Squore

Industrial Analytics for Efficient Project Monitoring

What is Squore?

Squore is an efficient decision-making dashboard that enables quality mangement of project development by:

> Improving Project Performance
> Driving Software and Systems Quality
> Ensuring Process and Standard Compliance

Why Squore?

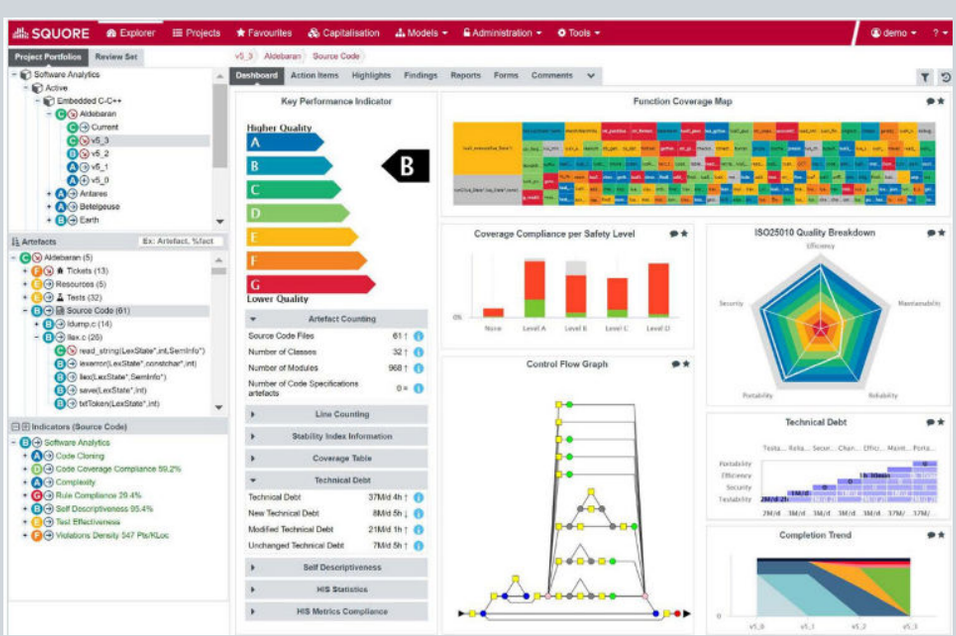
> Streamline monitoring and review process based on full traceability and industry proven indicators
> Share real time engineering data within a Business Intelligence framework
> Assess standard compliance using Squore Source Code Analyzer and external data sources
> Automate continuous quality checking and reporting, helping development teams benefit from agile/DevOps processes
> Manage application portfolio by easy comparison with other similar projects, enabling objective decision making

Highlights

> Aggregated View on your Projects
All along the project lifecycle, Squore Software Analytics automatically collects and reconciles results from the project's toolchain.
> Actionable Indicators
Breaking data silos, Squore delivers industry proven or corporate KPIs.
> Visualization and Reporting
Squore provides role-based navigation and sharing features (document generation, exports).
> Efficient Secision-Making
Squore generates an optimized action plan, helping you make an informed decision.

Benefits Overview

> Informed decision-making based on accurate indicators
> Continuous assessement with real-time dashboard
> Increased confidence on deliveries
> Improved reliability via early defect detection
> Reduced code review costs
> Lower maintenance costs via control of technical debt
> Broadcast of best practices and better process maturity
> Enhanced collaboration within project teams



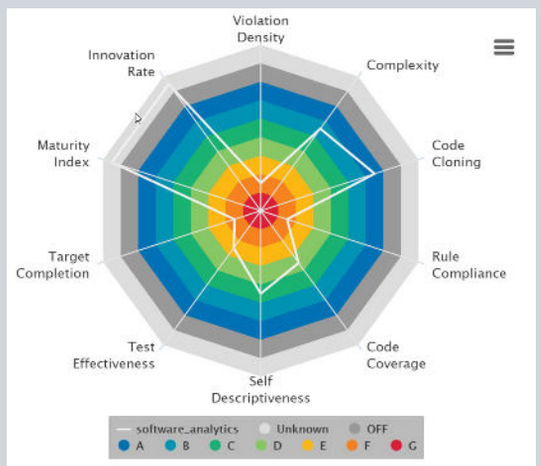
Squore dashboard: More transparency and faster decision-making by providing management-relevant information.

Application Areas

> **Project Mangement:** Efficient quality highlights of a project (based on specifications (requirements lifecycle and coverage), coding (style, complexity, documentation), tests (effectiveness, stability) and tickets (maturity index, innovation rate)
> **Test:** Optimization of test activities by focusing on critical components according to test strategy
> **Audit:** Template-based report generation
> **Development:** CI-compliant automated quality checking (including delta versioning analysis, standard compliance, etc.)

