QualitySoftware

User Manual

Introduction

Due to continuous changes, software systems are subject to constant quality deterioration – software erosion – which increases development and testing costs as well as operational risks. The maintenance of custom-developed software throughout its entire life cycle often exceeds the initial development costs many times over. For organizations operating a large number of custom-developed systems, mitigating the effects of erosion is crucial. The risks arising from the deterioration in quality of software systems that are functionally sound are borne by the party ordering the development and operating the delivered system. Without an objective evaluation of the quality of the source code, the quality of the delivered system can only be assessed ased on the results of acceptance tests, which characterize only the functionality of the system but do not provide an indication of its maintainability and operating costs.

Measuring source code maintainability

One of the cornerstones of measurement is the so-called benchmark database, which contains the source code characteristics of a large number of software systems. The benchmark database serves as a basis for comparison for the software systems to be evaluated. Using the same reference database, the maintainability of different systems or multiple versions of the same system can be compared, and changes in maintainability can be tracked over time.

Another important element in measuring source code maintainability is the maintainability model, which is a directed, acyclic graph consisting of nodes representing low- and high-level characteristics (see Figure 1). Low-level characteristics ─ the vertices of the graph without incoming edges (sensor nodes) ─ metrics that can be calculated directly from the source code using static analysis (e.g., source code element complexity, ratio of serious coding errors, ratio of code duplication, etc.). High-level features ─ vertices with incoming edges in the graph (aggregate nodes) ─ the source (e.g., maintainability, changeability, etc.). The edges between the vertices of the graph represent dependencies. The model allows for expert weighting of the edges of the graph, thereby addressing the uncertainty arising from the subjectivity of concepts in maintainability calculations. Each weighting determined by an expert is considered one vote, and the probability distribution of the votes cast is recorded on each edge, thus expressing the opinion of each expert.

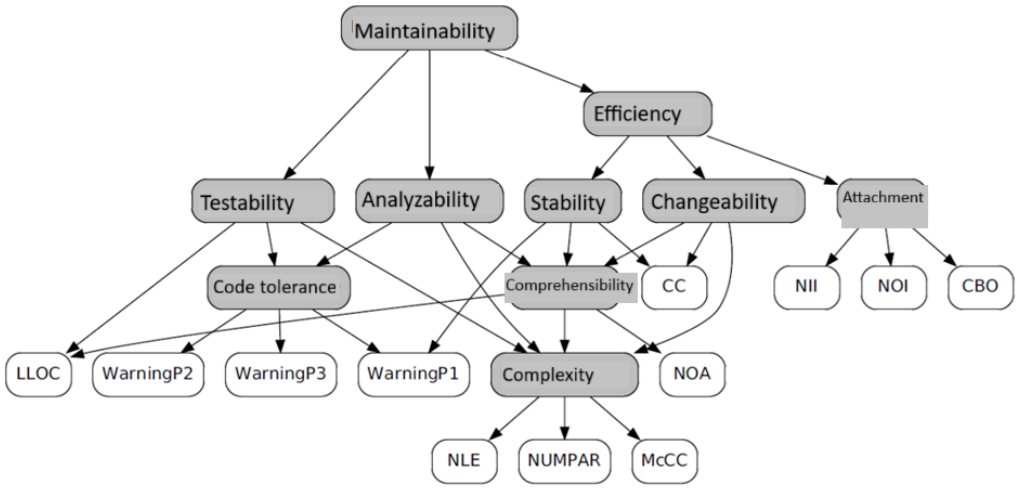

Figure 1. Source code maintainability model

The final building block of maintainability measurement is the evaluation method itself. In the case of the software system to be evaluated, the low-level source code characteristics included in the model are first generated. This is followed by a comparison with each of the software systems included in the reference database. The result of the comparison is a probability distribution function for each low-level feature defined in the model. Subsequently, along the edges of the model, the calculated probability distributions ─ based on the vote distributions assigned to the edges based on the vote distributions assigned to the edges ─ are aggregated using statistical methods, which serve as an objective measure for higher-level characteristics. By repeating the process several times, the root of the graph, which is Maintainability, can also be measured.

The complete theoretical background for measuring source code maintainability can be found in a publication that describes the basics of measurement in detail.

The relationship between source code maintainability and development costs

The importance of source code maintainability stems from its obvious connection to software development costs. Maintainability is most often defined as the resource requirements associated with modifying the behavior of software.

