QStudio® for Java
The Software Health Tool for Java

QStudio for Java Enterprise 2.0

QStudio for Java is a sophisticated code quality assessment and coding standards technology for Java development detecting potential software defects at compile time. It offers companies the ability to significantly reduce code review effort by automating a major portion of the code inspection and coding standards process. QStudio for Java gives developers the ability to automatically assess and control software quality concepts such as reliability, maintainability and efficiency.

QStudio for Java Enterprise is an enterprise strength collaborative solution that significantly improves software quality (and thereby productivity) by empowering all team members to automatically inspect and control the quality of Java based source code.

QStudio for Java Enterprise allows software process managers (software managers and QA managers) and software developers to automate, coordinate and support quality control through a distributed software quality control and management system. The client interface and functionality for the developer is identical to that of the Pro version. An additional web-based interface is available for the enterprise-specific functionality listed below. An online Enterprise server can be visited at www.qa-systems.com.

- Enterprise-wide control and reporting on quality metrics (ISO 9126 conforming) with respect to compliance of the source code to quality standards
- Code conformance to selectable departmental and project related code quality standards
- Trend analysis based on formal milestone creation showing the quality evolution
- Aggregated Code quality observations across software projects
- Fully web enabled graphical administration and reporting capabilities
- Command line interfaces for batch processing of large projects
- Interface with code management systems to ensure that only code satisfying the defined quality standard is committed to the code base

Enterprises are looking for ways to improve their software development processes. They want to improve time-to-market, software quality and the efficiency of software maintenance. QStudio for Java gives them the benefits of automated software code quality control:

- Reduce time to market by cutting testing time due to earlier detection of (potential) software errors
- Significantly reduce review effort (and therefore cost) by automating a major portion of the inspection process (adherence to coding standards, usage of best programming practices)
- Improve quality control and thereby code quality by directly supporting adherence to improved programming practices and corporate coding standards and avoid software quality degradation
- Assess the quality of its existing code base in order, for example, to establish maintenance budgets or for in-sourcing or outsourcing contracts

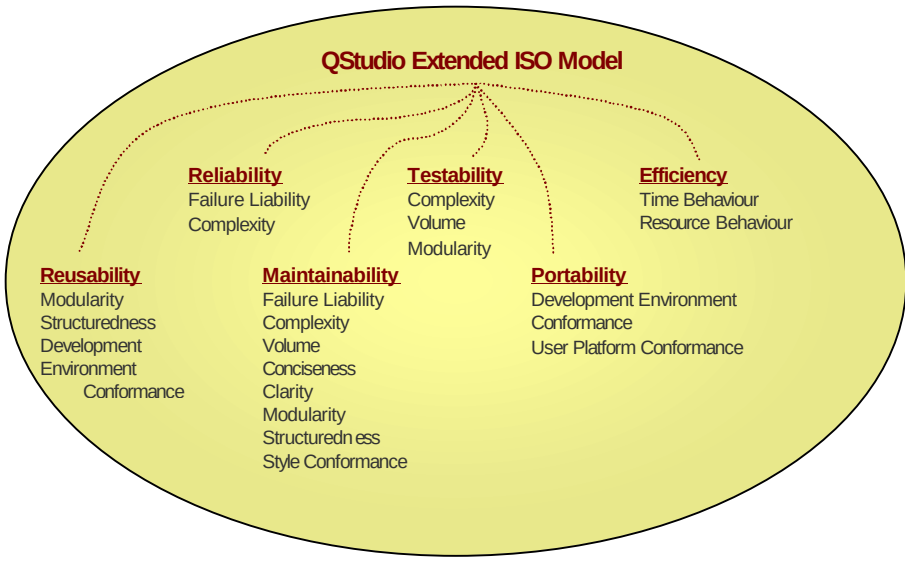
Research shows that static analysis can reduce defects by up to a factor of six. (Capers Jones, Software Productivity Group). Some 60% of the software faults that were found in released software products could have been detected by means of static analysis (Bloor Research). On average, research shows that 40% of the faults that could be found through static analysis will eventually become a defect in the production software. The payback is so powerful and direct that not making use of static analysis during the development cycle is throwing money away.

Control conformance to coding standards and the ISO 9126 quality standard

QStudio technology couples advanced static analysis capabilities to the ISO 9126 quality standard framework. QStudio for Java is the only static analysis tool on the market today that supports an explicit quality model that can directly tie into an organization's quality processes.

