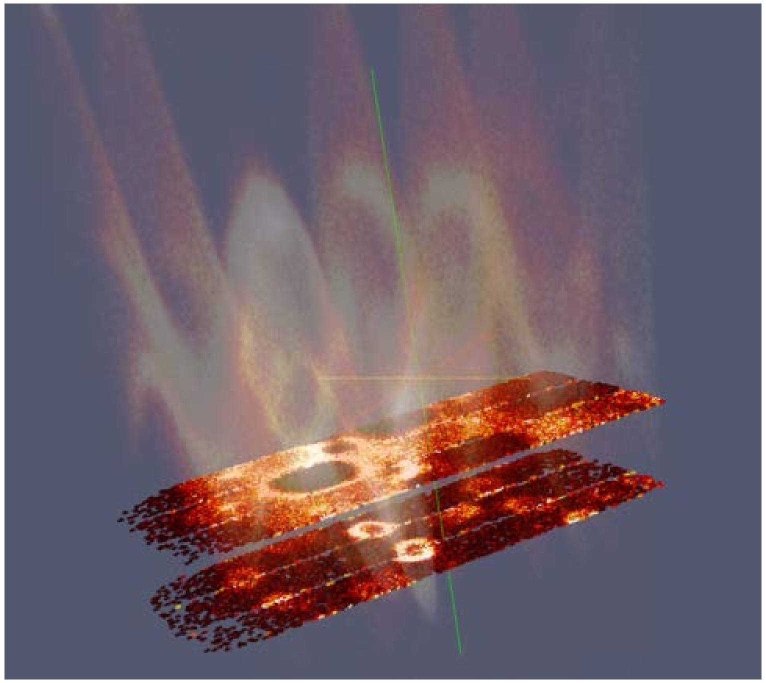
The Mantid Project:
Notes from an International Software Collaboration

Nick Draper
Tessella

www.mantidproject.org

Overview

- Mantid Introduction
- A Selection of Risks
- Management strategies
- Conclusion



Project Goals

- Goals
  - Consolidate the data reduction/analysis software for neutron scattering without restricting the needs of the instrument scientists
- Key requirement
  - Create a Data Analysis framework
    - not instrument or technique/dependent
  - Cross-platform
    - Windows, Linux, Mac
  - Easily extensible
  - Open source

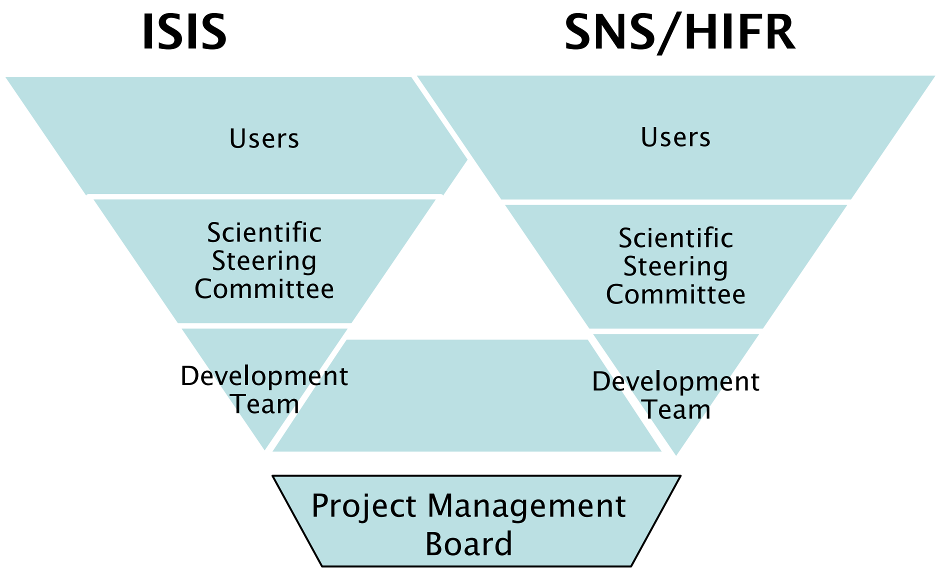
MANTID

A Selection of Risks

- Lasting engagement with a large number of stakeholders
- Design needs to support flexibility for future needs
- Technical single point of failure
- Development continuity across the team
- Larger development teams are less efficient
- Testing and deployment takes time & Active development can affect robustness