Based on the developed cost model, source code maintainability is an exponential function of the resources spent on development, i.e., the resources spent on development and the rate of

software erosion are exponentially related. The developed model allows the estimated development costs to be expressed as a percentage relative to a system with average maintainability. Figure 2 represents the development costs of systems with different maintainability values.

The complete theoretical background of the relationship between source code maintainability and development costs can be found in the publication laying down the foundations of the cost model.

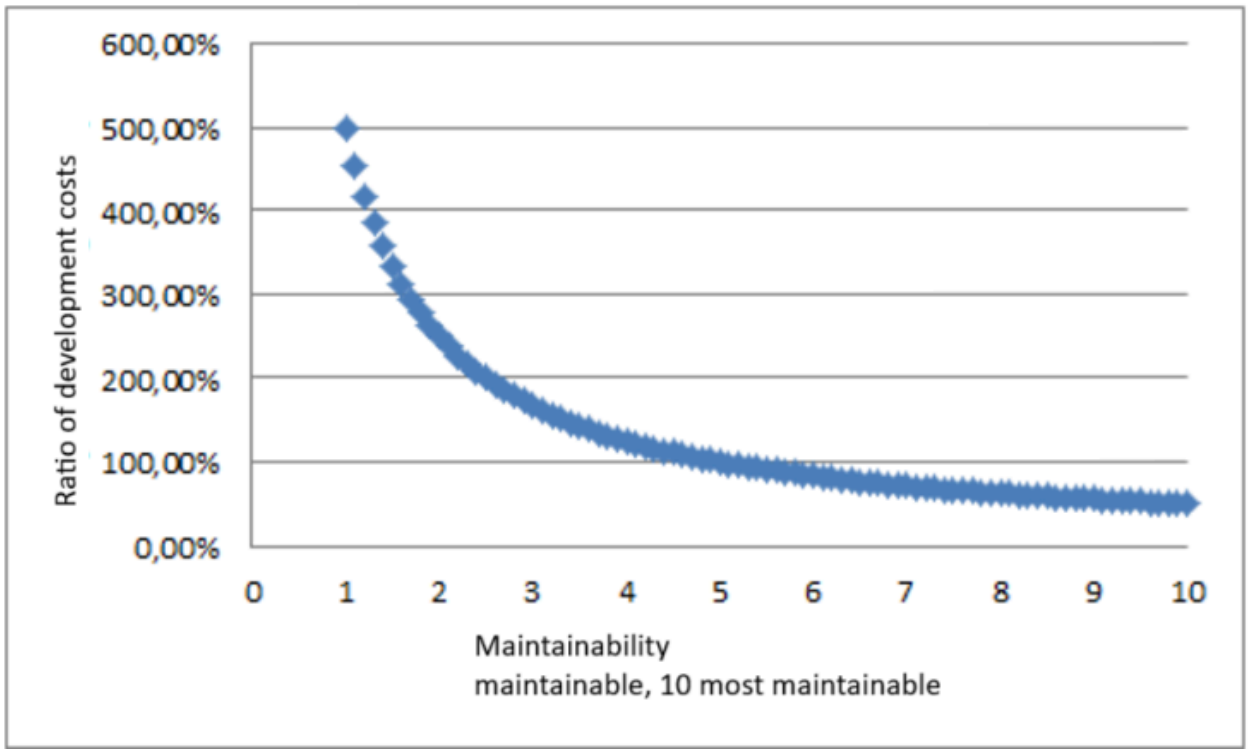

Figure 1. The relationship between maintainability and development costs

The QualitySoftware concept

What is QualitySoftware?

QualitySoftware is a source code quality assurance framework developed by XYZ Ltd. based on the ISO/IEC-9126 standard.

- QualitySoftware is a management tool that allows you to easily assess the quality of software development.
- QualitySoftware measures and monitors the maintainability of software by analyzing source code.
- QualitySoftware Holistically depicts the evolution of source code quality over time.
- QualitySoftware facilitates the management of software erosion.

- QualitySoftware makes recommendations in cases of excessive deterioration in source code maintainability.
- With QualitySoftware, code quality and developer team performance can be measured and
- improved.
- QualitySoftware provides an overview of expected maintenance costs.

Who are the users of QualitySoftware?

