metaware.

the complete IT modernization solution

Managing and Enhancing Code Quality

From Mainframe Asset Analysis to Code Quality Governance

Refine® for Quality Assessment

A Metaware White Paper

EXECUTIVE SUMMARY

Increasingly, organizations are faced with the challenges of managing and modernizing valuable, albeit aging, mainframe applications portfolio. Code quality is often a key concern with these valuable applications and can have a significant impact on the cost of running these applications, their performance, as well as the development and maintenance productivity. At the same time, the prerequisite to any modernization initiative is the ability to analyze the entire asset and the associated value. However, decades of developments and maintenance have increased size and complexity and the code quality governance challenge lies into the massive volume of source code, determine areas of interest, and then focus effort on the applications with the most urgent needs.

As the most powerful reverse engineering technology commercially available, Refine® is specifically designed to capture and provide factual data from an existing source code asset, allowing for decision making, and driving choices from the start.

- **Development Strategy** – How to build a development strategy for a specific application or a group of applications? What's the best modernization option; replace, rewrite, migrate, outsource?
- **Cost Control** – How to reduce software maintenance cost? How to negotiate trade-offs between development productivity (best practices), outsourcing, and code maintainability?
- **Communication** – How to share and communicate the existing situation and the development strategy with the rest of the organization (lines of business, finance department, senior management)?
- **Maintenance Continuity** – How to recover control over maintenance operations while planning resources for maintenance activity based not on the past experiences, but on measurable units of production?

Refine® for Quality Assessment is structured around two main dimensions to help organization:

- **Understanding a Mainframe Asset**: collecting and storing source code, analyzing components, associated attributes and mapping-out their dependencies, generating visual representations, etc.
- **Assessing and Measuring Code Quality:** as there's no one-size-fits-all modernization solution defining metrics on the basis of specific objectives identified by each organization, and obtaining and using the measured results

To fully support the evolutions of your valuable applications, Refine® for Quality Assessment is complemented by comprehensive solutions relying on the same Code Base Management System (CBMS):

- An **industrialization framework** to Manage and Enhance Code Quality over the maintenance releases
- The most **accurate and automated solutions for modernizing and transforming mainframe** assets including dead code removal, code refactoring, mass changes, language translation and replatforming

TABLE OF CONTENT

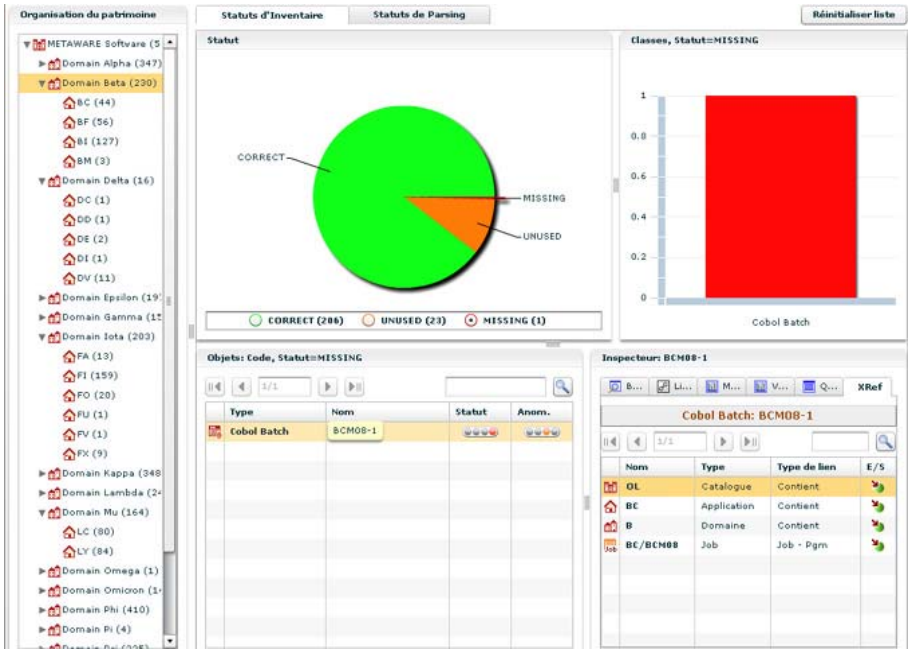
UNDERSTANDING MAINFRAME ASSETS

Over the years of developments and maintenance, mainframes have increased size and complexity and the challenge of understanding the asset lies into the massive volume of source code, intricacies and components used. Refine® is able to automatically discover and sort-out the asset components.

