Current Software and Practices

Ricardo FERRAZ LEAL

Institut Laue-Langevin

5th July 2012

NMI3-II
Data analysis software work-package 6

- Funding: 27 man-months started in June 2012.
    1. Review existing data analysis software and practices of software developers (2)
    2. Review existing solutions for a common data analysis infrastructure (2)
    3. Develop prototype software in chosen solution for representative applications (14)
    4. Evaluate prototype software (3).
- Proposed prototype:
    - $S(q,\omega)$ 4D data from reactor based multiplexed Xtal instruments
- One candidate:
    - Mantid <www.mantidproject.org>
    - VATES for 4D visualisation

Current software

TOF		
DAVE	Data Analysis and Visualization Environment	Inelastic Scattering (TAS,TOF,BS,spin echo)
Frida	Flexible rapid interactive data analysis	Inelastic Scattering (TOF and BS)
General		
LAMP	Large Array Manipulation Program	General purpose
ISAW	Integrated Spectral Analysis Workbench software project	TOF
Mantid		
Reflectivity		
GenX		Reflectivity
TAS		
Mfit	fit any type of (x,y) data with any fit function (even combinaisons)	ALL
Mview	manipulate and display up to 20 data files	ALL
Rescal	compute 4D resolution ellipsoid for inelastic scattering instrument	Inelastic Scattering
SANS		
DANSE	Distributed Data Analysis for Neutron Scattering Experiments	
Sansview	SANS data analysis and modeling	SANS
Grasp		SANS
Sasfit	analyzing and plotting small angle scattering data	SANS
Crystallography		
GSAS	General Structure Analysis System	Crystallography: powder + single crystal
EXPGUI	Graphical user interface to GSAS	
FullProf Suite		Crystallography: powder + single crystal
PDFgui	Pair distribution function fit (Gui for PDFFit2)	Crystallography: single crystal
PDFfit2	Python version of PDFfit	Crystallography: single crystal
Instrument Simulation		
McStas	Monte Carlo Simulation of TAS	TAS, TOF, polarised neutrons
Restrax	Monte Carlo simulations and data analysis	TAS
Vitess	Virtual Instrumentation Tool for the ESS	ALL?
Vtas	virtual Three Axis Spectrometer	TAS

Name	Version stable	Development
TOF		
DAVE	2.0 (2010)	s (based on IDL 8)
Frida	2.1.4c (2012)	yes
General		
LAMP	2012	yes
ISAW	v. 1.9.1_12a (2012)	yes
Mantid		
Reflectivity		
GenX	2.0.0 (2011)	SVN
TAS		
Mfit	2005	not anymore?
Mview		
Rescal		
SANS		
DANSE		
Sansview	2.1.1 (2012)	yes
Grasp	6.52 (2012)	Yes
Sasfit	0.93.3 (2011-05-4)	?
Crystallography		
GSAS	2009	?
EXPGUI	2011	2012?
FullProf Suite	01/05/12	Not avaiable
PDFgui	2.0-r3067`(2009)	No?
PDFfit2	3.0-r3067`(2009)	No?
Instrument Simulation		
McStas	1.12 (2012)	yes
Restrax	2011	yes
Vitess	2.11(2011)	yes
Vtas	4.1 (2010?)	No?

Name	Language	Libraries	Extendable	Source code
TOF				
DAVE	IDL 7.0		?	Yes (need IDL license)
Frida	C++	Yacc, Flex, Bison, GSL, gnuplot		yes
General				
LAMP	IDL	?	yes (IDL macros)	Yes (need IDL license)
ISAW	Java	Jython	yes (through operators)	Yes
Mantid				
Reflectivity				
GenX	Python	wxpython	yes (scripts, plugins)	yes
TAS				
Mfit	Matlab		Yes (routines + fit functions)	Yes (need Matlab license)
Mview				
Rescal				
SANS				
DANSE				
Sansview	C++ (Python bindings)	NumPy, SciPy, Matplotlib		yes
Grasp	Matlab		?	Yes (need Matlab license)
Sasfit	C	BLT for plotting	yes (plugins in C)	yes
Crystallography				
GSAS	C		?	No?
EXPGUI	tcl		TCL	Yes
FullProf Suite	Fortran	CrysFML	Difficult	Partly (CrysFML)
PDFgui	Python			yes
PDFfit2	C++ (Python bindings)			yes
Instrument Simulation				
McStas	C (Perl for scripting)	(scilab/matlab/pgplot)	yes (modules)	yes
Restrax	F77/90	RESCAL, VTAS	Difficult	Yes
Vitess	C	BLTwish, IDL, PV-Wave	yes (modules)	yes
Vtas	Java		no?	No?