QStudio® enables automated quality control on source code. The corporate code quality goals can be defined in a coding standard using QStudio® rules. The coding standard specifies which rules need to be applied and what their parameterizations are. The developer uses the QStudio® toolset to verify the source code against the conformance to the coding standard and, in the event of identified non-compliances, performs rework (guided by the observations, rule descriptions and patterns that QStudio provides).

QStudio specifies quality concepts in a measurable way based on an extended version of the ISO 9126 quality standard. QStudio recognizes quality attributes such as reliability, maintainability, testability, re-usability, portability and efficiency. The model defines a stepwise refinement of the notion of code quality into a set of ISO defined quality attributes and from these a further breakdown into quality sub-attributes. QA Systems proprietary technology maps these attributes in turn onto programming constructs.



User Definable and User Customizable Rules

QStudio for Java default supports over 200 rules some of which can be used to instantiate new customized rules. QStudio also allows users to add their own rules thus in practice therefore, the number of rules supported is unlimited.
QStudio for Java supports user rule definitions via the open-source PMD rule specifications (allowing:

- QStudio users to define their own rules via the PMD specification
- QStudio users to tap into the PMD user community and make use of the various rule sets being defined within the PMD community
- PMD users to seamlessly extend their rule sets with the extended ISO 9126 quality model by importing them into QStudio

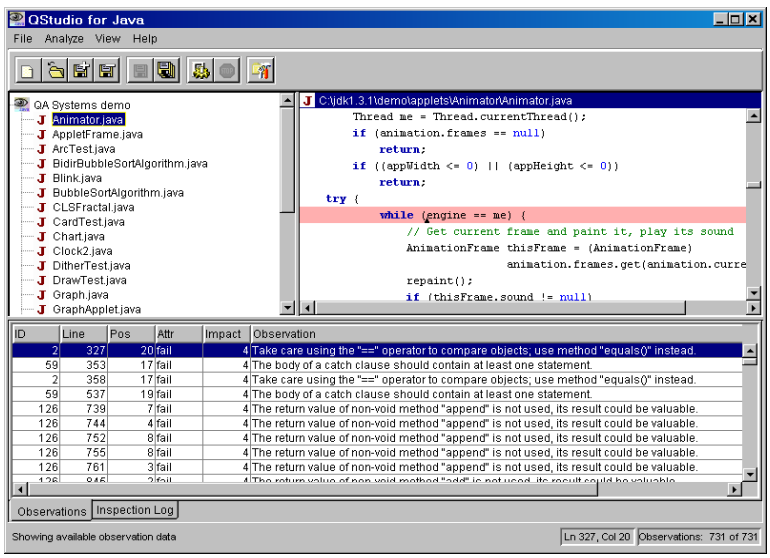
QStudio for Java has advanced rule customizability capabilities including:

- Rule configuration e.g. upper and lower boundary values, selectable scope and field modification and
- Rules instantiation for which a value can be entered at runtime
- Rules for which a regular expression can be entered as value (e.g. for naming conventions)

Project and File Level Code Inspection

Code Inspections can be performed per project, per package (all source files within a package-node) or per source file. During an inspection run the Java source code is checked on all the rules as configured in the checks configuration.

On rule non-compliance detection an observation is generated stating which rule was violated, the reason of the violation and the location where the observation was made.



QStudio for Java has the ability to analyze incomplete source bases. This is a particularly powerful feature since it means that analysis can take place at the subproject level providing support for very large projects.

When the analyzer hits files missing declaration information that cannot be resolved due to missing files/classes, it determines which rules can still be applied despite the missing information and applies those only. In practice this means that up to 80% of the rules can still be applied despite incomplete source trees.

Annotated Source Code Generation

A powerful feature is the automated annotation of source code allowing easy review of code.

```
//Title: QStudio JAVA
//Copyright: Copyright (c) 1999-2001 QA Systems Technologies B.V.
//Company: QA Systems Technologies B.V.
//Description:

package com.qasystems.io;

import java.io.File;
import java.io.Serializable;
import java.util.Date;

/**
(242) Do not write documentation comments that exceed position 70. ( Style Conformance )
 * This interface provides all functionallity required to load and store data
 * from and to a file.
 *
 * @author Sweder Schellens
 * @version %full_filespec: AbstractDataFile.java,9:java:1 %
 */
(20) Method "toString()" not implemented for class "AbstractDataFile". ( Modularity )
(49) Maximum ratio public/private class members exceeded (3.25 > 3.00) for class "AbstractDataFile". ( Structuredness )
public abstract class AbstractDataFile
  implements DataFile, Cloneable, Serializable
{
  /**
   * Sets the java.io.File used to read the data.
   *
   * @param file the java.io.File to be used for read actions
   */
(175) Avoid declaring method "setInputFile" synchronized . ( Modularity )
  public synchronized void setInputFile(File file) {
    inputFile = file;
  }

  /**
   * Gets the java.io.File used to read the data.
   *
   * @return the java.io.File to be used for read actions
   */
  public File getInputFile() {
```