Software operating companies

- QualitySoftware stops poor-quality systems at the gate, preventing the influx of substandard code.
- When purchasing software, QualitySoftware allows customers to compare the quality of the source code of systems delivered by different suppliers, providing them with an additional decision-making criterion.
- QualitySoftware provides operators with meaningful results even if they lack software development knowledge and expertise.
- QualitySoftware users include financial institutions, insurance companies, technology companies, telecommunications companies, representatives of the energy sector, and many others who typically develop and operate custom (non-off-the-shelf) software systems.

Software development companies

- Software development companies can use QualitySoftware to measure and monitor the quality of their own developments. With this information, they can take the necessary steps before the developed product is released, preventing damage to the company's reputation and spiraling maintenance costs later on.
- Software development companies can also monitor the performance of their own processes by providing feedback on the development of source code maintainability.
- With QualitySoftware, software development companies can advertise the real-time development of the quality of their own systems on their websites, thereby supporting their sales.

How does QualitySoftware work?

QualitySoftware is installed together with Jenkins, an open source tool that supports continuous integration. QualitySoftware's own Jenkins plugin controls the entire analysis process as follows:

- Regularly checks whether there is a newer, as yet unanalyzed source code version in the version tracking system;
- It launches static code analysis of a given version of the code base, which is performed within the framework by the XYZ CodeAnalyzer product;

- During source code analysis, source code metrics, coding violations, and code duplicates are calculated;
- The results of the analysis are uploaded to the QualitySoftware central database;
- The high-level characteristics of the source code, including maintainability, are calculated according to the appropriate models;
- The QualitySoftware user interface allows you to view the data, create various reports, and investigate the reasons for changes in code quality.

Product features

- Supported programming languages: Java
- Integration with other tools: Jenkins, SVN, LDAP, PMD, XYZ CodeAnalyzer
- Report generation: PDF (management level), Excel (developer level)
- Supported platforms: Windows/Linux (32-bit/64-bit)
- Supported application servers: Glassfish
- Supported databases: PostgreSQL
- Maintainability calculation: probabilistic, reference system-based
- Drill-down approach: maintainability index calculated for source code elements
- Customizability: custom reference database, possibility to create custom models
- Cost model: based on source code maintainability
- User interfaces: Web UI
- Source code characteristics: source code metrics, coding violations, code duplication

Installing QualitySoftware

The QualitySoftware system should ideally be **installed and operated on a dedicated server**.

System Requirements

Minimum Requirements

- Required CPU cores: **4**
- Required memory: **8 GB**
- Required storage: **50 GB** (may vary depending on project size)

Recommended (Optimal) Requirements

- Recommended CPU cores: **8**

- Recommended memory: **24 GB**
- Recommended storage: **100 GB** (may vary depending on project size)

Supported Operating Systems

QualitySoftware supports the following operating systems:

- Windows x86/x64
- Windows 7
- Windows Server 2008 (R2)
- Linux x86/x64 (Debian-based)
- Ubuntu 12.04 LTS
- Ubuntu 12.10

Prerequisites

Before installing QualitySoftware, several prerequisite applications must be installed and configured.

1. Oracle Java JDK 1.7.0_10 (or newer)
    - Install the 32- or 64-bit version corresponding to your OS.
    - Download from: https://www.oracle.com/technetwork/java/javase/downloads/index.html
    - After installation:
        - Set the `JAVA_HOME` environment variable to the JDK directory.
        - Add `JAVA_HOME/bin` to the system `PATH`.
2. LaTeX / MiKTeX (version 2.9.x)
    - For Windows: download from https://miktex.org/download

For supported Linux systems: install via

```
sudo apt-get install texlive-full
```

3. Graphviz (version 2.26.3 for Linux, 2.28 for Windows)
    - For Windows: download from http://www.graphviz.org/Download.php

For Linux:

```
sudo apt-get install graphviz
```

4. PostgreSQL 9.1 database server
    - Download from https://www.postgresql.org/download/

Or install on Linux via:

```
sudo apt-get install postgresql
```

    - After installation, it's advisable to **tune the default settings** for better performance.
        - Configuration file location:
            - Linux: `/etc/postgresql/[version]/main/postgresql.conf`
            - Windows: `C:\Program Files\PostgreSQL\[version]\data\postgresql.conf`
        - Key parameters to adjust:
            - `shared_buffers` = 25% of available system memory
              (Linux users must also adjust kernel memory settings via `/etc/sysctl.conf`)
            - `effective_cache_size` = 50% of available memory (remove `#` comment mark).