Name	Simulation	Reduction	Visualisation	Analysis	Refinement (Rietveld analysis):	GUI
TOF						
DAVE	No	yes	yes	yes	No	+/-
Frida	No				No	No gui
General						
LAMP	No	yes	yes	yes	No	-/+
ISAW	No	yes	yes	yes	NA	+ **(**Swing**)**
Mantid						
Reflectivity						
GenX	No	Yes (differenti	yes	yes	NA	+++
TAS						
Mfit	No	yes (fitting?)	yes	?	No	+/-
Mview						
Rescal						
SANS						
DANSE						
Sansview	no					+ wxPython
Grasp	No	yes	yes	yes	NA	+/-
Sasfit	no	?	yes	yes	NA	TCL/TK
Crystallography						
GSAS	No				Yes	No Gui
EXPGUI	No				NA	+
FullProf Suite	No	DataRED	yes	yes	Yes	+/- (winteracter)
PDFgui			yes	yes	PDF	+ wxPython
PDFfit2			yes	yes	PDF	No Gui
Instrument Simulation						
McStas	Yes	NA	NA	NA	NA	Perl-TK
Restrax	yes	NA	NA	NA	NA	through SIMRES
Vitess	yes	NA	NA	NA	NA	+ (TCL/TK, IDL, PV-Wave)
Vtas	yes	NA	NA	NA	NA	++ / Swing

(bad) Practices

- Overlap of functionalities:
  - Common functionalities in different software:
    - Rewritten not imported!
- Poor collaboration:
  - “Fork” projects rather than contribute
  - e.g. Sassena @SNS (nMoldyn fork)
- Legacy code:
  - Spaghetti code: too difficult and risky to modify
  - Ongoing development in Fortran (e.g. crysFML)
  - Lack of testing

(bad) Practices (cont...)

- Unstructured code:
  - Lack of modules, objects, design patterns
  - Ongoing development in procedural languages
  - Difficult to (easily) extend
- Not all source code available
- Proprietary development frameworks:
  - IDL, MatLab, IGOR, PV-wave
    - Doesn't stimulate collaborative, pro-active development.
- Attempts to “re-invent the wheel”:
  - Gumtree, Mantid

Ideas for the Future

"Build to change instead of building to last."

Rewriting *vs* Refactoring

Software Architecture: in the Age of Compositionality

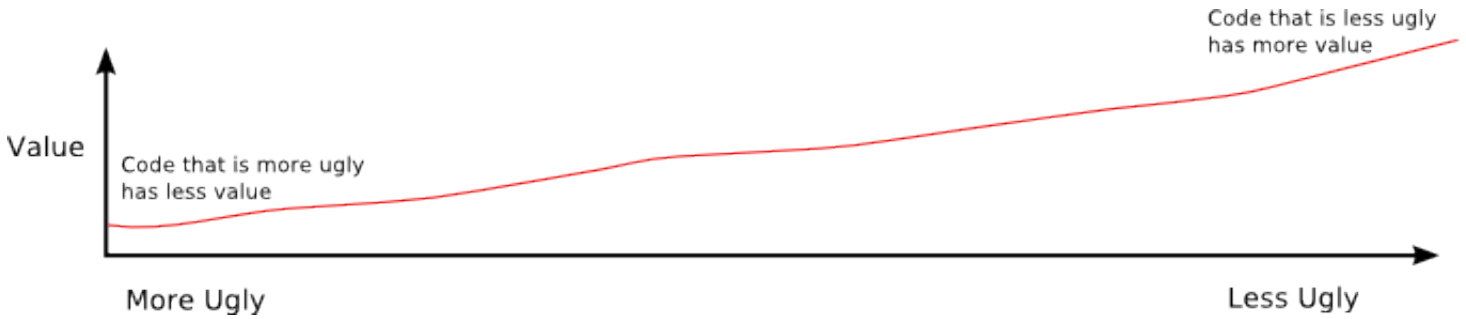
- Software engineering is changing
  - Building systems was previously the predominant activity
  - Focus has more recently shifted toward composing systems:
    - Open-source
    - Commercial and Proprietary components
  - Only build the functionality that truly is competitively differentiating!
- Short development cycles + client feedback

