Software
Sustainability
Institute

SSI/Mantid - Technical Review Executive Summary

Project Title	SSI/Mantid
Authorship	Steve Crouch
Document Version	0.1
Date	15/12/16
Distribution Classification	Project-Internal
Distribution List	*Project Management Board; Project Team*
Approval List	*Project Management Board*

Revision History

Date	Comment	Author	Version
15/12/16	Initial draft	STC	0.1

Executive summary

The Mantid project[1], a large international collaboration between the Science and Technology Facilities Council (STFC) and the US Department of Energy (DoE), provides a framework for the analysis of neutron and Muon data. It represents a collaboration between 4 of the major international neutron centres and the Tessella commercial company. The project has been running for 9 years, and a review of the collaboration, use and development around the software was requested from the Software Sustainability Institute to feed into a general review of the project's strategic science objectives, governance and software quality.

This report provides an analysis of results from a questionnaire sent to developers of Mantid across the facilities, which was open from the end of September to early November, and received a total of 36 responses. From those responses:

- A third of respondents identified themselves as junior software engineers, 22% as senior software engineers, 19% as team leaders, project managers or project management board members, and 11% as students
- Nearly half of respondents are resident at ISIS (47%), followed by ORNL (33%) and ILL (11%), with small numbers at ESS and 'Other' locations
- The greatest proportion of developers have been working on Mantid for 1-2 years (35%), followed by 5-10 years (30%), 3-5 years (19%) and less than 6 months (16%)
- Nearly half of respondents indicated they conduct most of their development work with Linux (Ubuntu being the most commonly specified), 35% with Windows, and 18% with Mac.

The software's development processes, practices, standards and infrastructure are generally seen as at least good. Governance is also seen as good, and developers consider the project as well led, and able to influence the direction of Mantid, although there is a call for more transparency and communication between developers and higher-level committees and boards. Communication across and within development teams is also seen as good, and developers feel they have a significant degree of freedom in prioritising activities, although there is a strong request for more direct contact with users and scientists. In such cases, face-to-face meetings are overwhelmingly seen as most efficient means of communication. In terms of the software itself, documentation is seen as good and generally up to date, and that understanding and accomplishing basic tasks is seen as easy. However, conceptual understanding of advanced tasks is seen as difficult.

The biggest issue highlighted across the questionnaire responses is software maintenance, and the variability of code quality across the application suite. Key areas for improvement include the huge number of backlogged issues and mounting technical debt which is seen by developers as an increasing problem. This aspect was also strongly observed in the *Collaboration Review*. These aspects should receive greater attention, and are often de-prioritised over local institutional demands and requirements. Greater acknowledgment, backing and drive from management is required to address these issues, and assigning effort across the facilities to accomplish this is suggested and should be considered.

[1] http://www.mantidproject.org

The general findings from analysing the responses are split across the following categories of development, processes, planning and infrastructure, governance and communication, and the software itself.

Development

- ***Overall, the code base is very hard to maintain, requiring substantial development effort, and incorporates significant technical debt***. Most notably, in terms of portability, consistent code design, third-party dependencies, tests and bugs, patches and new features. Overall, the variability of maintainability across the components is considerable. Dealing with new features requests, bug reports and questions on how to use Mantid are dealt with the most often. The Mantid team is currently considering better ways to make framework-level updates.
- ***In terms of understanding the code, there is strong agreement that the source code naming standards are clear and consistent with generic coding standards, and agreement that coding standards are enforced.*** Generally, there is agreement that source code structure relates clearly to the software design, although there is far less agreement that it's easy to understand individual source code files and how they fit into the implementation. There is generally sufficient documentation regarding the structure of the source code repository and how it maps to the software, although general documentation for new developers could be useful. The main barrier to learning how to develop with Mantid is that the code is not well structured nor commented in many places - there is a high variability of code quality and commenting. In addition, it is noted that there is a lack of deep understanding of key Mantid concepts across the team due to a low bus factor.
- ***There is agreement that the code is relatively straightforward to modify to address issues, modify functionality, and to a lesser extent add new features***, and that the project's contribution policy is well defined and publicly available. However, there isn't clear agreement that there is a well defined public stability or deprecation policy, and there is an inclination to disagree that the project's intermediate and long-term development goals are well defined. Reusing existing functionality or changing aspects at a low level can prove difficult.
- ***In terms of new feature development, generally users are engaged with the development process to ensure solutions meet their needs, and that user requirements are gathered and updated regularly.*** There is some measure of disagreement that there is a well defined and documented process of requirements capture, and that they are well defined, understood and communicated. Each new feature has a clear owner who is equipped with adequate knowledge, skills and tools to implement that feature. It's noted in the comments that it can sometimes prove difficult to engage users in the development process, which can be highly variable.
- ***In terms of portability, there is very strong agreement that Mantid should be supported across as many platforms as possible*** to support the needs of users. However, generally most indicate that this is not an easy process. Many problems encountered with regards to portability are due to peculiarities of certain platforms rather than Mantid design problems.