5. PostgreSQL JDBC driver
    - Download from https://jdbc.postgresql.org/download.html
    - The installer includes version 9.1 by default.
      If a different database version is installed, replace the `.jar` file located in

`APPLICATION/DATABASE/POSTGRESQL/DRIVER` with the appropriate version. Only **one .jar file** should exist in that directory.

Additional PostgreSQL tuning tips can be found at:
http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

Installation Configuration

Installation parameters are configured in the file:

`install/scripts/configuration.properties`

Each parameter is defined as a **key-value pair**.

Important: Do not include extra whitespace (e.g., spaces or tabs) after property values!

Main configuration parameters:

- `xyz-admin-user` *(optional)* – GlassFish admin user for XYZ domain (default: `admin`)
- `xyz-admin-password` *(optional)* – Password for above user (default: `admin`)
- `jenkins-admin-user` *(optional)* – Admin user for Jenkins domain (default: `admin`)
- `jenkins-admin-password` *(optional)* – Password for above user (default: `admin`)
- `postgres-port` *(required)* – Database server port (default: `5432`)
- `postgres-user` *(required)* – Database username (default: `postgres`)
- `postgres-password` *(required)* – Password for the above user (default: `postgres`)
- `postgres-db-name` *(optional)* – Database name for QualitySoftware (default: `xyz_db`)
- `xyz-instance-port` *(optional)* – Port for XYZ domain (default: `10000`)
- `xyz-admin-port` *(optional)* – Port for XYZ admin console (default: `10001`)
- `xyz-java-heap` *(required)* – Memory allocated to XYZ domain in MB (default: `8192`)

- `jenkins-home` *(optional)* – Jenkins workspace directory (default: inside installer directory)
- `jenkins-instance-port` *(optional)* – Jenkins port (default: `20000`)
- `jenkins-admin-port` *(optional)* – Jenkins admin port (default: `20001`)
- `jenkins-java-heap` *(optional)* – Jenkins memory allocation (default: `512`)
- `rmi-url` *(optional)* – RMI address used by QualitySoftware services (default: `localhost:1198`)
- `toolchain-max-memory` *(required)* – Max memory for XYZ CodeAnalyzer (default: `4096` MB)

Critical notes:

- Verify all **required parameters** before installation.
- Ensure **valid paths** with write permissions for the installing user.
- Avoid using ports already occupied by other services.
- The **total memory assigned** to QualitySoftware components should not exceed **75% of total system memory**.
- Open the ports defined for `xyz-instance-port` and `jenkins-instance-port` in the firewall if you want to access the interfaces remotely.

Installation Process

After installing prerequisites and setting configuration parameters, run the installer:

On **Windows**:

```
install.bat
```

On **Linux**:

```
sh install.sh
```

Installation typically takes **~40 minutes** on systems meeting the optimal requirements.

Do **not delete** the installation folder after completion — it contains files necessary for system operation.

After successful installation:

- The **XYZ ProductDashboard** is available at `http://[host]:[xyz-instance-port]/XYZ-ProductDashboard`
- The **QualitySoftware interface** is available at `http://[host]:[xyz-instance-port]/QualitySoftware`
- The **Jenkins interface** (for analysis control) is available at `http://[host]:[jenkins-instance-port]/jenkins`

License Installation

Before using QualitySoftware, install the purchased license files:

1. **QualitySoftware license file**
    - Upload via `http://[host]:[xyz-instance-port]/XYZ-ProductDashboard` using the **Browse** button.
    - Click **Activate license** to enable the tool.
    - Upon activation, the QualitySoftware icon will turn gray with a green background — you can then click it to access the main interface.
2. **Source code analysis controller license**
    - On the *XYZ CodeAnalyzer Plugin Job* configuration page, check if the license is valid.
    - If expired, upload a valid file via **Browse** → **Activate license**.
    - The analyzer verifies license validity before each run.
    - If invalid, the analysis fails (“Build failed”) and the job is disabled.
    - Once a valid license is installed, re-enable the job to continue analysis.

Configuring QualitySoftware Permissions

QualitySoftware can authenticate users through an **LDAP server**.
The connection settings are defined in the file:

`QualitySoftware-securityContext.xml`

This file is located inside the GlassFish application server directory created during installation:

`glassfish3/glassfish/domains/XyzDomain/lib/classes`

It can later be modified there if needed.

