How and why I used Rational Robot (VuC) to support high volume functional testing.

Mike Kelly

Key Points

- Performance test tools sometimes work better for functional testing then traditional functional test tools
- Performance test tools are better designed for high volume automation
- Performance test tools can sometimes offer easier access to functionality in application-under-test

Problem

- Rewrite of a legacy insurance application
  - From mainframe to web-based
- Parallel testing using 300-500 new applications/policies a week from production data
  - Written to flat file by production process every weekend

Traditional Approach – Using GUI

- It would take somewhere around 10 minutes per policy to enter the data
- A possible total execution time of around 80 hours on one machine.
- Even if we distributed the testing across 10 machines, it would take 8 hours (assuming no problems).
- Use all of our resources during that time.

Using Performance Scripts

- We had a more robust scripting language
  - Easier access to the production data
  - Allowed us to write more advanced data parsing and conversion methods
- We were no longer tied to the GUI
  - Cut execution time to around 45 seconds per policy
- We were able to leverage virtual user licenses instead of functional licenses
  - Allowed us to execute tests in batches of 100
  - We did not have to use all of our resources so we could continue working.
- Less impact of application changes (specifically GUI changes)

Prototype

- I got a prototype working before I left the company
- The only issue we ran into was a load limitation in the application-under-test (which we would not have found until much latter in the project otherwise)

Limitations

- No GUI level testing is executed (screen-based rules for data verification, consistent behavior, etc…).
- One of the reasons we wanted to use production data was not just to test the rating system for policies, but to also test the input constraints on the GUI.

Future Implementations

- I would consider putting in more elegant response time logging
- I would design the data access so it could be pointed to different data sources
- Going forward it would be nice to develop a balance between the GUI testing and high-volume testing.
  - Use the performance test tools for the bulk of the testing and then randomly select the data for some GUI level testing?
  - I would be open to discussions on how to work around this problem.