Australian Government

Department of Immigration and Citizenship

Sanity4J

***A product developed by DIAC that allows Organisations to reduce* 'Technical Debt' resulting in lower software development costs.**

Java Competency Centre

October 2010

people our business

Sanity4J is an Open Source product developed by The Java Competency Centre of the Australian Department of Immigration and Citizenship (DIAC).

The Java Competency Centre is the centre of Excellence for Java development in DIAC. DIAC has developed a large number of complex Java applications over the last 11 years. These applications are comprised of internal and external Web based applications, Desktop applications, and SOA services.

Recent Awards include the eGoverment 2009 e-Award for Excellence for the Visa and Citizenship Wizard application, and a Highly Commended eGoverment 2009 e-Award for the eVisitor application.

DIAC developed Sanity4J based on its experience in Java development over the years, and using some of its most senior and experienced resources in the industry. It is a key tool used by DIAC to measure and improve the quality of it's Java applications, as well as lowering development costs and ongoing maintenance costs.

Technical Debt

Organisations that do not have mechanisms to measure the quality of code delivered by outsourcers and project teams, risk under-estimating project whole of life costs.

Hasty software development or cutting corners during project implementation can result in reduced code quality, and a lower quality product being delivered to production. This results in increased operational expenditure on maintenance in future years.

Deferring costs for technical activities to future years is sometimes referred to as generating ***Technical Debt***.

> deferring technical activities to future years only generates a technical debt and increases your costs

Drivers of Technical Debt

Project pressures often degrade the quality of the finished product, these can include:

- Profit motive or budget pressures
- Schedule pressures
- Implementation complications
- Requests for change

How Sanity4J reduces Technical Debt

An objective process for tracking quality, allows enforceable quality targets to be set and measured against. This reduces the ***Technical Debt*** that is created during project development: producing a higher quality product, reducing development costs, and ongoing maintenance costs. **Sanity4J provides such a process.**

Sanity Check for Java has been developed by the Department of Immigration and Citizenship (DIAC) to provide code quality measurement for Java code. DIAC has a large number of Java applications and Sanity4J is a key tool for improving application code quality. It helps to reduce Technical Debt, lowering development and ongoing maintenance costs.

Features And Benefits

What is Sanity4J?

Sanity4J is a code quality measurement tool for Java code developed by DIAC. It is:

- **Proven:** Targets can be written into Contracts and Deeds.
- **Measurable:** Scores are generated for all levels of the code hierarchy.
- **Actionable:** Developers are given advice on how to fix issues.
- **Easy to Use:** Easily integrates into the Software Development Lifecycle (SDLC).

Sanity4J measurements are useful for Stage Gate assessments which internal projects must pass to transition through various environments and ultimately into production.

Sanity4J Proactively Builds Quality

Using Sanity4J allows teams to build quality as they develop code, and provides an opportunity to make quality targets part of the development lifecycle.

It helps developers and managers to identify coding issues throughout the process, reducing defect rates and ongoing development costs.

Feature Summary

Code Quality

Increases Quality: Increases Java code quality in an application and increases developer programming skills over time.

Reduces Costs: Highlights application defects at coding time, thereby reducing development and ongoing maintenance costs.

Code Health Check: Provides an automated code level view of the health of an application.

Integration

Integrated: Developers can build-in quality as they go using ad-hoc quality reports. Managers can see reports when software builds happen.

Automated: Quality measurement can be run automatically and requires no ongoing effort after setup.

Standards

Industry-based: Java Coding Rules are based on popular Open Source Software.

Consistent: All Java projects for an organisation can be measured against the same quality standards, in a repeatable and consistent fashion.

Familiar: Developers navigate through Sanity4J data using the Javadoc presentation layout they are already familiar with.

Configuration

Configurable: Coding rules are prioritised by risk and impact. Priority can be modified to suit an organisation.

Extensible: Organisations can extend Sanity4J by adding their own coding rules.