Integrated Development Environments

QStudio for Java seamlessly integrates with the following IDE's: JBuilder™, Oracle® 9i JDeveloper, Eclipse™, WebSphere Studio® and Visual Age® (IntelliJ, NetBeans and Sun ONE planned later in 2003).

QStudio® for Java is available for Windows (98/2000/NT/XP/ME), Linux (RedHat Linux 6.1 and higher, SuSE Linux 7.0 and higher) and Solaris (Solaris 6.1 and higher

Enterprise Extensions

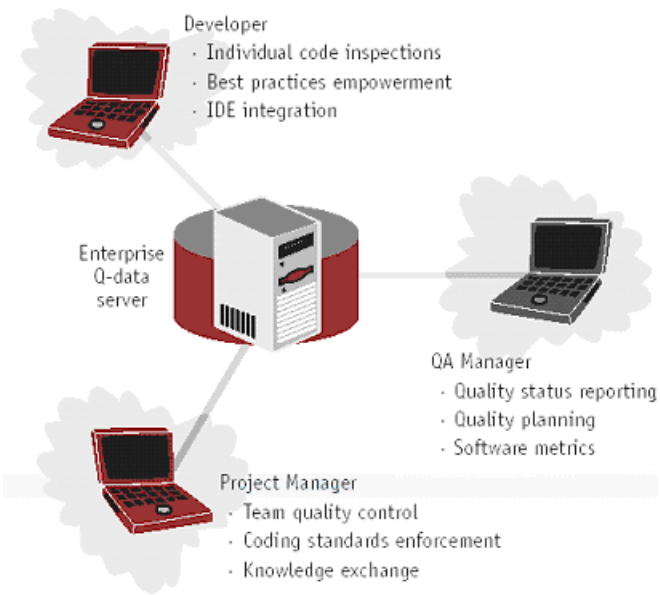
QStudio for Java Enterprise automates team/departmental based enterprise quality control as opposed to the single user approach supported through QStudio for Java Pro.

The Enterprise version is role based distinguishing software engineers, quality managers, project managers and application managers. Each role is granted different (related) functionality.

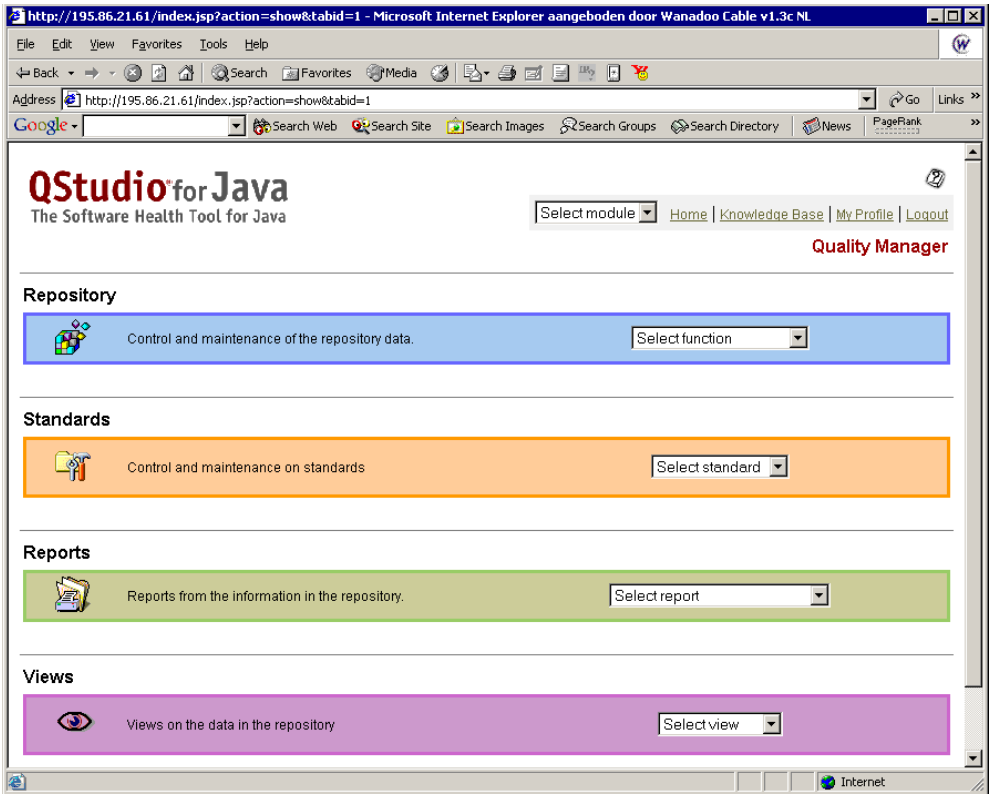
Multiple coding standards and quality standards configurations can be managed and made available to the software development team.