To enable proper operation, the following settings must be configured:

Enabling LDAP Authentication

To enable login authentication via LDAP on the QualitySoftware web interface, set the parameter:

```
<bean id="loginConfiguration"
      class="com.XYZ.QualitySoftware.security.LoginConfiguration">
    <property name="loginRequired" value="true"/>
</bean>
```

- If `loginRequired` = `true`, users must log in before accessing the interface.
- If `false`, no login is required, but the following functions become **unavailable**:
    - Creating, editing, or deleting benchmarks
    - Creating, editing, deleting, voting, or calibrating models
    - Starting or deleting evaluations
    - Setting thresholds for evaluated systems

LDAP Server Connection

Define the LDAP server connection string (the directory providing user data):

```
<bean id="contextSource"

class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
    <constructor-arg value="ldap://localhost:389/dc=XYZ,dc=com"/>
</bean>
```

LDAP Groups and Roles

Specify how LDAP groups correspond to QualitySoftware user roles:

```
<bean id="grantedAuthoritiesMapper"

class="com.XYZ.QualitySoftware.security.LdapGrantedAuthoritiesMapper">
    <constructor-arg value="user=USER"/>
</bean>
```

Note: Currently, QualitySoftware only supports the **USER** role.

User DN Pattern

Define how users are identified (Distinguished Name, DN) in the directory:

```
<bean
class="org.springframework.security.ldap.authentication.BindAuthenticator">
    <constructor-arg ref="contextSource"/>
    <property name="userDnPatterns">
        <list><value>uid={0},ou=People</value></list>
    </property>
</bean>
```

By default, users are identified by their `uid` in the `People` organizational unit.
Multiple DN patterns can be specified if necessary.

Group Configuration

Define how groups and their users are structured in LDAP:

```
<bean
class="org.springframework.security.ldap.userdetails.DefaultLdapAuthor
itiesPopulator">
    <constructor-arg ref="contextSource"/>
    <constructor-arg value="ou=Groups"/>
    <property name="groupSearchFilter" value="(uniqueMember={0})"/>
    <property name="groupRoleAttribute" value="cn"/>
</bean>
```

By default:

- Groups are located under `ou=Groups`.
- Users are listed in the `uniqueMember` attribute.
- Group names are identified by the `cn` (common name) attribute.

These default settings are generally sufficient.

Configuring Jenkins Permissions

Jenkins provides its own authentication and authorization system.
To configure user access:

1. Open Jenkins at:
   `http://[host]:[jenkins-instance-port]/jenkins`
   (default port: 20000)
2. In the left menu, select **Manage Jenkins**.
3. On the right side, click **Configure Global Security**.
4. Check **Enable security** to activate security settings.

5. Under *Access Control → Security Realm*, choose **LDAP**.

6. Click **Advanced...** and enter the LDAP details (Server, root-DN, User search base, Group search base, Manager DN).

Possible Installation Errors

Uninstalling QualitySoftware

To remove QualitySoftware and its components completely, run one of the following from the `install` folder:

On **Windows**:

```
uninstall.bat
```

On **Linux**:

```
sh uninstall.sh
```

The uninstallation is successful if:

- The database specified by `[postgres-db-name]` is deleted (or did not exist).
- The GlassFish domains installed during setup are no longer running.
- The `glassfish3` directory in the installer folder is deleted.

Starting and Stopping the System

After installation, the XYZ products and Jenkins start automatically.

To **stop** them manually:

On **Windows**:

```
stop.bat
```

On **Linux**:

```
sh stop.sh
```

To **start** them again:

On **Windows**:

```
start.bat
```

On **Linux**:

```
sh start.sh
```

Controlling Source Code Analysis

After installing QualitySoftware, source code analysis is controlled through the **Jenkins interface**, available at:

```
http://[HOST]:[PORT]/jenkins/
```

where `[HOST]` is the computer name (or IP address) where the system was installed, and `[PORT]` is the port number assigned to the Jenkins domain (see installation configuration).

When the user logs into Jenkins, the main interface appears as shown in the figure in the original manual.

Starting Source Code Analysis

The **XYZ CodeAnalyzer Plugin** is a Jenkins plugin that enables **continuous analysis of software systems' source code**.

Each system's analysis is managed through a **dedicated job**.
To analyze a system, you must create one job per system, as follows:

1. In Jenkins, select **"New Job"** from the main menu.

2. Enter the project name and choose **"Xyz CodeAnalyzer Plugin"** as the job type.

3. Verify that the license for the analysis tools is valid.

    - If expired, the message **"License expired"** appears.

