QJ Pro

QJ Pro is a sophisticated code quality assessment and coding standards technology for Java development detecting potential software during development time. It significantly reducse code review effort by automating a portion of the code inspection process and coding standards enforcement. QJ Pro gives developers the ability to automatically assess the sortware on the basis of high level software quality concepts such as reliability, maintainability and efficiency.

QJ Pro gives the benefits of automated software code quality control:

- Cuts testing time due to earlier detection of (potential) software errors
- Significantly reduces review effort by automating a portion of the inspection process (adherence to coding standards, usage of best programming practices)
- Improves quality control and thereby code quality by directly supporting adherence to improved programming practices and corporate coding standards

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. QJ Pro supports an explicit quality model that can directly tie into an organization's quality processes.

QJ Pro enables automated quality control on source code. The corporate code quality goals can be defined in a coding standard using QJ Pro rules. The coding standard specifies which rules need to be applied and what their parameterizations are. The developer uses the QJ Pro 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 QJ Pro provides).

QJ Pro 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.

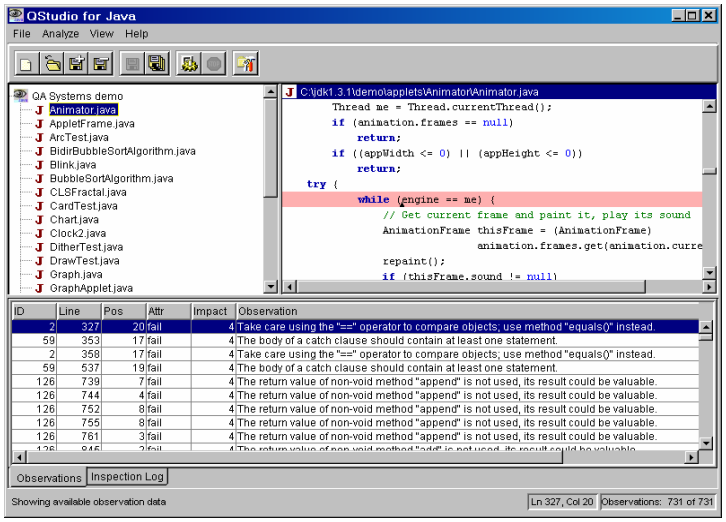
QJ Pro default supports over 200 rules some of which can be used to instantiate new customized rules.
QJ Pro has advanced rule customizability capabilities including:

- 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.



QJ Pro 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

QJ Pro seamlessly integrates with the following IDE's: JBuilder™, Oracle® 9i JDeveloper, Eclipse and WebSphere Studio®.

QJ Pro 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

QJ Pro is an open source project available at qjpro.sourceforge.net.

QJ Pro 2.1 Features

Analysis Capabilities	Pro
Intuitive Graphical User Interface with project organization and integrated editor	✓
On-line descriptive rules and pattern guide including best practice recommendations	✓
Advanced Java pattern based source code analysis at file, class and method level	✓
Over 200 Default User Customizable Rules	✓
Large Project Support (Incomplete Source Base Analysis Capabilities)	✓
User Configurable Coding Standards	✓
Coding standards conformance at project level	✓
Coding standards conformance at team and departmental level	▪
ISO 9126 based quality analysis	✓
Impact Level Analysis	✓
Configurable and extendable checks for naming conventions based on regular expressions	✓
Enterprise Java Bean compliance support	✓
Language based analysis including:	✓
Thread coverage	✓
Check for unresolved classes	✓
Package level coverage	✓
Sorted ordering of imported packages	✓
Coverage of Inner classes	✓
Inheritance Analysis	✓
Exception Handling	✓
Method level checks	✓
Naming conventions	✓
Interface implementation analysis	✓
Metrics based analysis including:	✓
Number of statements per method	✓
Number of statements per class	✓
Number of methods per class	✓
Ratio private/public class members	✓
Static Path count	✓
Non-final fields per class	✓
Max lines of code per method	✓
Code density	✓
Coupling	✓
Code Nesting	✓
Cyclomatic Complexity	✓
Method Nesting	✓
Inheritance depth	✓
Lines of code metrics (LCOM)	✓
Text and HTML based annotated source code reports	✓
Output view filters on individual rules, impact level and quality (sub)attributes	✓
Seamless integration with Borland JBuilder, Oracle9i JDeveloper, Eclipse, Websphere Studio	✓
Standalone Environment	✓
Windows 98,2000,NT,XP,ME, Solaris, Linux	✓

QJ Pro is an open source project. See qjpro.sourceforce.net