Formal inspections are supported. A formal inspection creates a milestone in the quality repository.

Prior to an inspection, a coding standard to inspect against can be selected from the server. Quality analysis data is stored in the quality data repository in order to facilitate



advanced reporting and to measure quality evolution and quality conformance checking.



The quality process manager can define an organizational structure and select which standard needs to be applied against which project, team or department.

Formal milestone generation is supported. Milestone data can be compared to measure quality evolution for quality trend analysis.

Various views and reports on the captured quality data can be generated.

Visualizations and Drill Downs

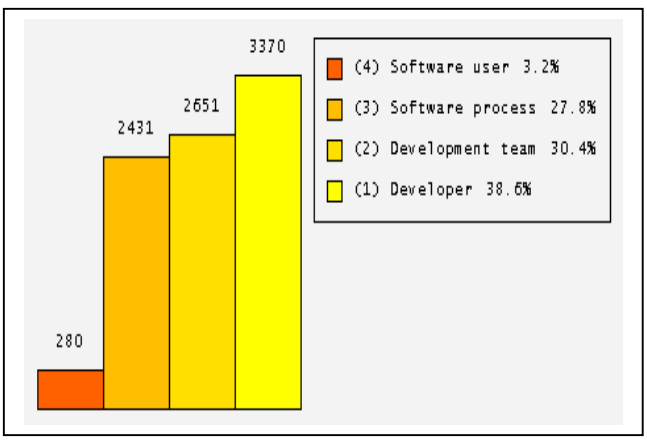
The Quality Data Repository can be examined from multiple viewpoints.

Chart, bar or table oriented reports can be generated giving a high level overview of the quality status of the inspected software source tree.

The displayed charts are clickable to drill down to more detailed information.

Impact Level

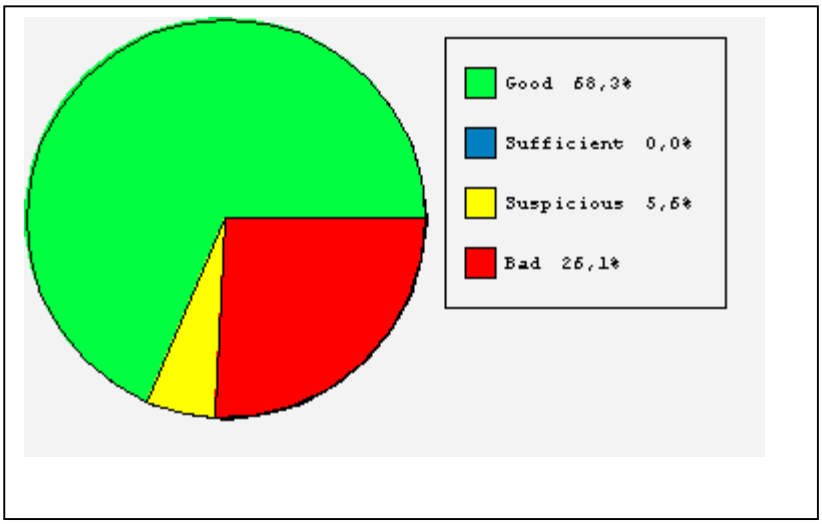
QStudio diagnoses 5 impact levels indicating the severity of a (potential) software defect. For example, impact level 5 (software failure) the highest) is the highest level and points to a risk that the product fails operation. Software defects at this level may cause the application to stop show no response.



