20h to spend, what will be the best investment for the project ?)
- Very opened framework => versatile solution
- Multi-sources data analysis capabilities (not only an information buffet)

Project deployment feedbacks

- Best ROI if deployed at the start on the project
  - The education will start at the beginning
- Even “one shot” or retro analysis brings added value…
  - E.g: Client delivery review, bug tracking, reuse assessment (complexity, coupling, comment ratio, technical debt….)
- Low entry ticket for project for the tool bundle deployment
  - compared to a PC-Lint deployment/customization and analysis effort
- Handle legacy code:
  - SQuORE able to only trigger action on new/modified code (not oblige to retrofit everything to start improving)
- Most of the time, needs to have 2 analysis scopes:
  - Delivered code only
  - Full code scope (Inc. test cases, debug perimeter, all platforms…)
- Good integration with Agile principle (continuous integration)
- Best results when integrated in the building procedure
- SQuORE suite tools can be run aside the project without impacting it
- Every project has specific needs in the setup (project calibration):
  - Files to ignore from the automatic analysis
  - Need to agree on deviations (ex: data file, rom, special construction)
    - setup relax mechanism that remain in the source code to solve special deviation
  - Legacy practices

Software Code analyser feedbacks

- SQuORE rules mechanism allow to aggregate violations from analyzers
  - less fuzzy information to analyze by the teams and leaders
    - Limit the noisy output from analyzers
  - The violations are described directly from your standard (full control, URL..).
- Even with a non customized model, projects can benefit from SQuORE.
- Objective and fast assessment for code developed by a subcontractor/partner. (Apple to apple comparison…)
- Test case suite needs to be developed to verify analyzer detection capabilities and correct interpretation in the model.
- Standardization of SLOC analysis (count, stability, comment ratio…)
- Global relax mechanism: the rule is relaxed at the source level and SQuORE takes into account the relaxation whatever the static analyzers used.
- Good cloning detection at different level : block, function, class, file, component.

Model definition feedbacks 1 / 2

- The model definition is a never-ending activity
  - there is always something to improve, perfection cannot be achieved…
    - Model development must be iterative and time-boxed…
  - «Essentially, all models are wrong, some are useful » (G. Box)
  - Keep in mind that **KPI are there to trigger questions**
- Limited integration of models : (typically 1 model per language/domain)
  - too complex to try to setup and maintain a multi-domain model
  - workaround : models integration through rendering in ALM Confluence/JIRA
- Under-estimated effort on language standard development
  - Refresh of the existing standard
  - Enhancement with the different static analyzer tools capability
  - Traceability between rules ⇔ standard
  - Test-cases for the different analyzer tools
- For SW model, reuse existing models delivered by SQuORE, and inject changes by iteration
- Model calibration is essential :
  - A set of representative projects need to be defined early to efficiently support the calibration

Model definition feedbacks 2 / 2

- Under-estimated calibration effort
  - « perfect » calibration is something quite unreachable
  - Difficult to calibrate with heterogeneous projects (size, legacy, application context…) => project parameters to the model
  - Need to find the balance between the feeling from the developers and the rating based on the violations
  - => time-boxing required….
- A “one size fits all” model is difficult to achieve…
  - Warning: multiplying/cloning models should be avoided (maintenance effort issue)
    - Try first to parameterize the model instead to better match project goals & risk (e.g: Maintainability, Reliability, Portability weights)
    - If a product line context is really far away the “normal way”, it will be smarter to define a dedicated model for the product line (loosing the capability to “compare” projects outside the product line) (too many parameters will increase the complexity of the model : definition, calibration, projet setup, maintenance efforts will increase drastically)
- For model accuracy, a perfect fit needs to be setup with your analyzer (each rule defined in the model shall be triggerable by static analyzers).

Questions ?