Squore in Action

> **Integration of third-party data**
Squore integrates results from source code, tests, tickets, design, requirements, etc.
> **Industry standards compliance**
Squore gathers essential indicators to monitor industrial software projects based on most common standards.
> **Data exploration**
Squore data historization allows trend analysis and version comparison.
> **Continuous Integration**
Connect Squore to your continuous integration and monitor reliable indicators in real time.



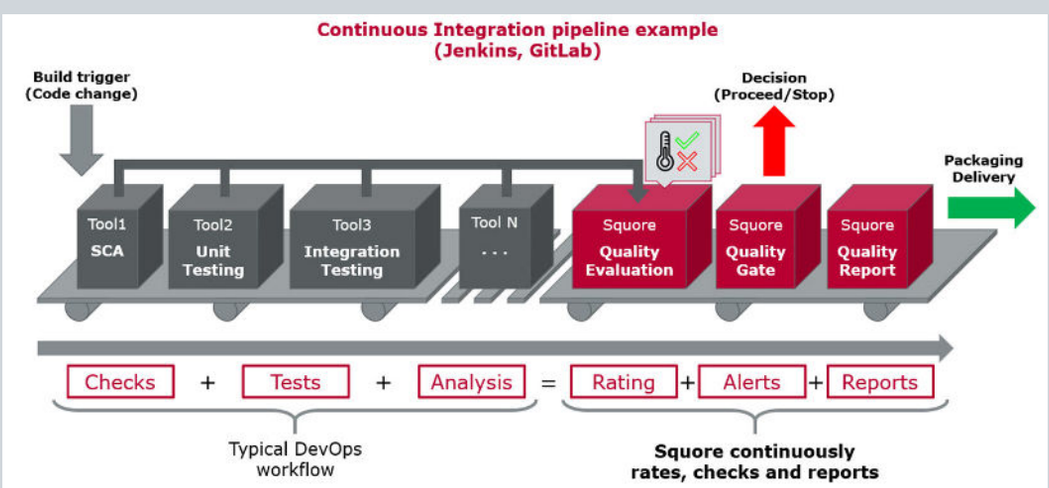
Analytics KPI breakdown

Squore Connectivity

Squore engineering platform provides:

> Ready-made interfaces to Vector tools: CANoe, vTESTstudio, Vector Connection Utility, VectorCAST, DiVa, etc.
> A toolbox of interfaces to third-party tools: VectorCAST, Cobertura, NCover, RTRT, Klocwork, CheckStyle, JaCoCo, QAC, PC-lint, FindBugs, Polyspace, StyleCop, Jira, Mantis, etc.
> An API to extend data source from third party: CSV, Excel, Json, ReqIf, Xml, Vector Trace Items, Database, REST API, etc.
> Embedded source code analyzers for several languages: C, C++, C#, Ada, Java, Python, etc.

More information: www.vector.com/squore



Typical Squore usage in an industrial production pipeline

V2.1 | 2022-2

VectorCAST/QA

Code Coverage and Test Automation

What is VectorCAST/QA?

VectorCAST/QA enables teams to implement consistent and repeatable processes for managing test activities and reporting key quality metrics such as code coverage. Test automation simplifies test execution. It is compatible with all test execution frameworks. VectorCAST/QA also provides an integrated workflow for white-box system testing. Application internals can be monitored, and faults injected using Probe Points. VectorCAST/QA Change-Based Testing reduces total test time by only running tests that are impacted by code changes.