Lasting engagement with a large number of stakeholders

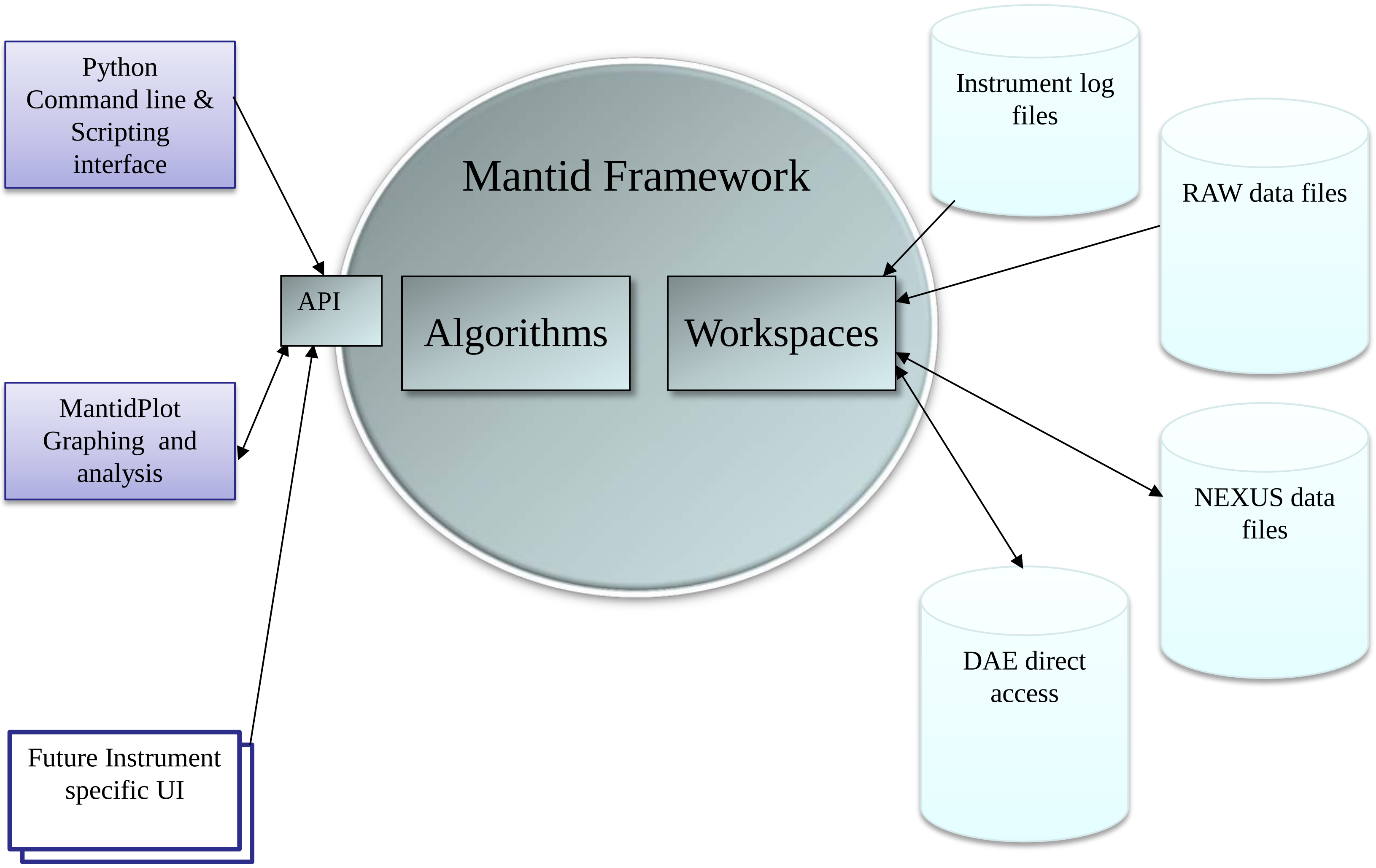
- Project Organisation
- Active project sponsors
- Frequent releases
- Responsive to change