SOFTWARE ARCHITECTURE Lecture Notes in Computer Science, 2010, Volume 6285/2010, 1-4

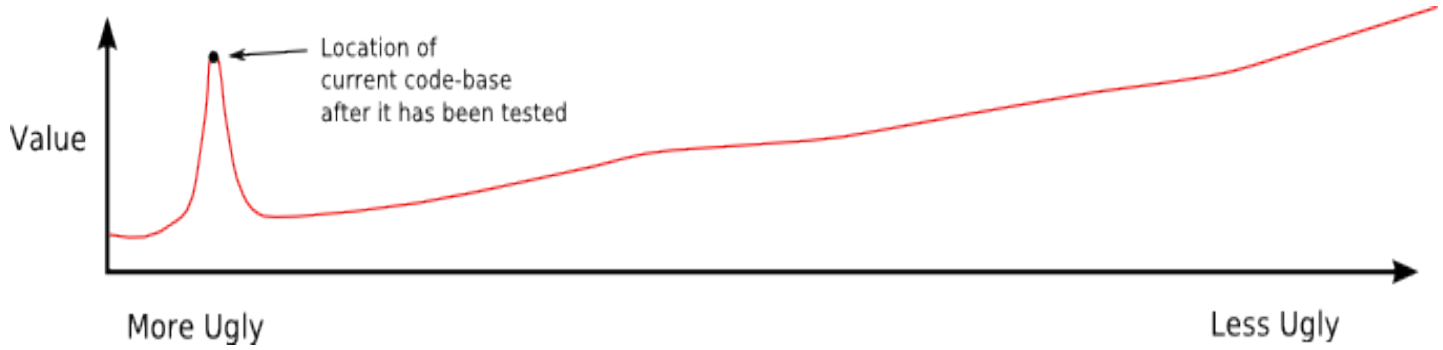
Production code

- “The longer the code you touch has been in production without issues, the more risk you take by changing it.”
- When the need arises to change the code:
  - Is there a very good reason to do it?
    - Quantify the benefit.
  - Refactor the code
  - Ensure that the code still works as it was designed to
- If the code is known/proven to work, it's value far exceeds how pretty or ugly it is!