    - Use **Browse** to upload a valid license file, then click **Activate license file**.

4. Selecting the **"Upload as benchmark job"** option marks the job as a **benchmark system**.

    - Benchmark systems cannot be evaluated or versioned — only one version may be uploaded.

5. On the job configuration page, select the **models** to be used for evaluation.

6. You can specify which directories should or should not be analyzed using the **Use filter** option.

Example:
If you want to analyze everything under
`svn.apache.org/repos/asf/logging/log4j/trunk/`
except for the `tests` folder, use a filter file like:

```
/
-svn.apache.org/repos/asf/logging/log4j/trunk/tests/
```

    - The first line (`/`) includes all subdirectories, and the line starting with `-` excludes the given path.

7. **Version control selection:**

    - **SVN:**
      i. Enter the repository username and password if required.
      ii. List the source-code paths to analyze — each on a new line.
      Jenkins will automatically detect available revision numbers.
      iii. In the **Select SVN revisions** field, specify revisions as either:
      - A comma-separated list (e.g., `120,125,130`), or

- A range (e.g., `1-12345` or `1-HEAD`).
Using `HEAD` means that new revisions will automatically be analyzed as they appear.

- **Local file system:**
If the codebase is local, choose the **Directory** option.
i. Specify the local path on the Jenkins host machine.
ii. Provide a version number.
- If the directory name includes a timestamp (format: `Version_YYYY_MM_DD_HH_MM_SS`), Jenkins fills it automatically.
- Otherwise, you must enter it manually (e.g., `3426_2013_02_12_14_25_19`).

8. If you want **automatic report generation** after each analysis (PDF and XLS), check **Generate reports**.

Default report directory:

```
${JENKINS_HOME}\jobs\${JOBNAME}\workspace\reports\
```

-
- The reports will be listed in table form on the job's page after successful analyses.

9. **Discard Old Builds**:
Enables automatic deletion of older Jenkins build data according to the selected policy.

   - Under *Log Rotation strategy*, specify the maximum number of days or builds to retain.

   - Recommended for performance when running many analyses — older build logs slow Jenkins down.

10. Save settings.

11. Click **Build now** to start the analysis and evaluation.

    - This runs analysis for one version at a time.

    - To enable **continuous analysis** of multiple versions, create a **MultiJob** (see below).

Creating a MultiJob

To run multiple analyses continuously or in parallel, create a **MultiJob** project:

1. In Jenkins, choose **New Job**.
2. Enter a name and select **MultiJob Project**.
3. Under **Build periodically**, define a **cron-style schedule** (e.g., CRON expression documentation):
    - Every minute: `* * * * *`
    - Every 20 minutes: `0,20,40 * * * *`
    - Every midnight: `0 0 * * *`
4. Add analysis jobs under **Add build step** → **MultiJob Phase**.
    - Provide a phase name and the name of each analysis job to execute.
    - Add more jobs via **Add jobs...**
5. Save settings.
6. The MultiJob will automatically start the defined jobs according to the schedule.

For **parallel analysis**, set the number of executors under
Manage Jenkins → **Configure System** → **# of executors**
to exceed the number of parallel analyses desired.

- Recommended: 4 (one for the MultiJob itself, three for analyses).

Monitoring Analyses

All analyses controlled by Jenkins can be tracked at:

http://[HOST]:[PORT]/jenkins/

The dashboard displays:

Ongoing analyses

Latest successful/failed builds

Duration of the last run for each job

Clicking a specific job opens its details.

The left menu lists previous and current executions.

Selecting one and choosing Console output shows the execution log, including tools run and any errors.

These logs are essential for diagnosing failed analyses.

Example Console Output
Started by user anonymous
Building in workspace jenkins_home/jobs/Sample_Job/workspace
Actual revision is: 308873
922 revisions more.
Collecting source to workspace/src directory...
Sources were successfully collected!
[scripts] $ ant -file toolchain.xml ...
BUILD SUCCESSFUL
Total time: 7 seconds
Finished: SUCCESS

Pausing and Restarting Analyses

To **pause** a job:

- Click **Disable Project** on the job page.
  The system will stop launching new analyses.

If a job fails during execution, it is automatically disabled.
After resolving the issue, click **Enable Project** to resume normal operation.

Modifying Settings

You can modify a job's configuration using **Configure** on its page.
Changes take effect starting with the next execution.