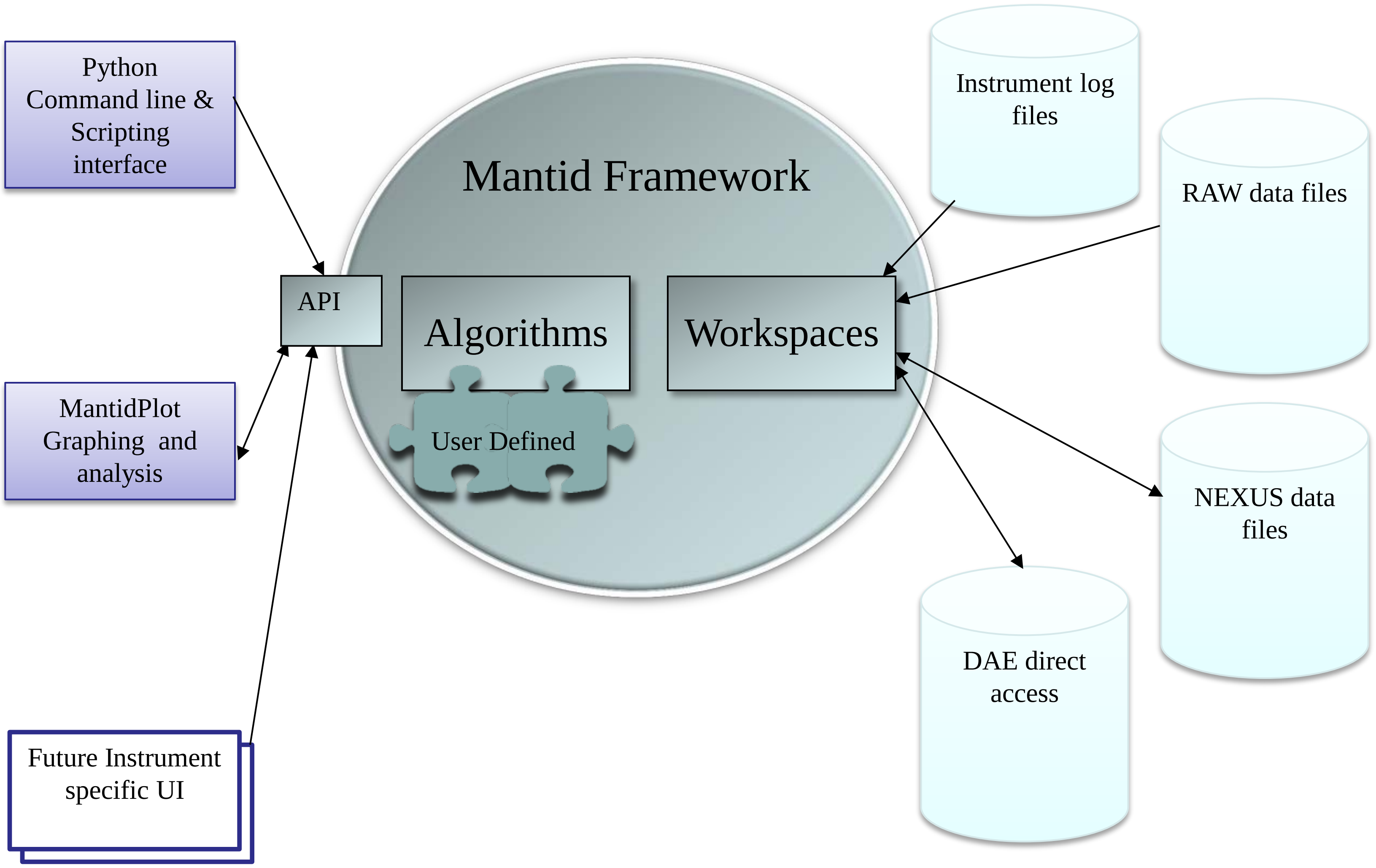
Design needs to support flexibility for future needs