The Value of ugly legacy code

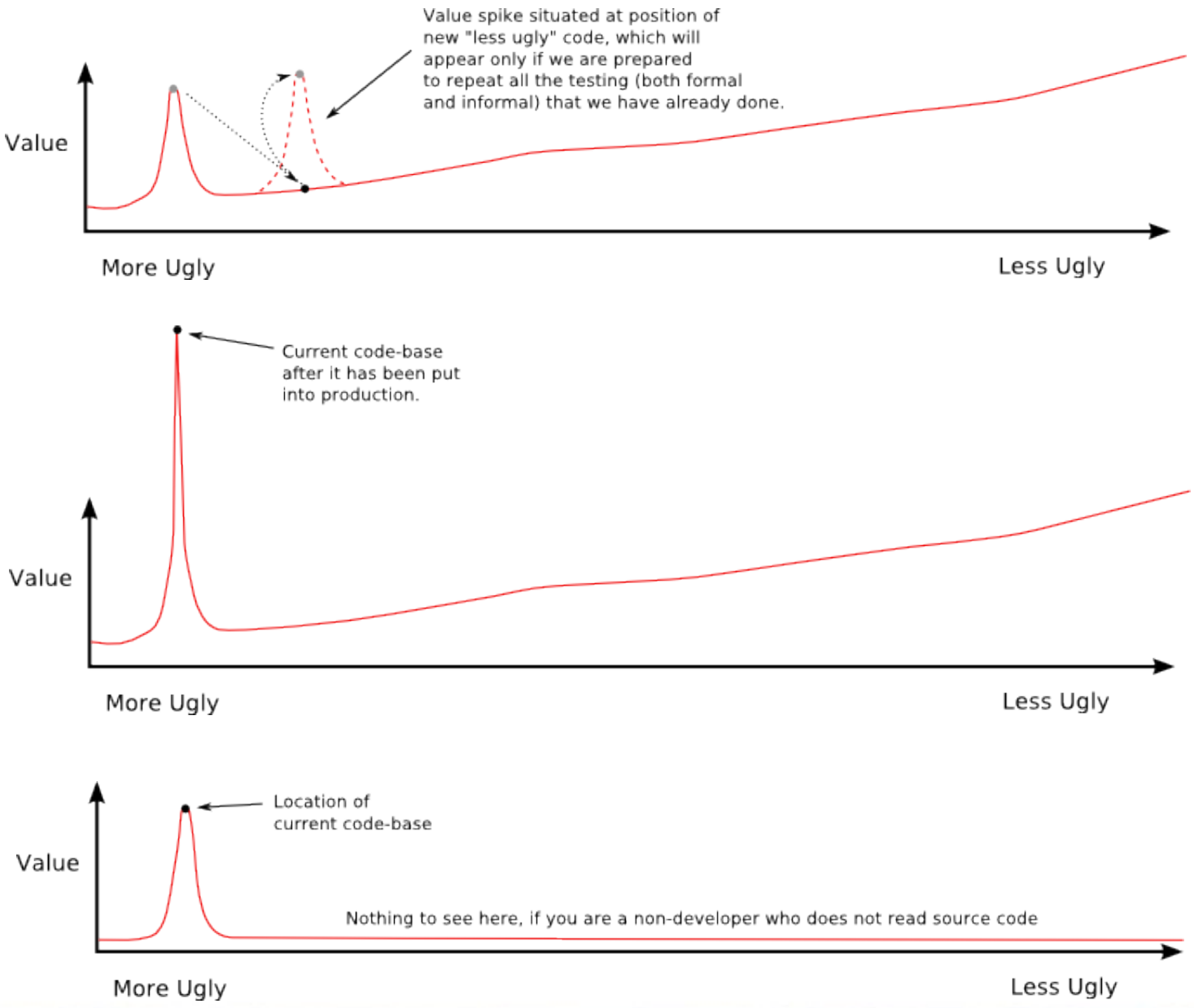


- "*The main thing that distinguishes legacy code from non-legacy code is tests, or rather a lack of tests.*",
  Michael Feathers, Working Effectively with Legacy Code



The Economics of Testing Ugly Code: http://www.1729.com

The Economics of Testing Ugly Code



How much does it cost to develop a line of code?

- 2002: 10 Must Knows for CIOs
  - ~$10 / line of code
- 2011: average salary $60K and 1850 hours worked per year, 20Klines/year/member.
  - between $12.33 and $18.5 / line of code.
- Technical debt: “it siphons money from IT innovation to pay for software repairs.”
  - 2010: CAST Software’s CRASH report:
    - ~$3.61 / line of code
    - Java: ~$5.42 / line of code

Solution for keeping legacy code: TDD – Test Driven Development

- TDD cycle:
  1. Create a unit test for a particular piece of legacy code
  2. Run the unit test and make sure it passes
  3. Refactor the code (in small safe steps) and check that the unit test still passes
- Use Design Patterns
  - Expose legacy methods through Façades