Once a job has successfully run at least once, only some settings can be edited — for example:

- **Quality model:** cannot be changed after a successful run.
- **Use filter:** can be changed; doing so may significantly affect results.
- **Version control settings:** SVN credentials can be updated, and branches can be added or deleted (but not edited directly).

Deleting Jobs

To delete a job:

- Click **Delete Project** on the job page.

This removes all settings and previous analysis results.

Warning: Deleting a job also removes all related data from the QualitySoftware database — they will no longer be available in the web interface.

Additional Information

Further documentation on the Jenkins framework, configuration, and usage can be found at:
https://jenkins.io/ (formerly jenkins-ci.org)

Error Messages

Common messages and their meanings:

Message	Meaning / Solution
Database is not available.	QualitySoftware is unreachable from Jenkins. Check whether the application server is running. Restart it if necessary.

This field is required.	A mandatory field is empty.
[PATH] does not exist.	The specified directory cannot be found — verify or create it.
Please check the SVN URLs.	No or incorrect SVN path specified — fix or add a valid one.
Revision [REVISION] is not in the repository.	Listed revision numbers do not exist — correct the list.
There should be at least one real revision.	The specified revision range contains no valid versions (e.g., use `1–HEAD`).
Wrong Date format.	Date fields have an invalid format — see on-screen help.
Year must be between 1970 and [current year].	Enter a valid year.
Minute/Second must be between 0–59.	Enter valid time values.
The date should be earlier than now.	Future dates are invalid.

Using QualitySoftware

After successful installation and licensing, QualitySoftware can be accessed through a web browser at:

`http://[HOST]:[PORT]/QualitySoftware`

where

- `[HOST]` is the computer name or IP address where QualitySoftware was installed, and
- `[PORT]` is the instance port number defined in the configuration file (default: 10000).

The login screen appears first (if authentication is enabled).
Users can then navigate the main interface, which consists of several **main modules**.

The Main Interface

The main interface is divided into several **tabs** and functional **modules**, providing access to all features related to analysis, benchmarking, and reporting.

The primary tabs are:

1. **Evaluations** – viewing and comparing maintainability results.
2. **Benchmarks** – managing benchmark systems used for comparison.
3. **Models** – managing quality models and weight distributions.
4. **Calibration** – refining the maintainability model based on expert feedback.
5. **Reports** – generating, viewing, and exporting reports.
6. **Administration** – user and system configuration (available to administrators only).

Each tab offers detailed tables, filters, and charts for in-depth inspection.

Evaluations

The **Evaluations** module displays all systems analyzed using the QualitySoftware tool.

Each entry represents one evaluated version of a software system.
For each evaluation, the following information is shown:

- **System name**
- **Version number**
- **Evaluation date**
- **Maintainability value** (on a 0–10 scale)
- **Relative maintainability category** (Excellent, Good, Moderate, Poor, Very poor)

Sorting and Filtering

Users can sort by any column and filter results by:

- System name
- Date range
- Maintainability category

This allows easy comparison of project maintainability across multiple time points.

Maintainability Graph

Clicking a system name opens its detailed **Maintainability Graph**, which visualizes the maintainability trend across versions.

Each point on the graph represents one version.
Hovering over a point displays detailed metrics (e.g., LOC, complexity, rule violations).

The color of the curve reflects the maintainability category:

- Green → Excellent
- Light green → Good
- Yellow → Moderate
- Orange → Poor
- Red → Very poor

This view helps identify where source-code quality degradation (software erosion) has occurred.

Benchmarks

A **benchmark** is a reference software system used for comparison.
Benchmark systems are selected from among the systems analyzed by the tool.

Adding a Benchmark

To add a benchmark:

1. Navigate to the **Benchmarks** tab.
2. Click **Add new benchmark**.

3. Enter the benchmark name and description.
4. Choose which systems and versions should be included.
5. Click **Save**.

Benchmarks are stored in the QualitySoftware database and become available for all maintainability comparisons

Models

The **Models** module manages the maintainability model — a directed acyclic graph representing dependencies between code attributes and abstract quality concepts.

Viewing the Model

The default view shows the model's structure:

- Low-level nodes (source-code metrics) at the bottom.
- High-level nodes (abstract attributes such as "Testability" or "Changeability") above them.
- The top node is **Maintainability**.

Arrows indicate dependency relationships between nodes.

Editing Models

Authorized users can:

- Add or remove nodes.
- Define dependencies (edges).
- Adjust weights between attributes.