- Separation of Data and Algorithms
- Encapsulated “User Code” in specific places
  - Algorithms
  - Workspaces
- Use of well designed interfaces to allow generic use of components
- Reuse of existing components
- Careful memory management when handling large datasets

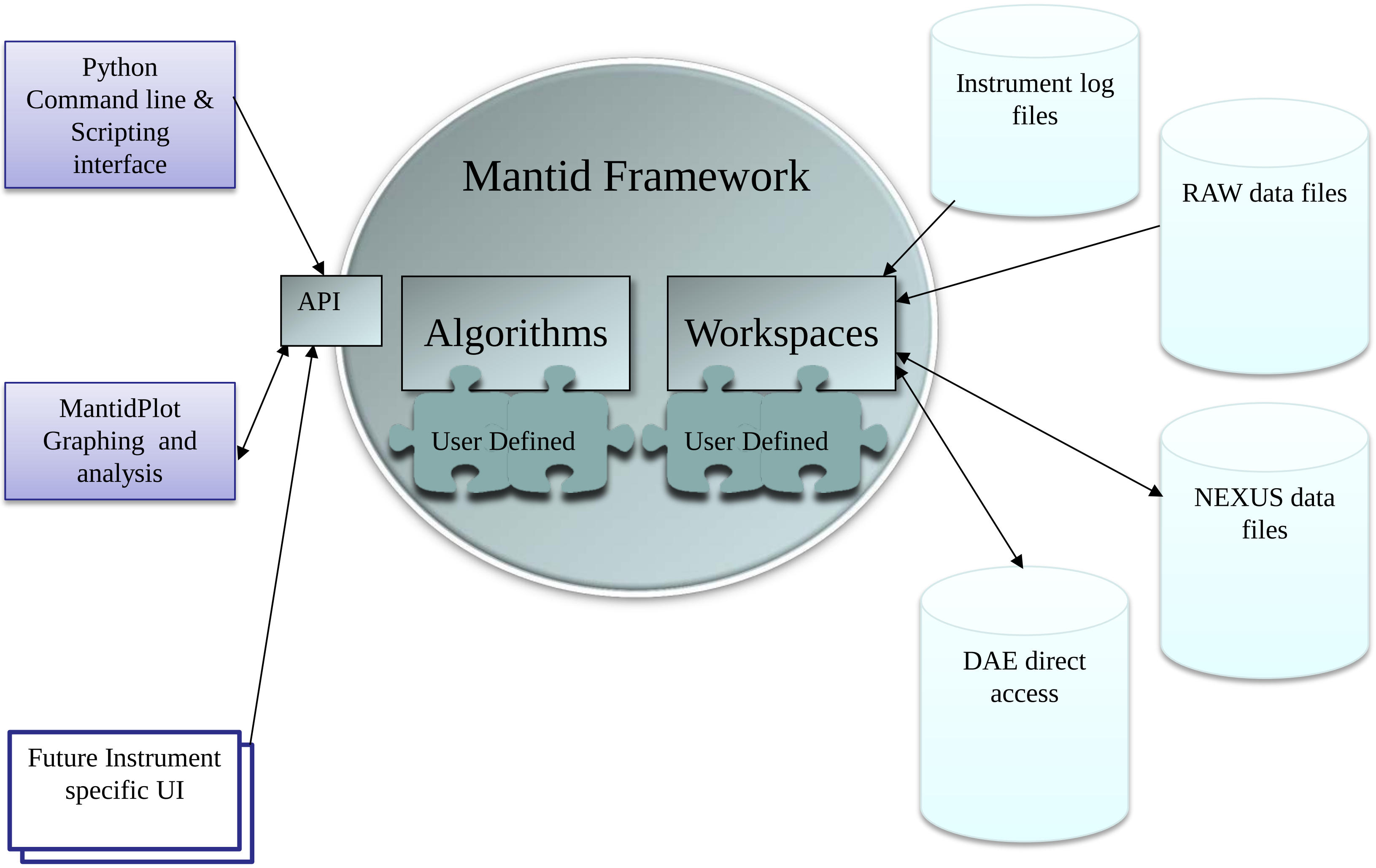