INPUT	OUTPUT
Components: programs, JCL, file, scheduler, On Line Transaction Services, screens	Inventory, rationalization and representation
Dependencies	Inventory, representation, navigation
File and database data structures	Inventory and rationalization (polysemy) in a centralized dictionary Semantic allocation
Internal component structure: programs, JCL, scheduler	Decomposition, representation and navigation
Documentation	Put together consolidate and extend program documentation, in a wiki
External information related to operations, maintenance, cost, performance	Aggregation, representation and correlation
Component Organization	Allocation, hierarchical organization, adherences calculation

Asset Inventory and Rationalization

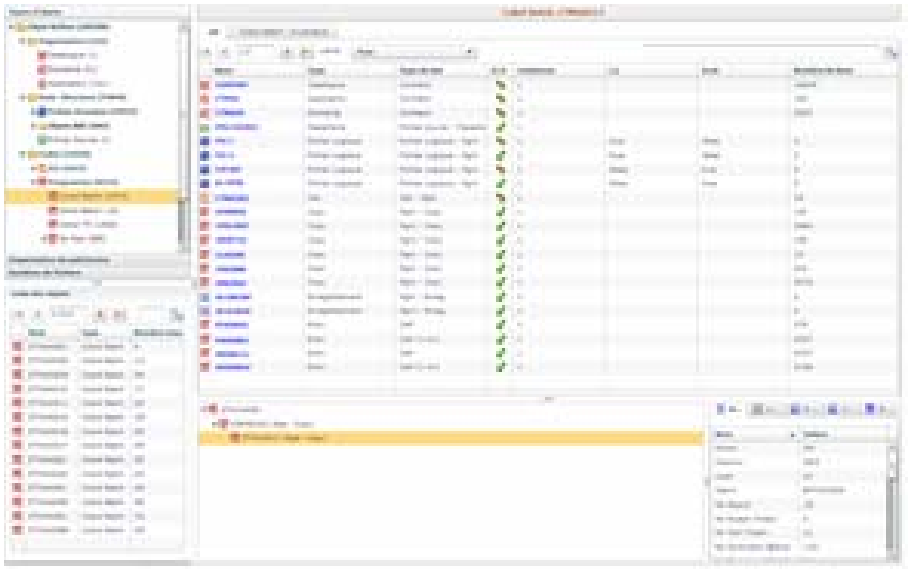
A specific desktop is designed to achieve the inventory and provides information on components by type (used, unused, missing, and orphaned).



Adaptability is a major criterion in the sense that the environments are rarely identical for complex and large environments. The solution relies on Refine® analysis capabilities and has proven its ability to process very large mainframe environments comprising several millions lines of codes and multiple languages.

Dependencies: Inventory, Rationalization and Navigation

The analysis is completed by identifying the dependencies between components (cross references) and delivers the initial level of mapping to serve for impact analysis. The power of Refine®'s engine associated with its code repository and manipulation capabilities can extract a significant amount of information related to inter- and intra- component relationships.



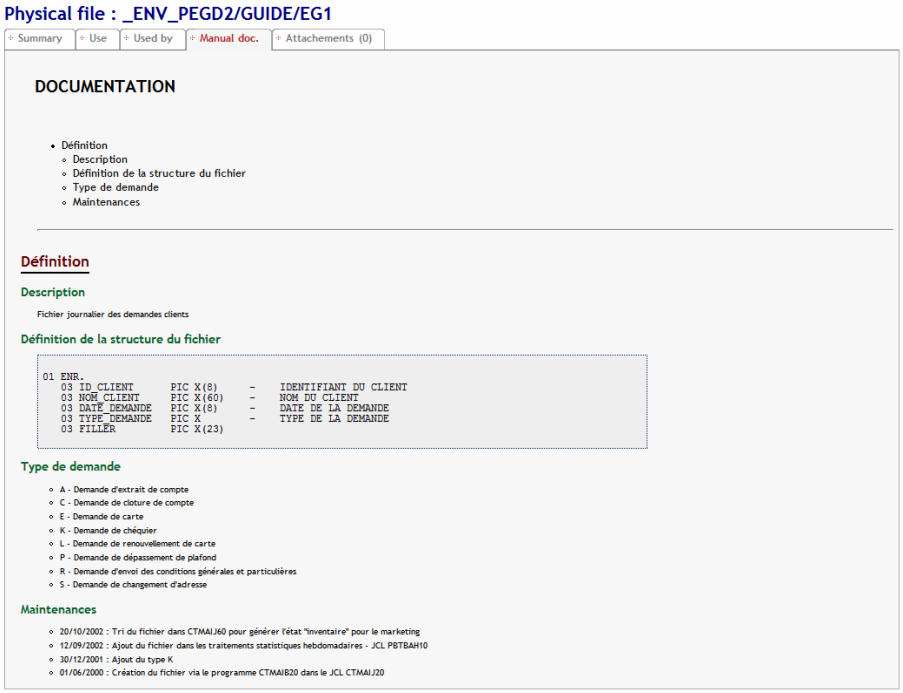
Collecting, Federating and Extending the Wiki Document Base