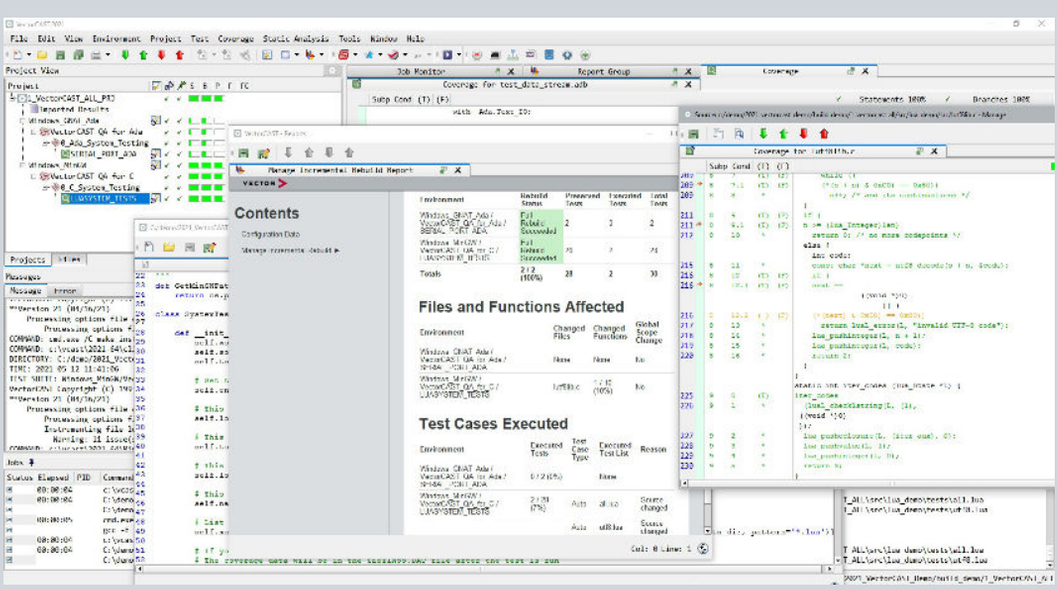
VectorCAST/QA allows team members to collaborate on test activities, shorten test times, and provide up to date metrics on release readiness.
VectorCAST/QA integrates with your build system and existing test infrastructure to silently collect key metrics such as, test case status and code coverage data.
VectorCAST/QA uses your existing workflow and tools so no changes are necessary.

Highlights of VectorCAST/QA in VectorCAST 2021

> Improvements to VectorCAST coverage environments for easier integration to CI workflow
> Integration with the static analysis tool Axivion Suite from Axivion

Overview of Advantages

> Supports C, C++ and Ada
> Certified by TÜV SÜD for industrial, automotive, medical and railway applications
> Tool Qualification Package for DO-178C
> Coverage types include Statement, Branch, MC/DC, Function and Function Call
> Uses the same compiler and build scripts you already use
> Coverage for host and embedded target applications
> Integrated workflow for White-box testing with Probe Points
> Faster testing with Change-Based Testing (CBT)
> White-box Testing with VectorCAST Probe Points
> Jenkins plug-in to support continuous and parallel testing



VectorCAST/QA system test automation and code coverage solution

Functions

> Creating a coverage environment
> Creating a test automation script
> Instrumenting for coverage
> Adding an interactive GUI for manual test steps
> Automating test runs
> Performing an incremental build and test execution with Change-Based Testing (CBT)
> Limiting coverage to a single software component
> Creating a Change Impact Report
> Using VectorCAST Covered by Analysis (CBA)
> Using the VectorCAST Probe Points for white-box testing
> Using the VectorCAST Jenkins plug-in

Augmenting Coverage Analysis with Covered By Analysis (CBA)

Covered By Analysis (CBA) allows users to augment test coverage metrics with coverage analysis data sets. Small portions of embedded applications are commonly impossible to test, but regulated industries require documented analysis of uncovered code to meet the requirement of 100% structural coverage.

VectorCAST CBA provides the ability to do code analysis directly within VectorCAST using the Coverage Analysis Editor and combines the test and analysis coverage metrics in a single report. VectorCAST CBA can also import analysis files, including those generated by third-party tools.

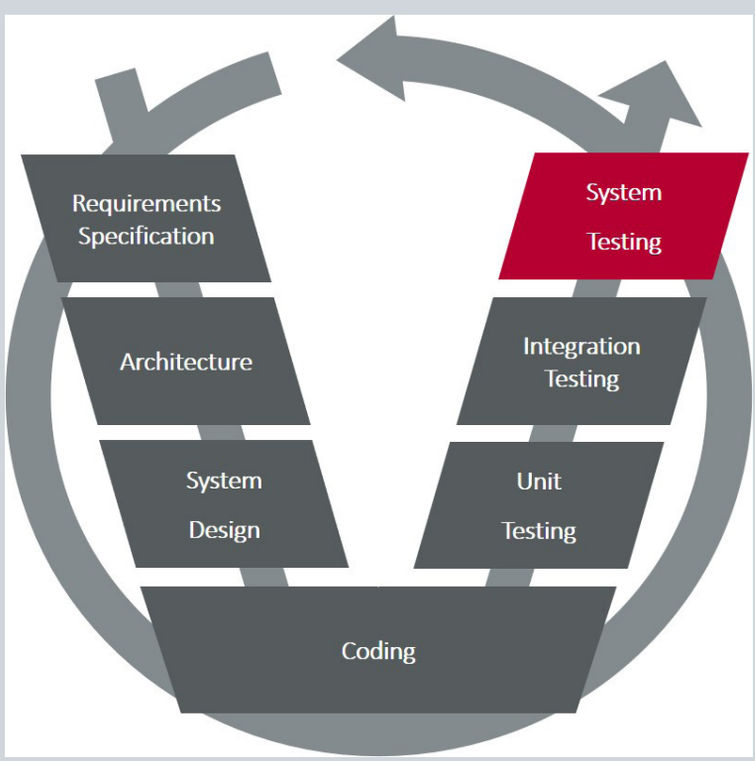
White-box Testing with VectorCAST Probe Points