Architectural Design - Overview



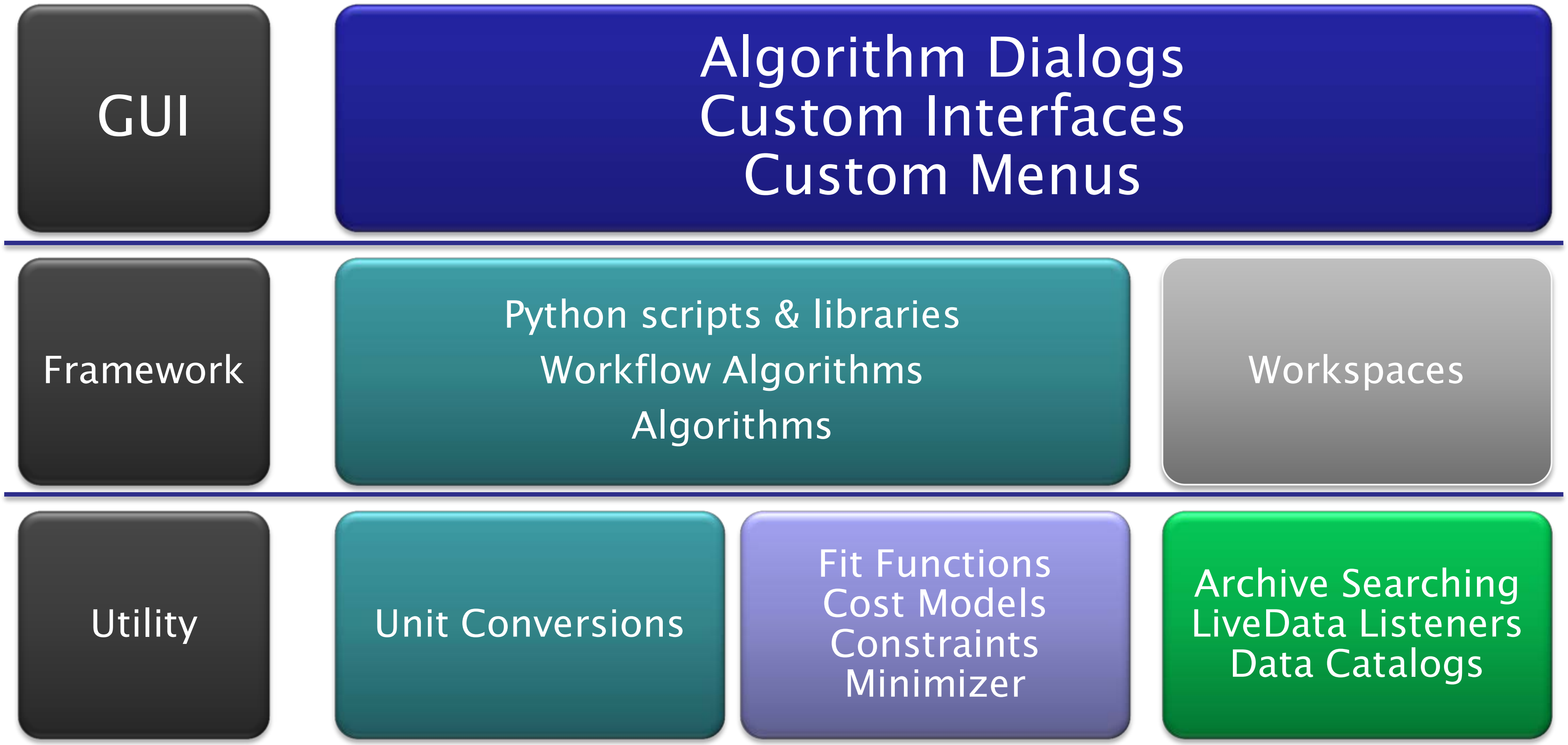
Architectural Design - Overview



Architectural Design - Overview

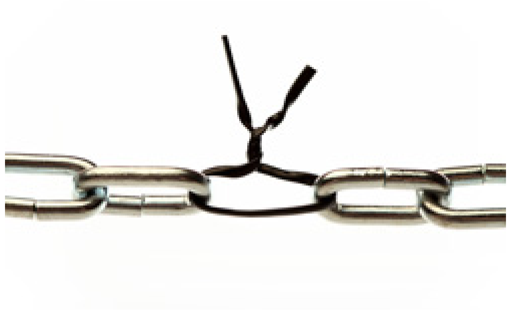


Plug in extensions



MANTID

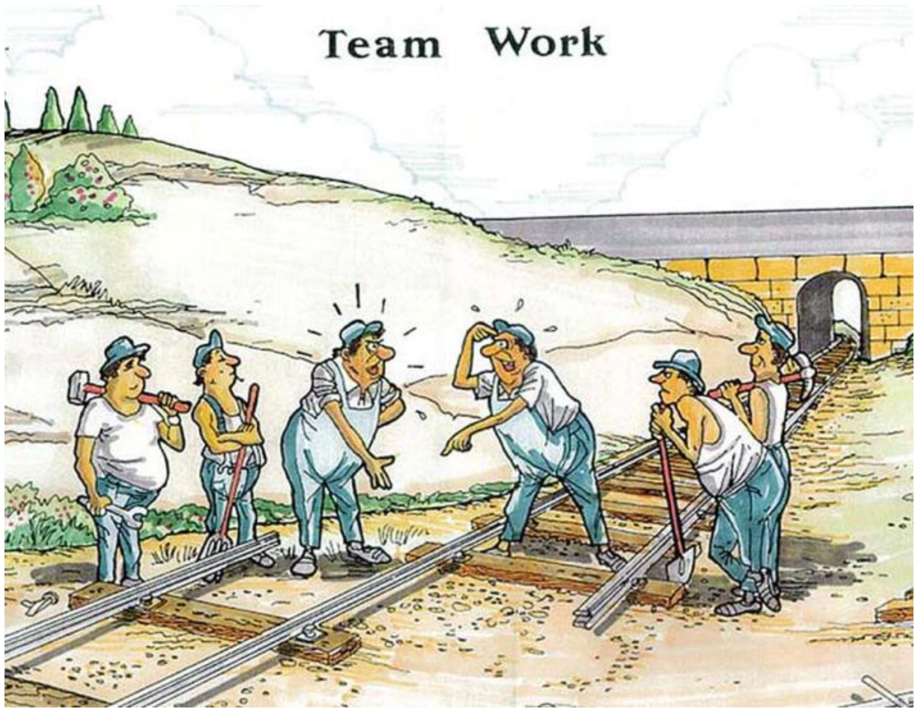
Preventing single points of failure



- No “Code Ownership”
  - Functionality protected via unit tests
- Mobile development talent
- Sub project teams to focus on significant developments
- Knowledge transfer
  - Daily & focused skype meetings
  - Code reviews
  - Architectural and detailed design documentation
  - Developer documentation
  - Annual developer meetings