Ratings

QStudio supports a ratings mechanism (Good, Sufficient, Suspicious and Bad). The ratings-mechanism provides a valuation for the acceptable number of observations (for each category). The lower the value, the better the code quality (on the aspect expressed by the metric).

A value of 68.3% for “Good”, for example, indicates that 68.3% of the files in the source package identified fully satisfy the compliance criterion for that particular metric. QStudio does not only provide the metrics at full project level but at any sub-project level. This powerful feature allows reporting of quality metrics at various sub-project (package) levels.



This visual shows the percentage of files satisfying (and not satisfying) an Impact Level 4 rating. Impact level 4 (Software User) points to the possibility that user assumed quality of the software is not satisfied due to a risk that parts of functionality cannot be used or that basic product features such as performance, resource behavior, user friendliness or accuracy are not within acceptable limits.

Quality Attributes

The following quality attributes are defined according to the ISO 9126 Software Quality Model.

Reliability: The ability of a software product to keep operating over time without failures that renders the system unusable. Observations generated on reliability indicate a risk that the application will fail during actual use.

Maintainability: The aptitude of the source code to undergo repair and evolution. Observations generated on maintainability indicate that changes are hard to implement.

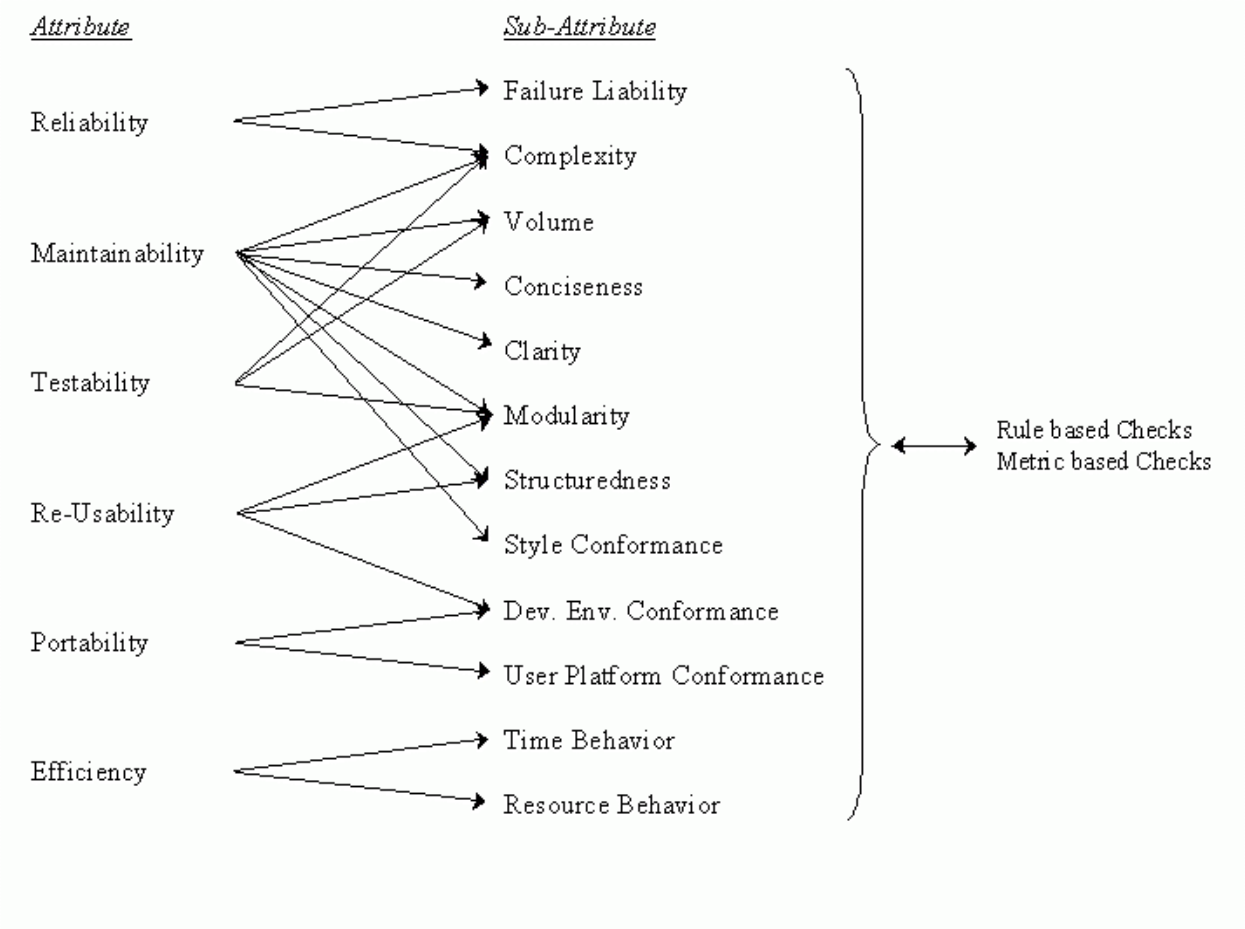
Testability: The amount of test resources needed to reach acceptable test coverage. Observations generated on testability indicate that a great number of test cases might be needed to reach acceptable test coverage.