The two major concerns of obsolescence are the loss of knowledge and code complexity.

It is therefore necessary to extract, consolidate and extend the available documentation to put a stop to the loss of knowledge and reducing the cost of learning.

Refine® for Quality Assessment is used to:

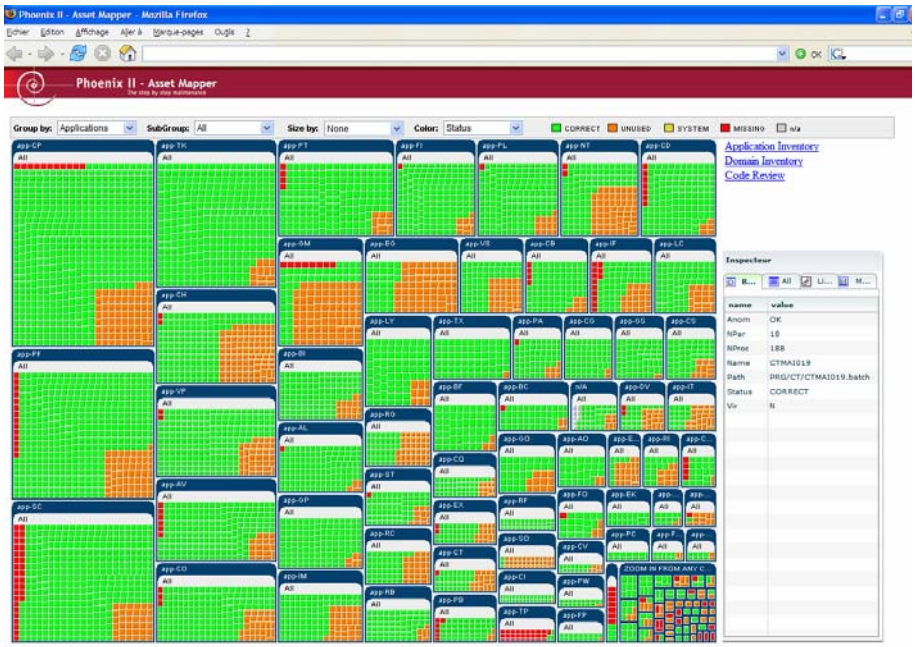
- Recover existing comments from the programs
- Federating all elements in a documentation
- Manually extending the documentation



The wiki is a means of accessing and highlighting application maintainer and operator knowhow.

Aggregating and Correlating External Data

Capturing the overall status of thousands of programmes and millions of lines of code can appear overwhelming, or even impossible. Refine®'s asset mapper instantaneously provides the overall status of a mainframe asset.