Development continuity across the team

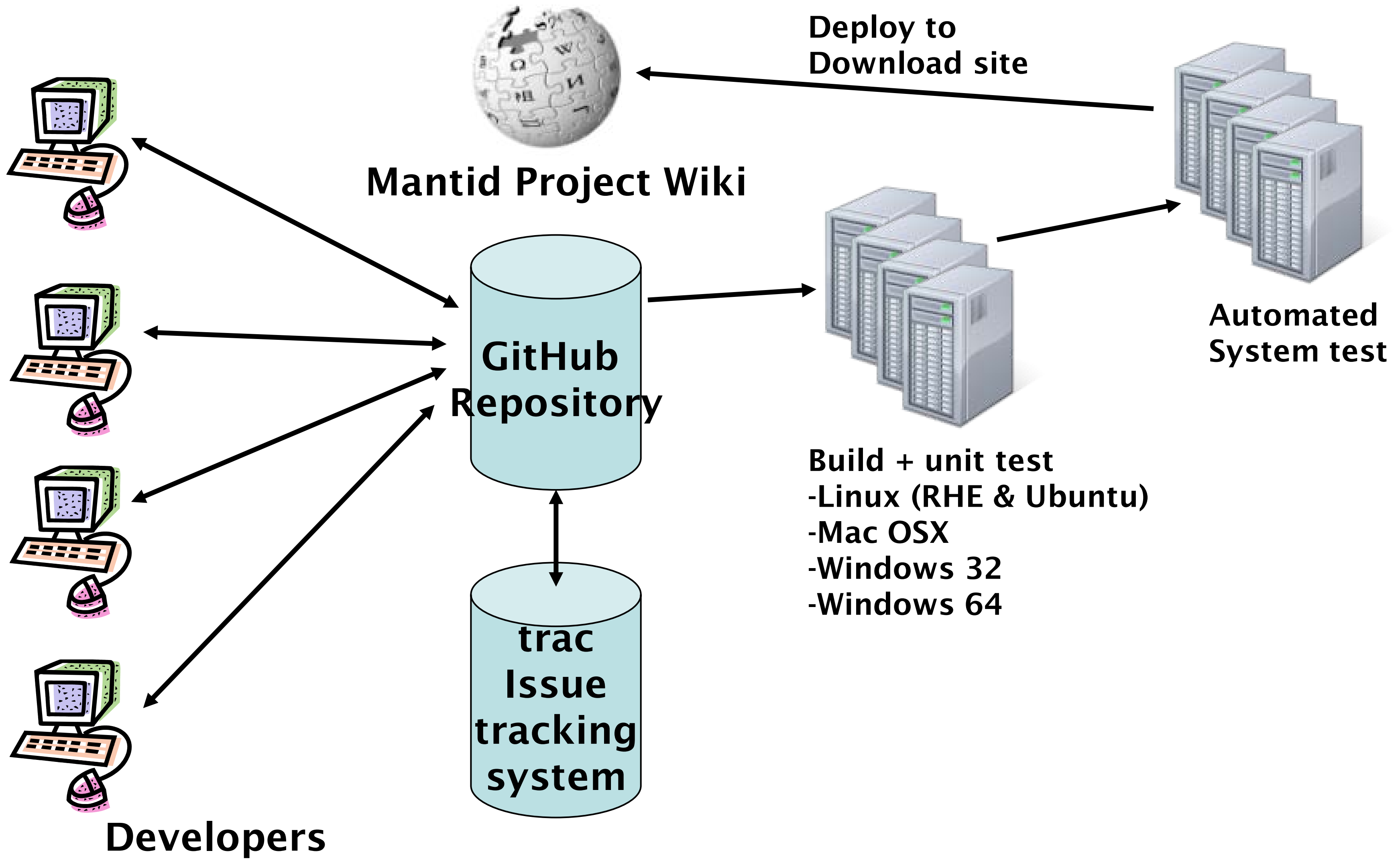
- Coding standards
  - Sensible
  - Agreed
- Shared code ownership
- Support within the team
  - Mentoring
  - Training
- Design and code reviews
- Developer meetings



Larger development teams are less efficient

- Automate repetitive tasks
  - Saves time
  - Ensures they happen
- Optimize meeting time
  - Control attendees at meetings
  - Use the right technology
    - Daily skype chat meetings
  - Ensure the right people talk together
- Use tools to prevent duplicated work and missed tasks
  - Development
  - Testing

Continuous Integration Environment



Testing and deployment takes time & Active development can affect robustness

- Automated Unit Testing
  - Test individual components
  - Over 6,000 tests
  - Fast – just a few minutes
  - Run on all platforms on commit
  - Rapid feedback to developers
- Automated System Tests
  - Test complete workflows
  - Compare numerical results with stored examples
  - Over 150 tests
  - Slow – minutes to hours
  - Run on all platforms daily
  - Feedback to all developers

Manual Testing

Developer Testing

- Each change reviewed and tested
- Whole development team, every week
  - Each developer tests other peoples work
  - Communication and knowledge sharing

Unscripted testing

- Usability and general usage tests
- Each environment tested
- Low coverage

User Testing

- Only once well tested & interactive development
- Instrument scientists
- Very high quality feedback & future requirements
- Generate confidence
- Must be well managed