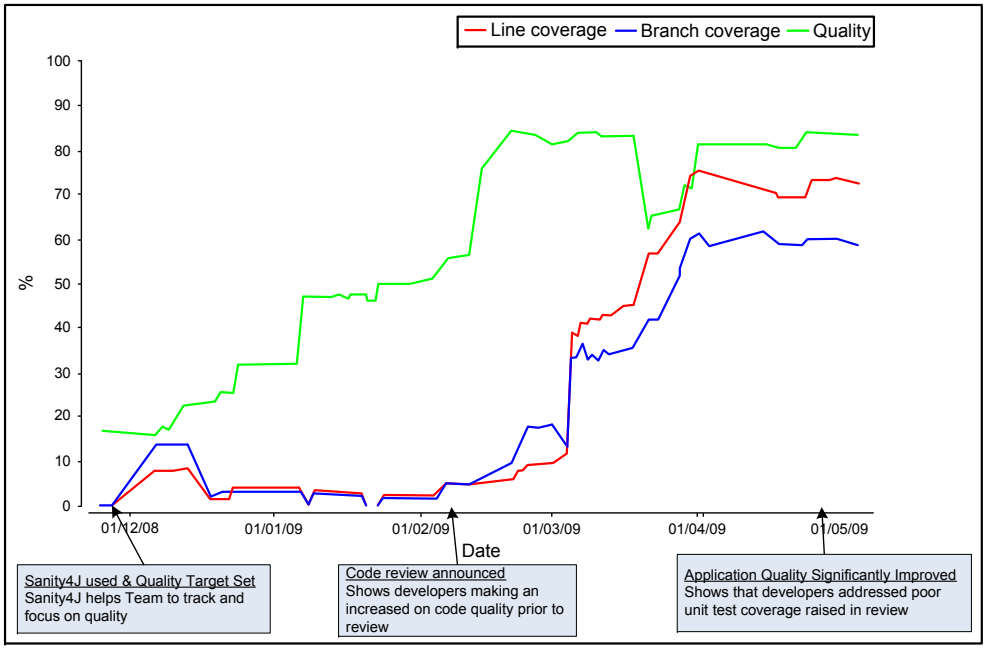
Reports

Management Reports: Provides easy to view graphical reports for both developers and management.

Case Study

How Does it Work in Practice?

The following diagram shows the graph produced by Sanity4J for a DIAC Portal Development Project. The project began by creating code quickly, with little regard for quality (**green line**). Sanity4J reports were configured for the project and an 85% quality target was set. Developers began to improve code quality by fixing issues as the project progressed. Towards the go-live date, the team was advised that their code would be subjected to an external review. This caused an increased focus on quality which Sanity4J tracked.



The external code review pointed out that the project had failed to reach Departmental targets for automated unit tests, represented by the **red** and **blue** lines. Prior to go-live, the project was able to dramatically increase their level of automated unit testing and their progress was tracked by Sanity4J.

The result was a higher quality product, which is estimated to cost much less to maintain in future years.

Monitoring Progress

The information that Sanity4J tracks was available to developers, project staff and the project executive throughout the development of the project. Sanity4J provided the ability to easily measure and track quality. ***It is unlikely that the project would have achieved such significant improvements without support from Sanity4J***.

Measuring and Reporting Quality

> at the core of Sanity4J is a rule base consisting of Java don’ts!

These rules are fundamentally sourced from four Open Source projects: FindBugs, PMD, PMD Copy & Paste Detector and Checkstyle.

Senior DIAC developers categorised and standardised these rule bases so that they could be aggregated and analysed as a group. Each rule was also assigned a severity rating.

Sanity4J allows organisations to modify severity ratings for individual rules. For example, some organisations may place increased significance on rules leading to increased memory or CPU utilisation.

Organisations are also able to write custom rules specific to their environment and experience. For example, an organisation may be aware of anti-patterns relating to how services are used. Sanity4J can warn developers if these rules are broken.

The number and severity of rule breaches are aggregated using a metric, which is compared against the size of the code base (measured in lines of code). This metric is expressed as a quality percentage (%).

Sanity4J tracks quality (%) over time in a graphical format so improvements and regressions are visible to management and developers.

Developers can drill down into specific quality issues, access information describing **the basis of the issue and potential remediation options**. This has proved valuable in increasing the skills of developers.

Measuring Automated Unit Test Coverage

Another factor in reducing project *technical debt* is the delivery of a suite of automated Java Unit tests (jUnit). Maintenance developers rely on these tests to ensure that they don’t break existing code when they make changes.

Sanity4J integrates results from the open source ‘Code Coverage’ tool Cobertura. Code Coverage is the proportion of code executed during a jUnit test run. Cobertura ‘watches’ while jUnit tests run, and records which lines of code get executed. It then calculates the percentage of lines covered.

A high Code Coverage % is another important indicator of quality. Sanity4J combines Code Coverage % and Quality % into a single graph which can be used to measure the performance of development teams on how they deliver quality code over time.

Introducing Sanity4J to Projects

DIAC has developed Sanity4J and has released it as Open Source Software licensed under the GNU General Public License.

In order to use Sanity4J:

1. Download Sanity4J from SourceForge, which is a public repository for Open Source Software.

   http://sanity4j.sourceforge.net/

2. Familiarise yourself with the Sanity4J configuration and usage documentation.
3. Configure Sanity4J to integrate into your Project Builds.
4. Share reports and analysis information within your organisation using your existing project wikis or Intranet sites.
5. DIAC provides support for Sanity4J via the SourceForge Forums and Bug reporting capabilities. Improvements to Sanity4J can be provided via SourceForge, which DIAC will review and integrate into future releases.

Introducing Sanity4J to Developers

As well as providing feedback after automated builds, Sanity4J includes a plug-in for the Eclipse Integrated Development Environment. This allows developers to quickly analyse individual Java classes.

By using the plug-in as they write code, junior developers can find and fix mistakes before peer review. This frees up senior developers from fixing simple ‘Java 101’ mistakes made by junior staff, allowing them to focus on implementation accuracy.