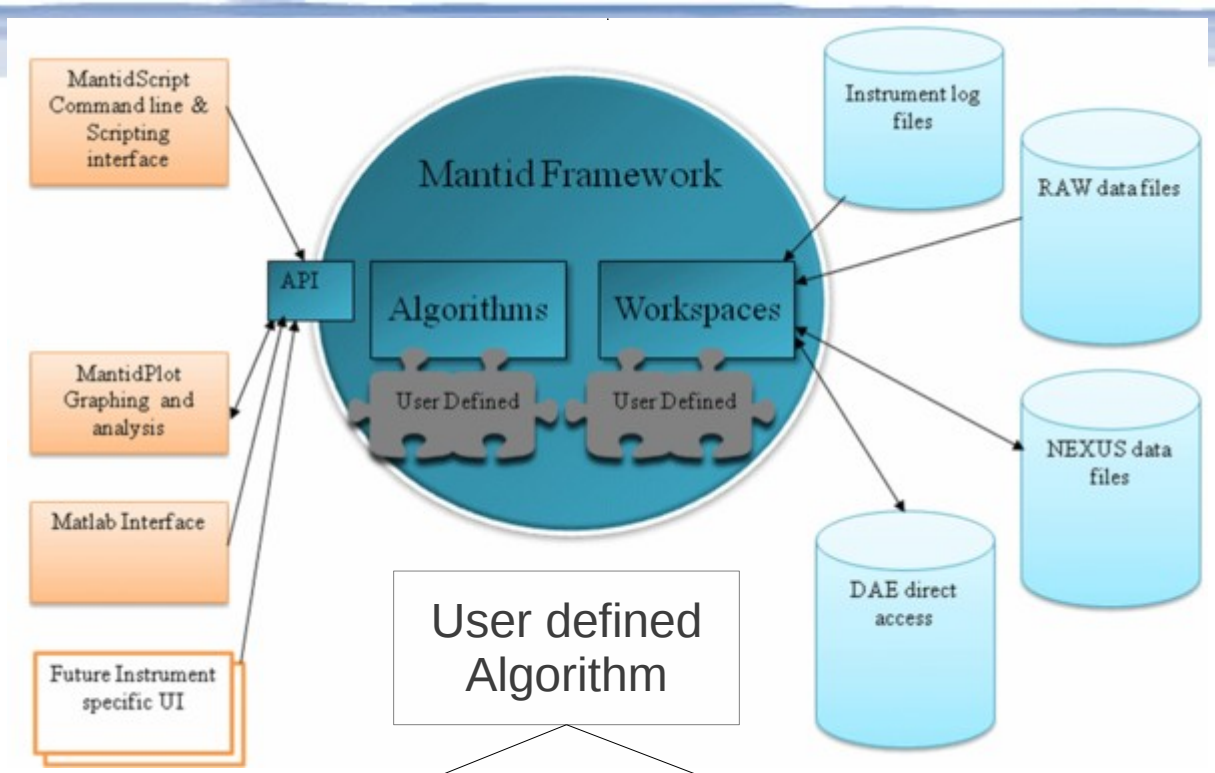
Writing the test: Creation of a form of specification.

