**Continuous build, test and deployment - the Mantid model**

Owen Arnold1, Martyn Gigg1, Stuart Campbell2, Peter Peterson2, Russell Taylor3, Nick Draper1

1 Tessella plc, Abingdon, Oxfordshire, UK

2 Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA

3 Tessella Inc, Boston, Massachusetts, USA

Mantid[1], our extensible framework for neutron and muon data reduction and analysis supports a large community of users across approximately thirty seven instruments at three facilities. At all facilities user beam-time is expensive and limited, so the software must work consistently, rapidly, and above all, provide the correct answers to a high degree of accuracy on every execution. Furthermore, our collaborating instrument scientists are often keen for us to quickly add new features to our existing code base, which is in excess of 1x106 lines of code. For this we need a team of twenty developers, situated at two facilities, 4000 miles and 5 hours apart. To compound issues, Mantid must build and deploy on a number of large range of completely different hardware and operating system environments. Continuous delivery of a project with these requirements is non-trivial.

Quality is best implemented from the ground up. Like many other open source projects, the Mantid project stipulates coding rules to keep the code consistent. However, the Mantid development also extensively uses Continuous Integration mythologies and tools; a strategy that gained momentum in the world of commercial software engineering. The benefits of this approach include very high levels of automation, which improves overall product quality and frees the scientists and developers to build exciting new features. Our tools include CMake, Git, Jenkins, CxxTest, Cppcheck, Gmock and Trac. We demonstrate using these tools to create a continuous integration suite capable of building, testing and deploying the Mantid product to all the platforms we support.

**References**

[1] www.mantidroject.org

Email corresponding author: owen.arnold@stfc.ac.uk Preference: Poster

Key theme: Best practice in software engineering