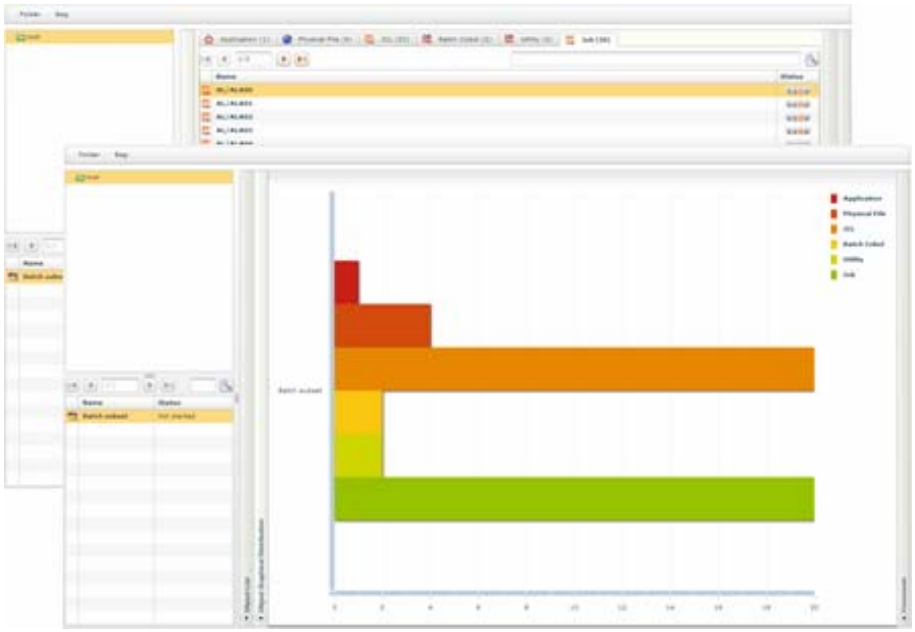


Refine® can also aggregate external information (e.g. maintenance and operation activities, applications usage) with the information extracted from code analysis. The resulting aggregated information can be also analysed and correlated into a powerful graphic interface.

Allocation

The allocation desktop is a precious asset in defining the list of components linked to a program or a set of components.

Allocation is used to bring together programs from the same functional/business domain or to build consistent batch subsets such as test plans or stages in resumption of production.



MEASURING MAINFRAME CODE

What you cannot measure cannot be improved.

Without measurements, i.e. without the ability of characterizing software aspects with quantifiable metrics, removing the qualitative fuzziness resulting from decades of management of IT assets is not feasible.

Lessons Learnt from Source Code Quality

More than fifteen years of practice have helped us understand that measuring software quality is a futile exercise when it does not lead to concrete actions. Without prior, accurate definition of the targeted objectives, it is also futile to accumulate metrics that are either too general or too abstract.

This is why Refine® allows to:

- select or define (if metrics are not defined yet) the metrics capable of translating the targeted objective into a concrete and accurate taxonomy; by relying on the unmatched expressivity of Refine® language
- simultaneously deploy several quality charters to take each user's role into account and therefore take his needs into account (development, testing, operation, domain management)
- exploit and translate the results into a work plan associated with a workload assessment.

The following diagram provides the main stages of the lifecycle of legacy software measurement.

Quality Measurement Charters

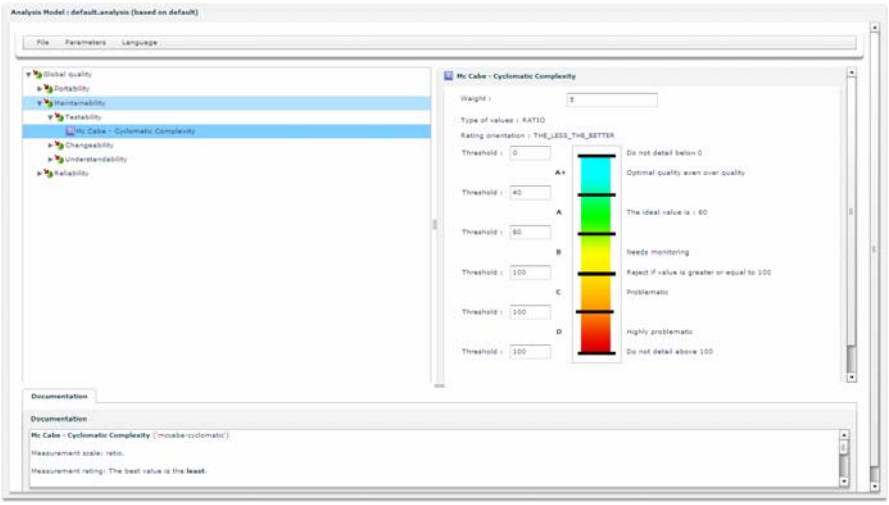
Measurement tools must be able to translate:

- the targeted objective: simplification of code, quality control, offshoring, etc.
- the role of each user in the organization
- the state of perpetual development in software to be measured

Refine® enables users to select and put together the most relevant metrics in quality charters that can be completely redefined.

