**A Guide To Using Continuous Integration Within The Verification Environment**

André Winkelmann, Jason Sprott, Gordon McGregor

---

**DIRECTED TOOLS**

CI (continuous integration), MDV (metric driven verification)

**CI (Jenkins)**

- Code health
- Keep main branch passing
- Release early & often
- Quickly spot and fix issues
- Integrated with SCM
- Automates test execution
- Communicates status
- PASS/FAIL dashboards

**MDV (e.g. vManager)**

- Finding RTL bugs
- Managing complexity
- Integrated with verification language features
- Verification closure
- Integrates with vplan
- Verification specific results
- Specialized analysis tools

**OBJECTIVES**

Provide integration of MDV with CI

**RESULTS**

Project Status in Jenkins

**IMPLEMENTATION**

- Jenkins
  - JUnit Plug-in
  - HTML Publisher Plug-in
- vManager
  - Regression Launch Script
  - Regression Control & Result Output
  - Export Per-Test Results (as CSV)
  - Export HTML Reports
  - Import to JUnit plug-in
  - Import to HTML Publisher plug-in

**CONCLUSIONS**

- CI and MDV are complementary
- Two Jenkins integration examples shown
  - Reusable: csv_to_junit_reporter.py
  - Ex. 1 (vManager) using a e-language API
    - Specific: initial launch script
    - Specific: e code to control exit and export results
  - Ex 2 (vManager CS) using TCL API
    - Specific: initial launch script
    - Specific: TCL to control exit and export results
- There are some integration requirements
  - Same results on Jenkins as standalone sims
  - Regression must run in batch mode
  - No return until all simulations complete
  - We can leverage standard Jenkins plug-ins

**REFERENCES**

Getting the code
https://bitbucket.org/verilab/jenkinsintegration

Full paper