Continuous Build, Test and Deployment - the Mantid Model

Mantid, our extensible framework for neutron and muon data reduction and analysis supports a community of X users across approximately Y instruments at two locations. At both 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 T 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, and code-review to keep the code consistent and reduce errors and faults introduced by developers. 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 and GMock and Trac. We demonstrate our continuous build, test and development strategy using these tools.