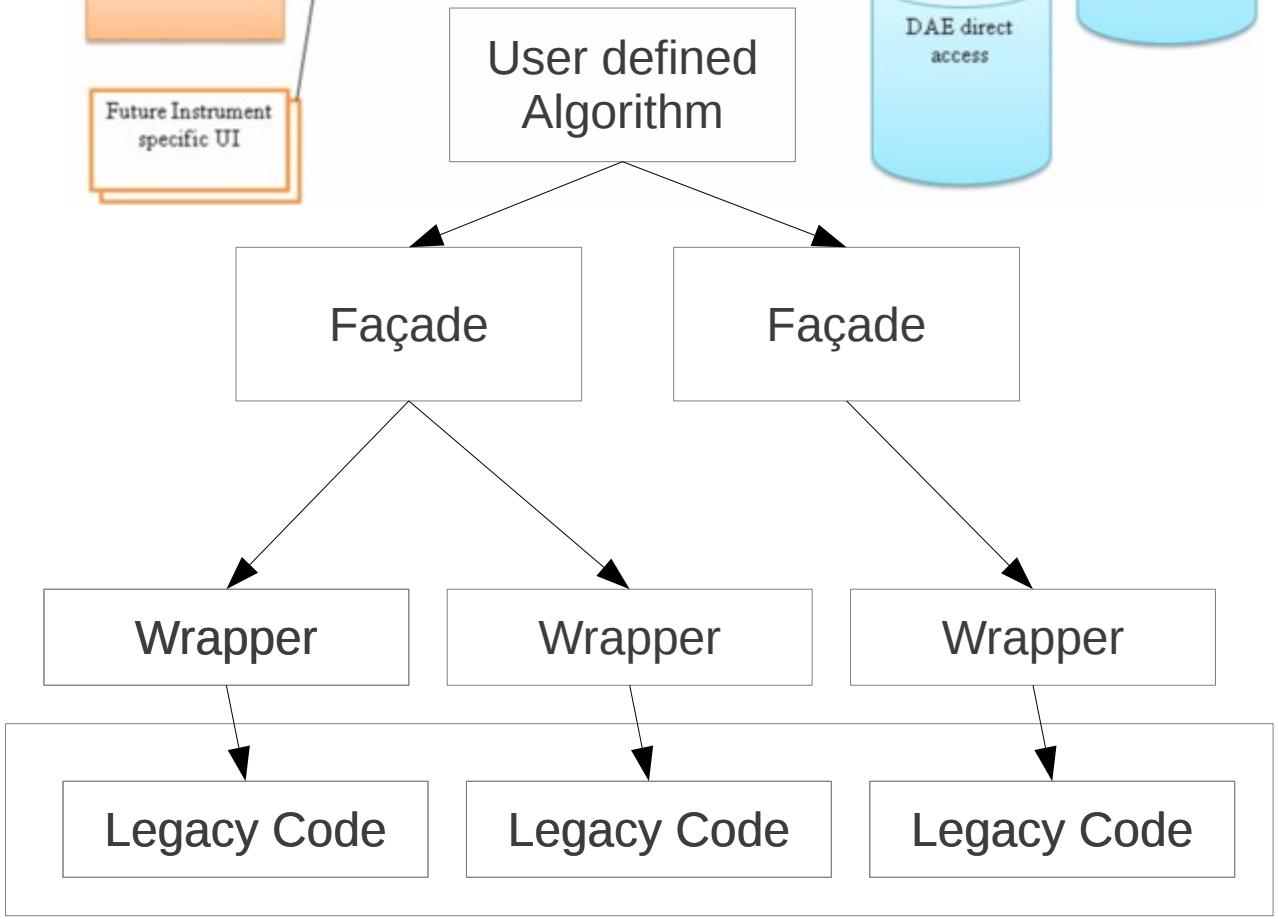
- Always write the test before refactor!!!

Making the test pass: Fulfilment of the requirement.

Solutions for proprietary software

- Convert MATLAB code into a C or C++ shared library using the MATLAB Compiler.
  - Development version would always need Matlab :(
- Same for IDL, PV-Wave, IGOR?
- Well documented façades
  - Legacy code remains “invisible” for the majority of future developers.