VectorCAST Probe Points allow the user to insert user-defined blocks of code, or Probe Points, before or after any executable statement or function. It extends the test coverage by using fault injection to test error paths or to add test conditions that are difficult to setup at the application level. The Probe Points are maintained as the source code changes to the greatest extent possible. Probe Points are available for C and C++ code.

Probe Points are frequently used in the following testing scenarios:

> Capturing local variables during program execution
> Injecting spurious values to allow testing of error handling code
> Patching faulty code to test a fix prior to committing the change
> Debugging hard-to-trigger race conditions
> Recording detailed control flow
> Dynamically instrumenting device software to isolate defects

More information: www.vector.com/vectorcast



VectorCAST/QA is a key part of your overall software development cycle.

V2.1 | 2021-8

VectorCAST/C++

Unit and Integration Test for C/C++

What is VectorCAST/C++?

VectorCAST/C++ is an integrated software test solution that significantly reduces the time, effort, and cost associated with testing C/C++ software components necessary for validating enterprise, functional safety and high integrity systems.

Generally, software component testing requires generating at least one line of test code (in the form of stubs, drivers, and test data) for each line of application code to be tested at the unit or integration level.

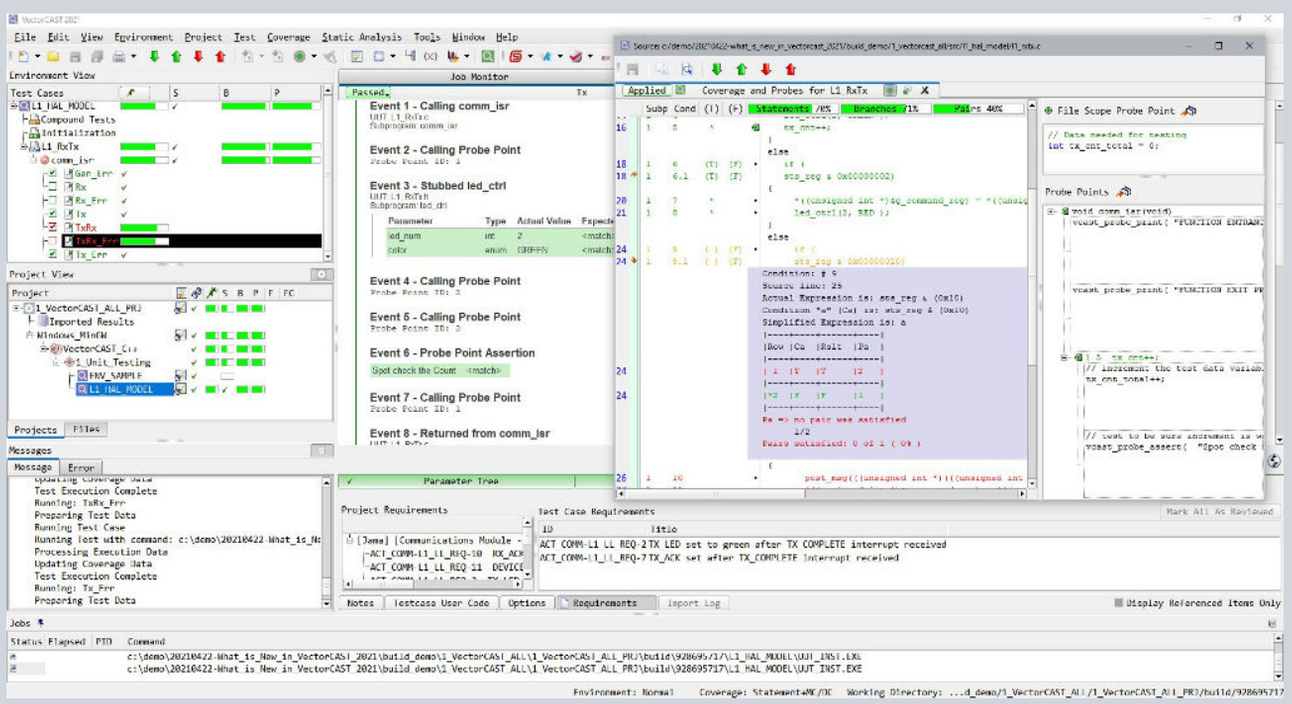
The necessity to create this "disposable" test software is the main reason why manual component testing is so expensive and inefficient. Test software not only has to be written but also has to be debugged to ensure that it performs as expected. With VectorCAST/C++, component testing at the unit or integration level can be performed without writing a single line of test code.