Re-usability: The suitableness of the source code to be used by a variety of users. Re-use is the practice of using parts of source code that already have been developed. Observations generated on re-usability indicate that re-use is limited or difficult.

Portability: The ability of the source code to be used on various user environments and development environments. Observations generated on Portability indicate risks that the software product can't be used on specific user platforms or that the source code can't be used "as is" in specific development environments.

Efficiency: The ability of the software product to perform its functions related to the amount of resources that are used by the application. Observations generated on efficiency indicate that resources can be more effectively used.

The figure below shows the relationship between quality attributes and sub-attributes.

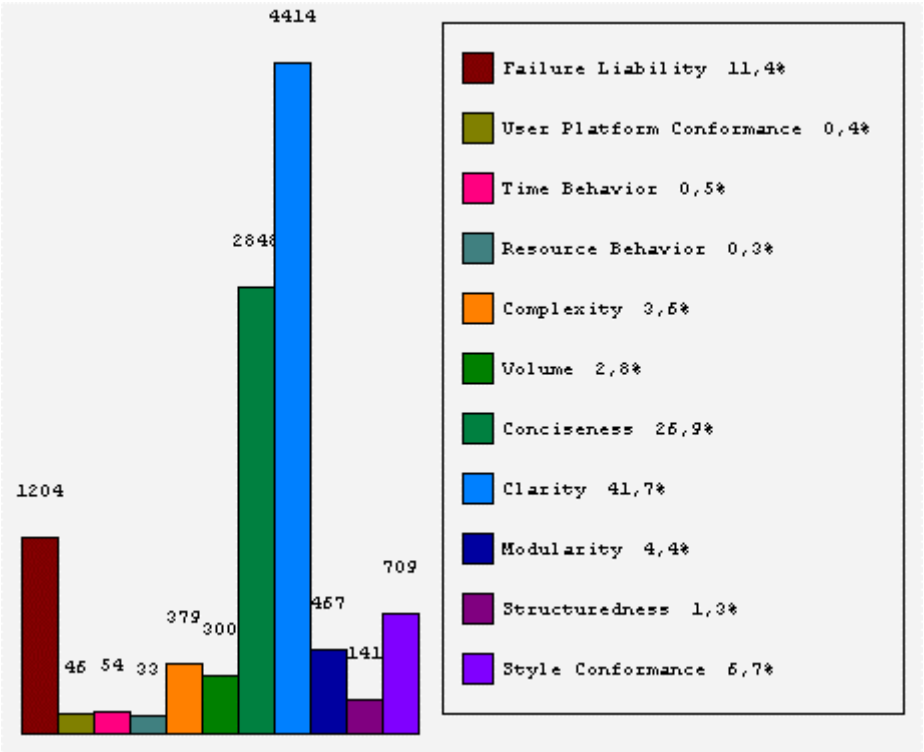


Quality Sub-Attributes Distribution

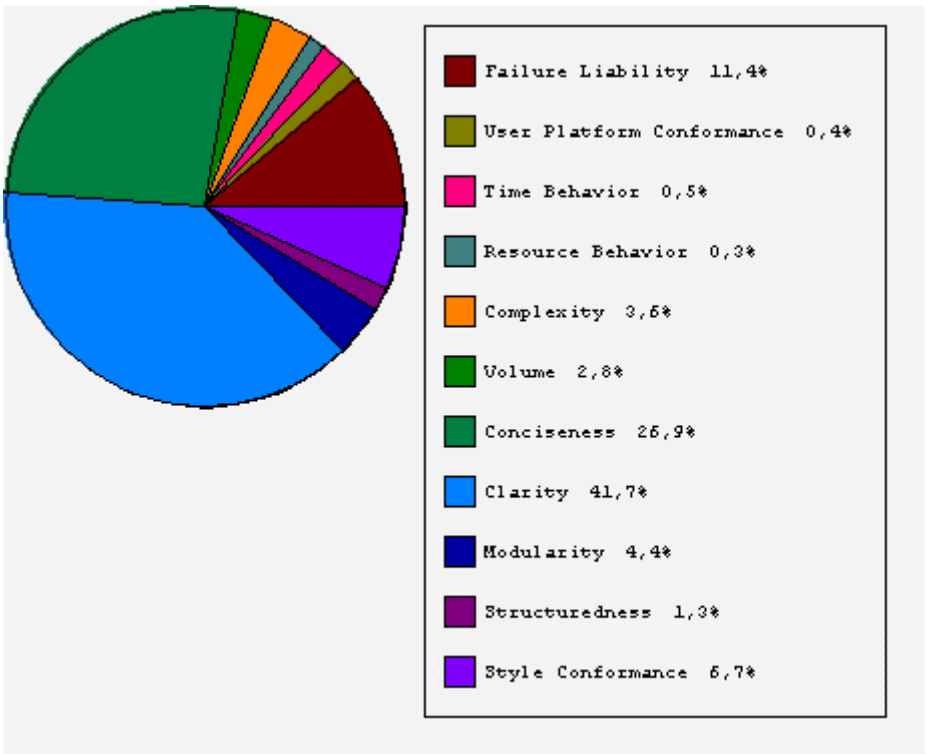