MantidScript Command line & Scripting interface
Mantid Framework
Instrument log files
RAW data files
API
Algorithms
Workspaces
MantidPlot Graphing and analysis
User Defined
User Defined
NEXUS data files
Matlab Interface
DAE direct access
Future Instrument specific UI


User defined Algorithm
Façade
Façade
Wrapper
Wrapper
Wrapper
Legacy Code
Legacy Code
Legacy Code

People seek pleasure

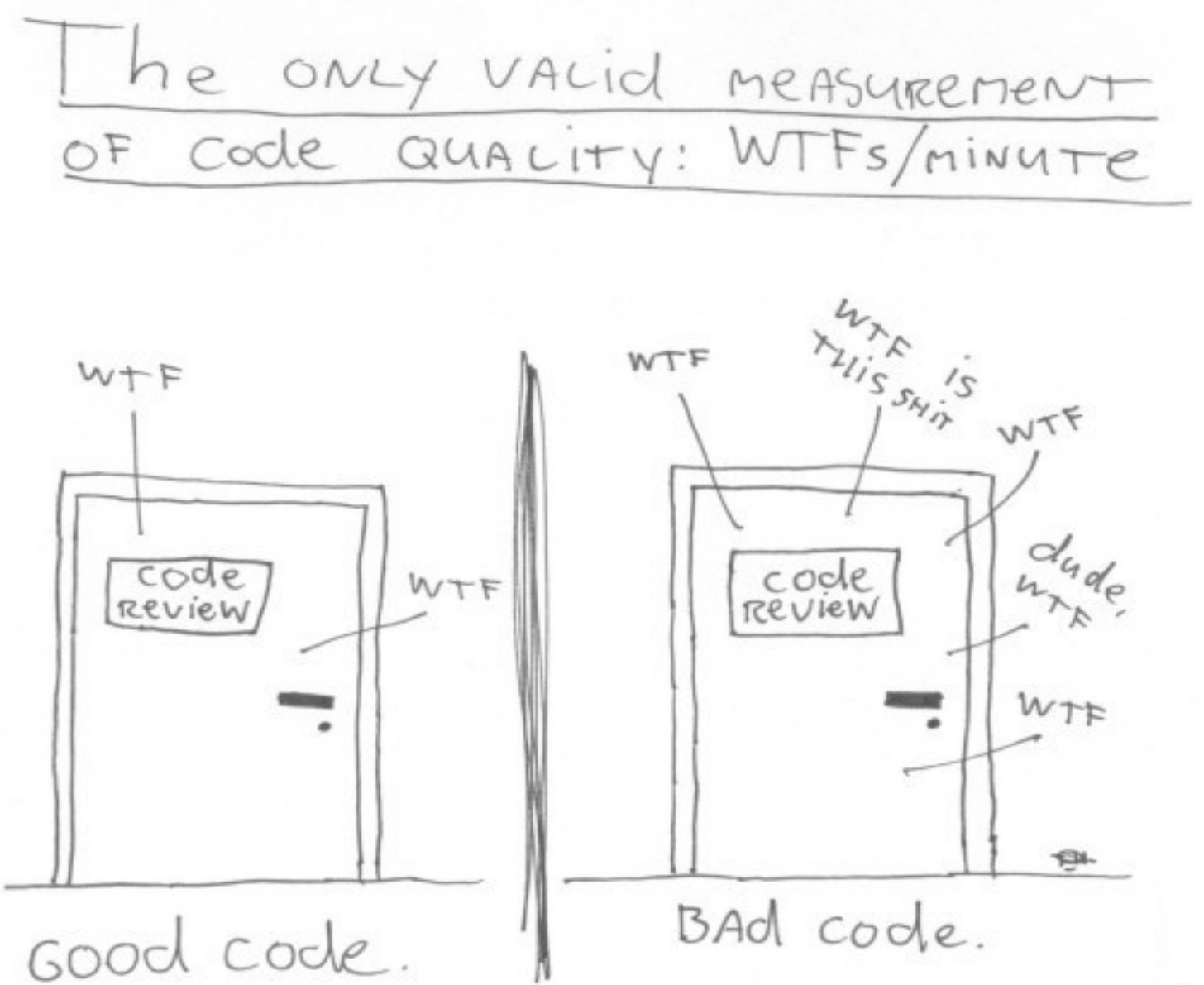
- “Did you ever wonder why cheap wine tastes better in fancy glasses?”
  - Emotion and cognition: Attractive things really do work better!