Refine® provides several hundreds of pre-defined metrics covering domains such as:

- enforcement of coding rules
- reliability (defects)
- robustness
- performance (CPU)
- maintainability
- security

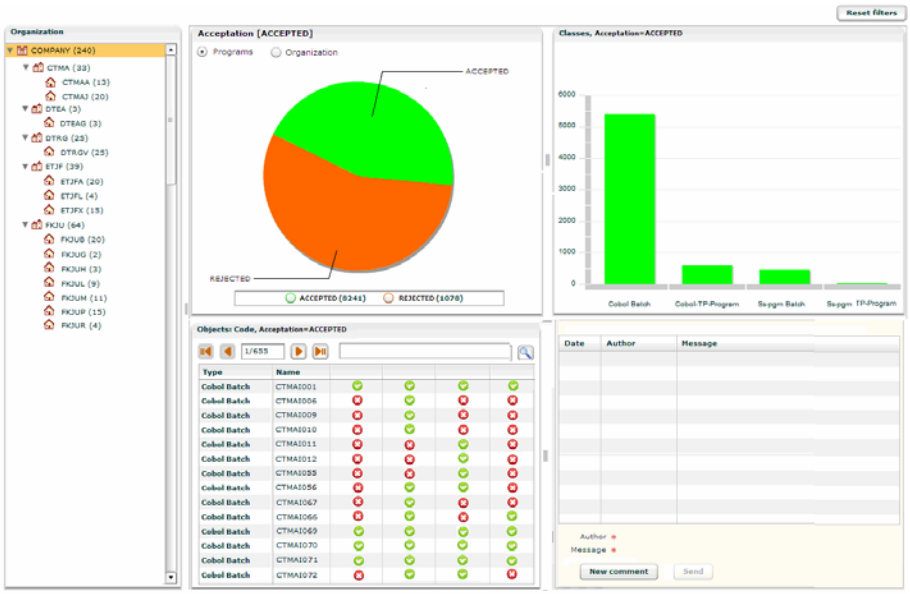


With this desktop, new set of quality charters can be easily changed or added by selecting the metrics or the measurement points defined in an ontology and associated quality thresholds.

Summarizing Measurement Criteria

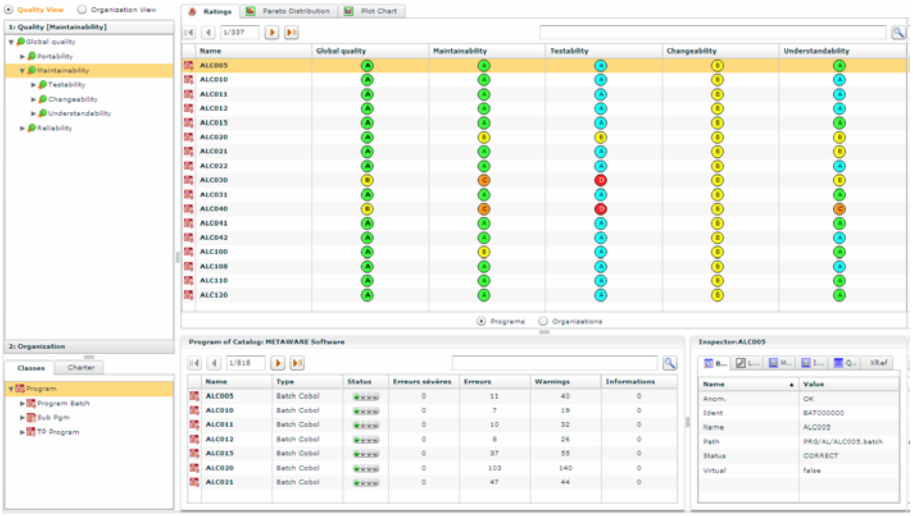
Apart from the detailed results produced by applying measurement quality charters Refine® produces an overview of the results that is easy to use, in particular, in the context of the acceptance procedure for new software releases.