- ***There is strong agreement that there is clear testing policy adopted within the project, that it's clear what and when aspects should be tested, and general agreement that it's clear how aspects should be tested and what the automated tests are testing for.*** Bugs and other issues are by far most often found by users and developers, compared to beta testers. However, it's noted that the division between unit and system tests is often unclear, more tests should be based on real use cases and data where possible, and that testing of GUI aspects should be made easier.
- ***There is strong agreement that the project has well defined testing standards and that unit, system, and documentation tests are essential***, although are slightly expensive in terms of effort to maintain.

Processes, Planning and Infrastructure

- ***Regarding software development processes and practices, there is strong agreement that pull requests are effective, and that development, automatic builds and issue tracking are also proving effective.*** A key issue issued raised in the comments was that there is a lack of planning and developer effort to address outstanding issues and technical debt, with the greater share of effort assigned to new features. Also noted by several respondents was that the build process often suffers from a number of unrelated or false build errors.
- ***Regarding planning and development practices, there is agreement that there is a clear timeline for short term goals, but less clarity on medium and long term goals.*** There is also agreement that the development process is transparent and efficient, that the codebase is tested thoroughly prior to release, and very strong agreement that there is a clearly defined and adhered to release schedule. There is notable disagreement that there are well defined processes for treating stale development branches, and that codebase stability is prioritised over new features. It's noted that more effort should be assigned to improving complexity, stability and maintainability, and that these aspects should be given greater attention. A key suggestion is to have facilities assign a fraction of effort to 'core' development - aside from facility, instrument, and user needs - for maintenance of the core parts of the framework.
- ***The issue tracking practices are clearly documented, easy to find and sufficient, and issue resolutions are reviewed appropriately. Issues also have clearly defined owners.*** However, there is considerable disagreement that stale issues are dealt with appropriately, and some disagreement that there are well defined threshold criteria to escalate significant issues. The biggest issues noted by respondents are that there should be better issue management, particularly for prioritising the longer-term issues, and that greater effort should be made available to resolve them.
- ***In terms of the use of revision control, aspects of efficiency, the enabling of teamwork, and maintainability are generally seen as good to excellent.*** There is strong agreement in the comments that the adopted revision control practices do not pose a hindrance to development, and some suggestions that there should be a greater emphasis on reviewing open pull requests for efficiency, and using forks instead of branches, which would create a common workflow for both internal and external developers.