Donald A. Norman - Emotional Design: Why We Love (Or Hate) Everyday Things

"It's all about experiences",
www.poetpainter.com

- Mantid:
  - Usable, neither pleasurable nor meaningful
    - Convenient (super easy to use?)
  - Too many functionalities for end user.
  - Same interface for user / scientist.
  - “Eclipse” workspace+views concept would be a plus:
    - Per instrument:
      - Scientist
      - User
  -

Questions?


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

References:

- Working Effectively with Legacy Code, Michael Feathers
- Refactoring: Improving the Design of Existing Code, Martin Fowler
- Refactor Low Hanging Fruit
  - http://c2.com/cgi/wiki?RefactorLowHangingFruit
- What is Software Architecture?
  - http://msdn.microsoft.com/en-us/library/ee658098.aspx
- When do I need to stop using design patterns?
  - http://stackoverflow.com/questions/1295524/when-do-i-need-to-st
- Emotional Design: Why We Love (Or Hate) Everyday Things, Donald A. Norman
- www.poetpainter.com

- *“You can create an absolutely beautiful architecture with the cleanest code in the world. You may have 100% test coverage, complete separation of concerns, flat hierarchies and methods without boolean arguments. You may have all that beauty, but still fail miserably if the program does not solve user’s problems efficiently.” do-really-all-projects-fail-because-of-code (www.targetprocess.com)*
- In my experience, projects more often go wrong when the solution tries to solve the wrong problem rather than the implementation of that solution.

Key Architecture Principles

- Build to change instead of building to last.
  - Consider how the application may need to change over time to address new requirements and challenges, and build in the flexibility to support this.
- Model to analyze and reduce risk.
  - Use design tools, modeling systems such as Unified Modeling Language (UML), and visualizations where appropriate to help you capture requirements and architectural and design decisions, and to analyze their impact. However, do not formalize the model to the extent that it suppresses the capability to iterate and adapt the design easily.
- Use models and visualizations as a communication and collaboration tool.
  - Efficient communication of the design, the decisions you make, and ongoing changes to the design, is critical to good architecture. Use models, views, and other visualizations of the architecture to communicate and share your design efficiently with all the stakeholders, and to enable rapid communication of changes to the design.
- Identify key engineering decisions.
  - Use the information in this guide to understand the key engineering decisions and the areas where mistakes are most often made. Invest in getting these key decisions right the first time so that the design is more flexible and less likely to be broken by changes.

Current Software for data reduction, analysis and visualisation

- Elastic Scattering
  - Crystallography:
    - Single crystal (Mono , Laue)
    - Powder
  - Liquid
  - SANS:
  - Reflectometry:
  - (d3 and d7) polarised neutrons
  - Spin-echo
- Inelastic scattering
  - TAS:
    - Spin-echo
  - TOF:
  - back scattering

Crystallography

- GSAS + EXPGUI
- FullPROF suit
- PDFfit2 + PDFGui (DANSE)
-
- Just Expgui (tcl) and Fullprof (Fortran) still being developed
- GSAS no GUI and no source code available
- FullProf not all source code available, uses proprietary software for GUIs. Data reduction available.
- PDFfit2 + PDFGui developed in python with C++ bindings. Gui in wxPython.

SANS

- Sansview (DANSE)
  - C++ with python bindings. Guin in wxPython.
  - Widely used. Still being developed.
- Grasp
  - Matlab. Supports data reduction.
  - Still being developed.
- Sasfit
  - C. Plugin options. Gui in TCL+TK.
  - Last version from 2011.

All supported in Osx, Win and Linux

Inelastic

- DAVE
  - TAS, TOF, Back Scattering and Spin-echo
-
- Frida
  - TAS, TOF, Back Scattering and Spin-echo
- IFit and Mfit + Mview + Rescal
  - TAS

General purpose

- LAMP
- ISAW
-

Instrument simulation

- McStas
- Vitess
- Restrax : only for TAS