- if the program meets the quality requirements, its status appears in green
- if it does not, its status appears in red



Diagnosis

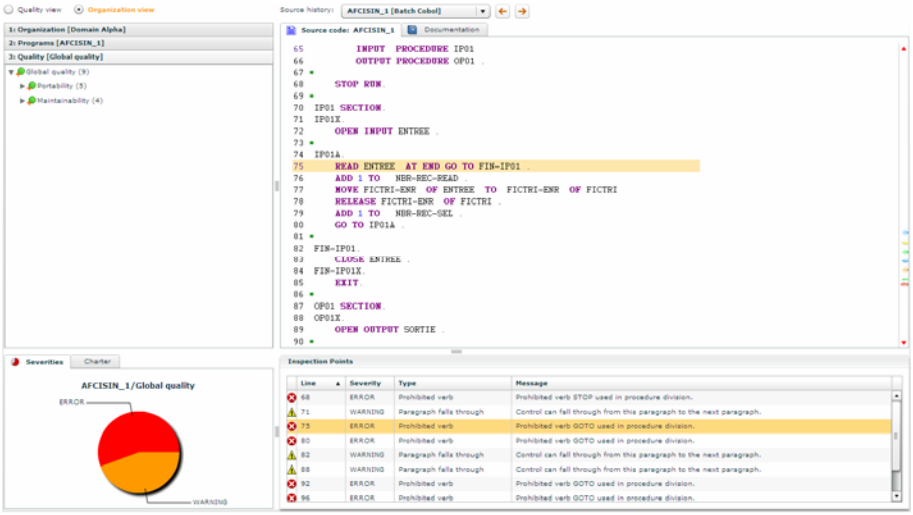
Once measurement has been completed, the next step is to go to the root causes, i.e. making a detailed diagnosis before commissioning an effective improvement plan.



This desktop is used to sort-out the measurements calculated either by the organization of the code (program, domain, application, asset) or the measurement criteria (quality charters, measurement theme, metric).

Code Examination

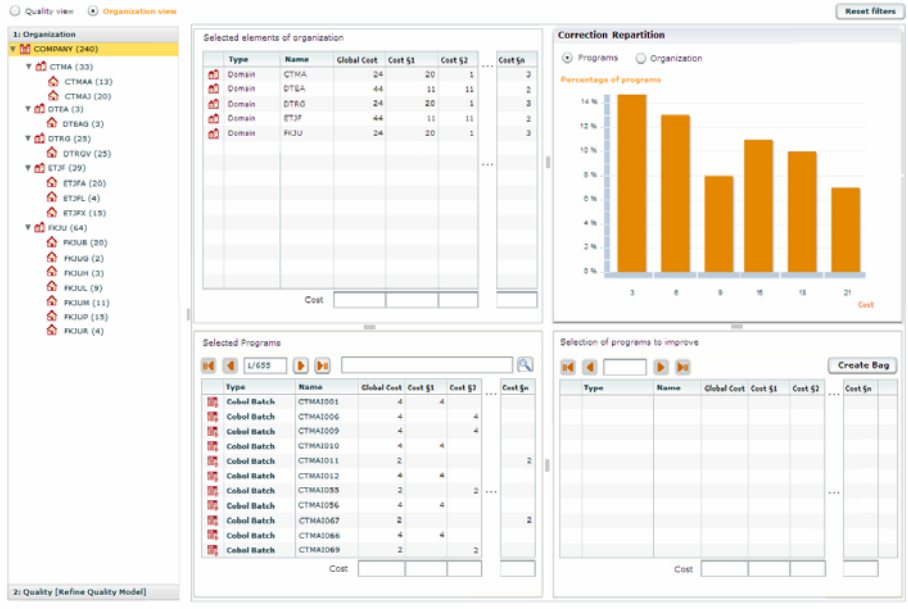
Drilling down at the code level can be required to look at the lines of code impacted by the measurement quality charters.



Improvement Plan

Making observations and diagnoses is one thing. Acting is the following key step.

Refine® allows evaluating the implementation workload of any improvement plan and therefore optimizing the cost of such a plan based on the targeted objectives.



INCREASING QUALITY AND REDUCING THE COST OF MAINTENANCE

The reduction of the maintenance costs requires:

- simplification of assets and code to maintain, by eliminating a proportion of application redundancies and reducing the complexity of code resulting of years of uncontrolled maintenance.
- selective or complete outsourcing of the maintenance activities in order to benefit from reduced costs, as long as tested and proven processes are implemented by the outsourcer.
- increased maintenance productivity by relying on the industrialization of their activity.

Beyond cost, the challenge lies into reducing project timeline and workload as well as the number of anomalies.