The absolute distribution in the project of quality sub-attributes is as follows. The numbers above the columns refer to the number of times an observation of the relevant type was made in the source code base. The figure shows a breakdown of non-compliance observations based on quality sub-attributes QStudio generated a total of over 10000 observations for this source code base. Rules and impact levels can be switched off to optimize important observations. Output is also filterable.

Observations can be grouped by file related, quality (sub)attribute, impact level or specific conformance violation.

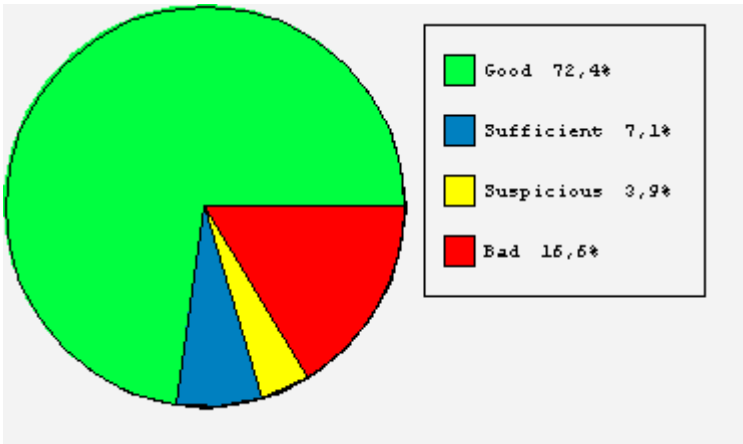
Trends analysis through formal milestone generation provides control over quality evolution at different levels: overall quality status, quality attributes or quality sub-attributes and metrics.



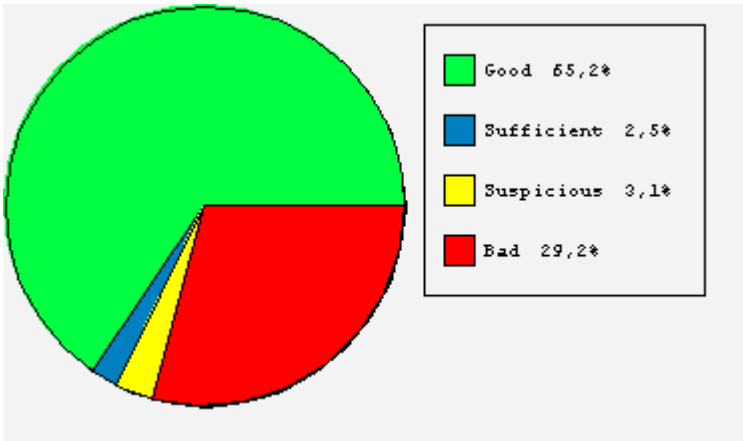
The relative distribution of on quality sub-attributes is as follows. The percentage numbers refers to the number of observations as a percentage of the total observations.