Highlights of VectorCAST/C++ in VectorCAST 2021

> Improved Requirements Linkage (RGW 3.0)
> Axivion Integration
> Improvements to VectorCAST Coverage Environments (Migrated Cover Environments)
> Python 2 to Python 3 Upgrade

Overview of Advantages

> Supports C++11, C++14 and C++17
> Headless command line operation to support CI/CT workflows
> Integrated Code Coverage capabilities, including MC/DC
> Tool Qualification package for DO-178C DO-178 (Avionics)
> Certified for use in ISO 26262 (Automotive), IEC 61508 (Industrial), IEC 62304 (Medical), EN 50128 and 50657 (Rail)
> Supports Agile and Test-Driven Development (TDD) Methods
> Pre-integrated to work with a wide range of compilers, simulators, and processor architectures



VectorCAST/C++
Integrated Software Test Solution

> User Configurable Compiler Interface
> Automatic test creation based on Classification Tree Method
> Complete test-harness construction for unit and integration testing
> Pre-integrations with leading requirements traceability tools
> Eliminates need to build test drivers and stubs manually
> Supports Host, Simulator, or Embedded Target Testing
> Automates Regression Testing

Functions

> Creating a Unit/Integration Test Environment
> Building a Test Case
> Executing Tests
> Using Code Coverage
> Using VectorCAST Probe Points
> Using VectorCAST Covered by Analysis
> Using the Static Analysis Interface
> Using the VectorCAST Requirements Gateway

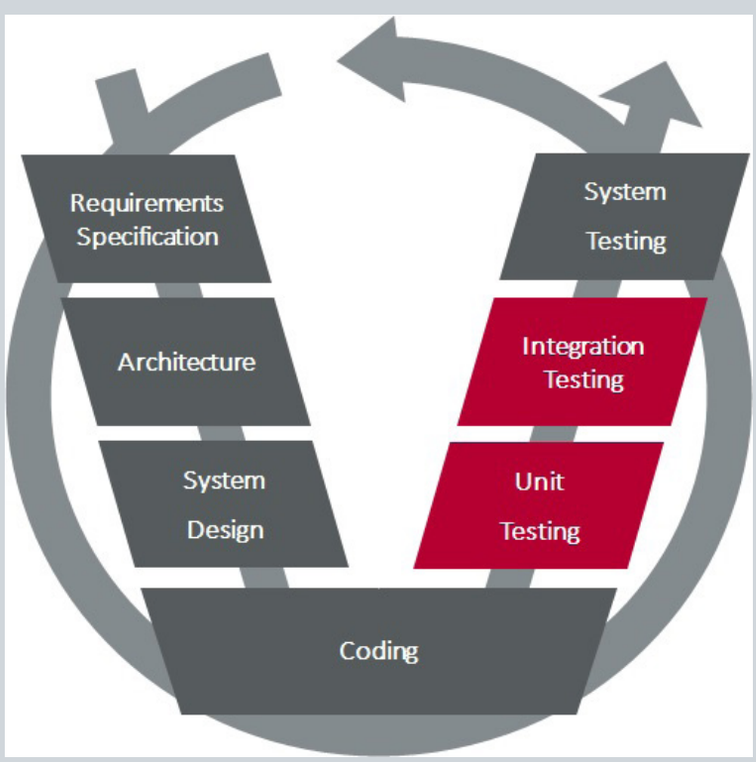
Integrated Code Coverage

Without a code coverage tool, it is difficult to determine which portions of the source code have been exercised during testing. VectorCAST/C++ provides an integrated code coverage utility that allows you to gauge the effectiveness of your component testing by reporting on the source code statements or decision points exercised during individual or multiple test runs. Code coverage data can also be shared with VectorCAST/QA to produce combined coverage reports that reflect unit, integration, and system testing.

Testing is Repeatable

Once test cases have been developed, you can use VectorCAST/C++ to automatically run test cases against successive versions of the source code. The management of test execution and the cataloging of test results are automatic. Comparing results of the same test cases against new software versions, prior to system integration, results in fewer surprises caused by "one small change" to a software component.

More information: www.vector.com/vectorcast



VectorCAST/C++ is a Key Part of your Overall Software Development Cycle

V2.2 | 2021-4