If achieving new maintenance releases in smaller amount of time is always more difficult, an incremental and repetitive approach of each releases can solve the problem. Metaware has built a methodology by relying on the deployment of structured and iterative processes providing frequent and systematic feedback, and hence continuous improvements by capitalizing on:

- the source code and its upgrades stored in Refine® repository Code Base. New releases come with a number of incremental changes, but the general structure of the code remains unchanged
- testing: test plans, test cases and anomalies as consolidated within a single test repository
- project data (workload and duration) are stored in a central repository and used to define and monitor productivity, quality and deadline metrics
- detailed process definitions (models, procedures, e-learning) are also stored in a central repository and its content is accessible from a dedicated user interface.

Metaware's methodology for industrializing the maintenance consists in:

- setting up a series of industrialization requirements in line with the goals of the organization
- designing, adapting and teaching the industrialization processes
- optimizing their implementation in order to meet and even exceed initial requirements

Key industrialization process templates include:

- continuous code quality improvement
- continuous application knowledge capitalization
- Industrialization of testing

Continuous Improvement of Quality and Performance

The goals are defined and formalized in the form of quality charters. A charter includes a series of measurable quality criteria (metrics and KPIs). Metrics such as the following are provided by Refine® CBMS. Each metric is associated with a Refine® inspection rule; out-of-the-box rule set can be customized and new ones can be easily and quickly developed.

- performance
- security
- robustness
- documentation
- risk of uncovered defects associated with the changes implemented in a new release
- complexity
- reliability

Continuous Knowledge Capitalization

Code quality, and particularly its structure, as well as productivity are largely impacted by the availability of business knowledge and skill set. As a result, capitalization is instrumental to the increase in maintenance productivity. By relying on the CBMS of Refine®, metadata are produced to compute, represent, and recover functional knowledge.

- functional breakdown
- dependency relationships
- data dictionary
- semantic assignment

Industrialization of Testing

Systems are growing in complexity, production schedules are getting tighter, so IT departments look at anything that saves time, money and help keep projects on schedule. With testing, the objectives are to reduce cost, improve coverage, and reduce QA phases and overall time-to-deployment.

The triangle of optimization is:

- increase test efficiency and subsequent maturity
- outsource testing to reduce daily rates
- simplifying the assets to test and increase "testability"

By relying on Refine®, exploiting the repetitive and incremental nature of each release is easy to validate each release within shorter cycles. The methodology is based upon the capitalization on:

- testware, i.e. on plans, cases, anomaly patterns centralized in a single test repository
- source code and its evolution, persistently stored in Refine® code base repository
- processes following their iterative execution including frequent and systematic feedback.

Metaware has designed, developed, automated and deployed a complete set of test processes detailing the lifecycle, tools, environments, organization and associated skills. They have been run, validated, measured, and optimized against predefined and thorough specifications. As a result, the specifications cover all the aspects of test industrialization.

CONCLUSION

Refine® comprehensively addresses the collection, analysis and metric management on mainframe source code. Refine® not only produces consolidated reporting on and across mainframe assets, applications and maintenance throughout time, but it becomes the central place to manage code quality. In addition, Refine® modernization solutions rely on the same very large Code Base repository to produce transformations such as code cleansing, refactoring, mass changes, automated translation, and replatforming.

Mainframe assets are quickly and visually sorted-out through specialized desktops that combine collected measures and calculated metrics on all your components and maintenance releases. Refine® desktops enable to navigate seamlessly through various granularity levels, from applications and maintenance portfolio to source code level. Metrics related to coding rules integrate hundreds of inspection rules. Those rules are grouped into families to offer a synthetic view (performance, reliability, maintainability…). New inspection rules can be easily and quickly developed by relying on Refine® very-high-level, wide-spectrum language. The transparency at all level (domain, application, intra- and inter-component relationships…) enables the maintenance teams to take immediate and targeted actions, to share and communicate existing situations and progresses across the maintenance value chain and with the rest of the organization.

By combining Refine® with a Code Quality Governance, customers can accelerate improvements of Code Quality, reduce maintenance costs, and pragmatically plan their modernization options.

metaware.
the complete IT modernization solution

www.metaware.fr
Tel. +33 1 30 15 60 00
Fax. +33 1 30 15 06 71

Global Headquarter:
1. Parc des Grillons
60, route de Sartrouville
78230 Le Pecq Cedex – FRANCE