Weights represent how strongly one attribute influences another (based on expert opinions).

Calibration

Calibration adjusts the maintainability model based on aggregated expert feedback.

Expert Voting

Experts can provide their assessments on the importance of relationships between attributes (e.g., how much “Code Complexity” affects “Changeability”).
Each vote modifies the probability distribution of edge weights in the model.

Results of Calibration

After calibration, the maintainability model reflects a statistically aggregated expert consensus.
This ensures that future evaluations better align with real-world developer experience.

Reports

QualitySoftware can generate **automated reports** on system maintainability.

Report Types

1. **Management Report (PDF)**
    - Summarized visual overview of project maintainability.
    - Includes charts, maintainability trend, and cost estimations.
    - Suitable for decision-makers and non-technical stakeholders.
2. **Developer Report (Excel)**
    - Detailed tables of metrics, rule violations, duplications, and complexity measures.
    - Helps developers identify problematic components in the source code.

Generating Reports

To generate reports:

1. Go to the **Reports** tab.
2. Select a project and version.
3. Choose the desired report type (PDF or XLS).
4. Click **Generate report**.

Reports are created using the installed **LaTeX** and **Graphviz** tools and can be downloaded from the interface.

Administration

The **Administration** tab (visible only to admin users) allows configuration of:

- User accounts and access permissions.
- Connection settings for external systems (e.g., LDAP).
- System parameters such as database location, memory usage, and licensing.

Admins can also define **thresholds** for acceptable maintainability levels.
If a project's maintainability drops below a defined threshold, QualitySoftware automatically flags it as a **critical risk**.

Interpreting Maintainability Values

Maintainability is expressed as a **numerical value between 0 and 10**, where:

- **10** = extremely easy to maintain,
- **0** = practically unmaintainable.

The thresholds for each category are:

Maintainability Value	Category
8.0 – 10.0	Excellent
6.0 – 7.9	Good

4.0 – 5.9	Moderate
2.0 – 3.9	Poor
0.0 – 1.9	Very poor

Projects below **4.0** typically require urgent corrective actions (refactoring, redesign, or code cleanup).

Visualizing Code Attributes

Each project's maintainability can be further explored by inspecting its **code attributes**, such as:

- Code duplication ratio,
- Average method complexity,
- Number of coding-rule violations,
- Comment density,
- Ratio of test code to production code.

Interactive charts allow drilling down from high-level maintainability scores to individual file-level metrics.

Summary of Workflow

The general workflow when using QualitySoftware:

1. Jenkins runs source code analyses periodically.
2. Analysis results are uploaded to the QualitySoftware database.
3. Users access results through the web interface.
4. Benchmarks and models provide reference comparisons.

5. Maintainability is computed and visualized.

6. Reports are generated for management and developers.

7. Calibration ensures the system reflects expert consensus over time.

Support

XYZ Ltd. provides its customers with a dedicated interface for reporting errors and requesting support during the installation and use of the QualitySoftware system, which can be accessed at the following address:
https://support.XYZ.com
When an order is placed, a project and a user account are automatically created, and the parties concerned are notified by email.
The system allows the monitoring of reported errors and the progress of their resolution.
When an order is placed, a project and a user account are automatically created, and the parties concerned receive an email notification. The system allows the life cycle of reported errors to be tracked
from the moment they are reported until they are closed and a patch is released.

When creating a new bug report, please provide a detailed description of the bug and the sequence of events that led to it,
as well as a screenshot illustrating the issue. If possible, please attach the relevant log files in compressed form,
which can be found in the following locations:
$glassfish-rootdirectory/domains/$qg-domain/logs
$glassfish-rootdirectory/domains/$jenkins-domain/logs
Where $glassfish-rootdirectory is the root directory of the Glassfish server specified during installation, and
$qg-domain and $jenkins-domain are the names of the QualitySoftware and Jenkins domains specified during installation
.
After the error is reported, the support team at XYZ Ltd. is responsible for assessing the severity of the error,
notifying the relevant parties, and informing the customer who reported the error about the expected repair time. At the end of the
error reporting process, the customer will be notified of the release of the repair package, which can be downloaded from the same system.
Please send any questions regarding the use and operation of the QualitySoftware system to support@XYZ.com. Questions will be answered by the support team at XYZ Ltd.
who will provide all the information necessary for the smooth operation of the system.

Appendix

Description of source code metrics

List of coding violations

Description of code duplication metrics