Example of metrics visual: average score over file related metrics, class related metrics and method related metrics.



Example of quality sub-attribute visual: failure liability - the possibility that a product failure occurs as a result of the applied coding practice. Observations with this sub-attribute indicate that there is a risk that the software product will fail during use.



Pricing and Availability

QStudio® for Java Enterprise is available for Windows (98/2000/NT/XP/ME), Linux (RedHat Linux 6.1 and higher, SuSE Linux 7.0 and higher) and Solaris (Solaris 6.1 and higher). It is priced at $295/€295 for the client and $2950/€2950 for the server so a 10 client/single server configuration costs just $5900/€5900.

QStudio for Java Pro

QStudio for Java Pro is the single user version of the QStudio product family. QStudio for Java Pro is available at no cost to the developer community. Download from www.qa-systems.com.

QA Systems

QA Systems - The Software Health Company™ - is focused on improving its customers' software health. QA Systems develops software tools to assess, support, monitor and control the health (quality) of software applications developed by its customers both from both a preventative viewpoint as a diagnostic viewpoint.

www.qa-systems.com
info@qa-systems.com

QStudio for Java 2.0 Features

	Pro	Enterprise
Analysis Capabilities		
Intuitive Graphical User Interface with project organization and integrated editor	P	P
On-line descriptive rules and pattern guide including best practice recommendations	P	P
Advanced Java pattern based source code analysis at file, class and method level	P	P
Over 200 Default User Customizable Rules	P	P
Large Project Support (Incomplete Source Base Analysis Capabilities)	P	P
User Definable Rules based on PMD specifications	P	P
User Configurable Coding Standards	P	P
Coding standards conformance at project level	P	P
Coding standards conformance at team and departmental level		P
ISO 9126 based quality analysis	P	P
Impact Level Analysis	P	P
Configurable and extendable checks for naming conventions based on regular expressions	P	P
Enterprise Java Bean compliance support	P	P
Language based analysis including:	P	P
Thread coverage	P	P
Check for unresolved classes	P	P
Package level coverage	P	P
Sorted ordering of imported packages	P	P
Coverage of Inner classes	P	P
Inheritance Analysis	P	P
Exception Handling	P	P
Method level checks	P	P
Naming conventions	P	P
Interface implementation analysis	P	P
Metrics based analysis including:	P	P
Number of statements per method	P	P
Number of statements per class	P	P
Number of methods per class	P	P
Ratio private/public class members	P	P
Static Path count	P	P
Non-final fields per class	P	P
Max lines of code per method	P	P
Code density	P	P
Coupling	P	P
Code Nesting	P	P
Cyclomatic Complexity	P	P
Method Nesting	P	P
Inheritance depth	P	P
Lines of code metrics (LCOM)	P	P
Comment Density	P	P

Quality Data Mining capabilities		P
Software quality trend analysis based on formal milestone definition		P
Text and HTML based annotated source code reports	P	P
Output view filters on individual rules, impact level and quality (sub)attributes	P	P
Observations overview in sortable table format	P	
Multidimensional views of observations		P
Click through overviews of type Table, Pie Chart and Bar Chart		P
Hierarchical views on Quality Attributes including conformance status		P
Web based graphical views on metrics and observations distribution		P
Application usage reporting		P
Quality data export capabilities		P
Integration Capabilities		
Seamless integration with Borland JBuilder	P	P
Seamless integration with Oracle9i JDeveloper	P	P
Seamless integration with Eclipse.org Eclipse	P	P
Seamless integration with IBM WebSphere Studio	P	P
Seamless integration with IBM VisualAge for Java	P	P
Seamless integration with IDEA IntelliJ (later in 2003)	P	P
Seamless integration with NetBeans (later in 2003)	P	P
Seamless integration with Sun ONE (later in 2003)	P	P
Standalone Environment	P	P
Command Line Interface		P
Integratable with Code Version Control Systems		P
Department/Team based quality control and management		
Web enabled quality control and management		P
Absolute and relative quality measurement and control		P
Managing quality data for multiple projects and multiple products (software releases)		P
Collaboration among distributed development teams		P
Distributed Coding and Quality Standards Management		P
Storing and maintaining historical quality data based on milestone analysis		P
Role dependent and customizable user profiles		P
Platform support		
Windows 98,2000,NT,XP,ME	P	P
Solaris	P	P
Linux	P	P