Governance and Communication

- ***For project governance, there is clear agreement that there is strong evidence of an active user community, the project identity is clear and unique, and it's easy to see who owns the software and project.*** It's also clear that it's easy to understand how the project is run and managed. There is also agreement that the user community is growing. There is also a clear level of satisfaction with the way the project is being lead, and the majority of respondents indicate they feel they have at least some influence on the future development of Mantid. A clear theme arising from a few of the comments is that there needs to be clearer transparency and communication in both directions between the development team and the higher-level boards and committees.
- ***Regarding project communication, developers rate their communications with other developers as good to excellent, and generally good with users.*** There is less satisfaction when communicating with management and other boards and committees. The greatest proportion of developers use Slack by far to keep track of Mantid's activities, and to a great extent, GitHub activities. Email is also commonly used, although use of the website and mailing lists for this is mixed. Forums are generally not used for this purpose. The vast majority think there are just enough regular team, yearly developer, and management meetings, and most think there are just enough face-to-face meetings and informal chats with other developers. A substantial number of developers would like more contact with users and instrument scientists. The comments also include a suggestion to reduce fixed meetings when not necessary.
- ***Likely factors causing coordination or communication problems within the project are different work priorities or background, and different staff locations, and to some extent, different time zones.*** When communication issues occur, generally the greatest proportion of respondents were at least mostly satisfied with how they were dealt with in terms of lessons learnt, procedure appropriateness, timeliness, efficiency, and the appropriateness of the procedure employed. A comment mentions that whilst inter-facility development team communication is often good, this is often not the case with instrument scientists, with missed opportunities for shared development work to avoid diverging codebases.
- ***Regarding work assignment, there is clear agreement that tasks fit well into developers' areas of expertise, and general agreement that tasks fit into their areas of interest and it's clear who is working on which aspect***. There is also agreement that tasks are mostly self-assigned, and tasks are well balanced for each developer. To some extent, human resources are seen as well managed. There is obvious agreement that what is expected of developers is clear, and that they have adequate tools and skills to complete tasks using adequate procedures and processes. There is also clear agreement that developers have sufficient training opportunities, a creative working environment, and generally realistic timescales. It's noted that developers are given a very free reign at the moment to prioritise requirements, since they work closely with users and scientists. Comments also include having developers at facilities other than ESS assigned to work on the core framework, and that training opportunities differ across the facilities. It's also noted

that there should be involvement from more technical staff as opposed to research staff for development work, to free up scientists' time for user communication.

- ***Mantid developer workshops are generally found to be at least good in terms of usefulness, organisation, frequency, learning new things, and strengthening teamwork.*** There is a suggestion from a few comments that more time should be spent in smaller groups, to encourage wider participation, and one comment mentioned the last round of workshops that was split into a developer 'core' part and a wider dissemination part worked well.
- ***With user communications, there is very strong agreement that face-to-face meetings are important when engaging with users, and that overwhelmingly, it is the most efficient.*** Emails are also seen as an important channel of communication, and to some extent emails sent to the mantid-help@mantidproject.org email address are read by more than one person and are responded to according to a time period defined in policy, and there is a well defined policy for dealing with user queries. In terms of the user forum, there is general agreement that it is a useful resource for solving user queries, and is to some extent engaging, although there is only some agreement that it provides users with news about the project.

The Software

- ***In terms of how the developers regard the software, they generally consider it easy to understand what it does and its purpose, as well as its basic functionality, although conceptual understanding of advanced functionality is generally seen as difficult.*** The software and documentation is seen as easy or very easy to access and download. It's also generally easy to meet its prerequisites on a target platform, and easy to install and configure and quite easy to verify the installation for use. Achieving basic functional tasks is seen as easy, although learning how to achieve advanced functional tasks is difficult. Generally, it's quite easy to find out where and how to get software support, although keeping up with project news and developments slightly less easy. There is variability across the software, in terms of these aspects, due to how Mantid has grown organically.
- ***Generally, the documentation is seen as generally quite good in of terms of clarity, accuracy, and how up to date it is, and generally average in terms of its completeness.*** The most useful documentation categories were for installation, build and test, documentation available on the website, other tutorials and guides, and Python. The documentation on coding standards, good practices, Doxygen code documentation are seen as somewhat useful. It's good that the documentation has improved significantly in recent years, although it's noted that the developer documents need updating and are often difficult to find. Availability of documentation from a single location is suggested.
- ***Installing and configuring Mantid across the supported platforms is generally straightforward.*** However, there is some ambivalence as to how straightforward it is to obtain all third-party dependencies across the supported platforms, with building on Windows and OS X mentioned as problematic. Linux is noted as generally more straightforward.