MANTID

Releases

Development

- Automated release
- Daily
- If system tests pass
- Useful
- Not stable

Full Release

- Quarterly
- Full manual testing
- Full release notes
- Wide announcement
- Stable

Patch

- 2-4 weeks after a full release
- Targeted improvements & fixes
- Low risk
- Targeted testing
- Code review
- Stable

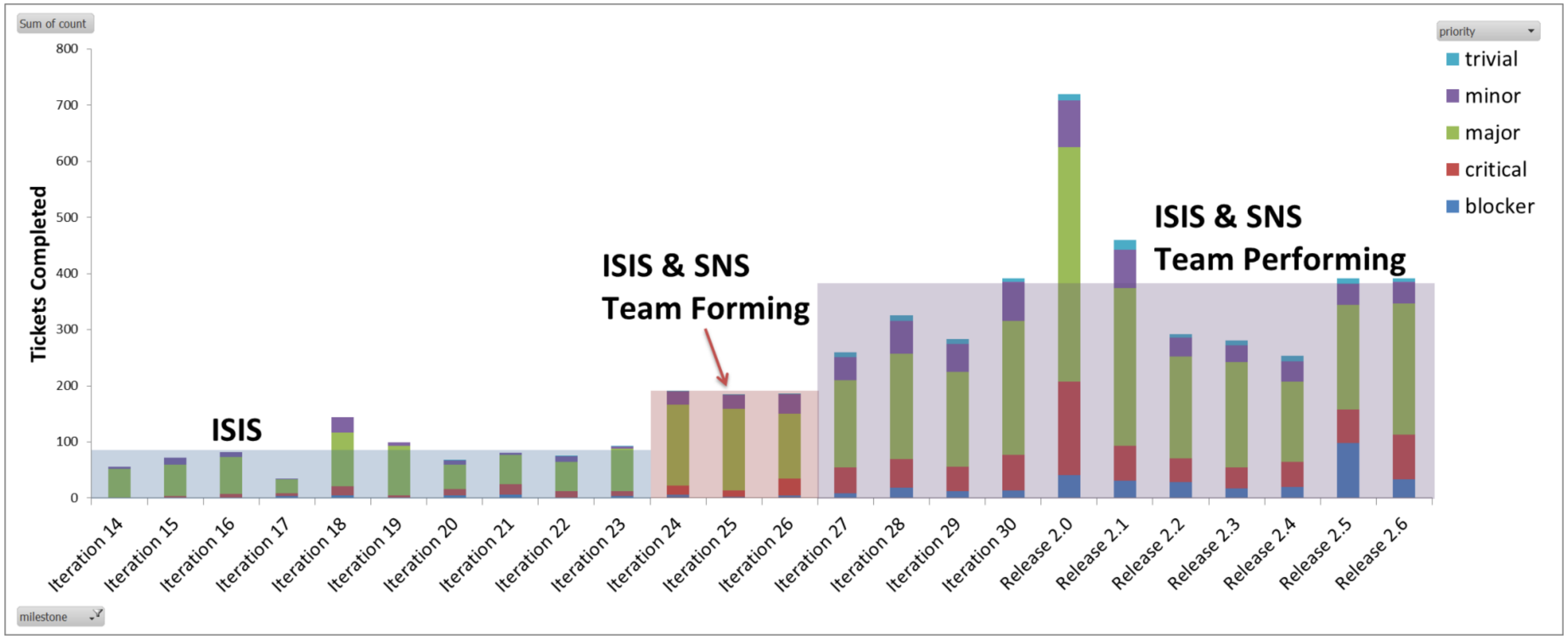
Conclusion

- Software is mission critical to modern neutron facilities
  - High performance
  - Reliable
  - Leading edge
  - Responsive to change
  - Maintainable
  - Well documented
- To get these a project needs
  - Vision
  - Resource
  - Stability
  - Scientific and Technical Leadership
  - Talented developers

MANTID

Conclusion

- A facility alone can provide these needs
  - Although many are not used to devoting their resources toward software developments.
- Working together can be more productive than the sum of the parts.



MANTID