BornAgain: simulation and fitting framework for scattering experiments at grazing incidence and reflectometry

Jonathan Fisher, Marina Ganeva, Walter Van Herck, Gennady Pospelov, Joachim Wuttke

*Scientific Computing Group at MLZ*
*Jülich Center for Neutron Science*

SXNS14, Stony Brook University, July 10-14, 2016
Outline

- Motivation
- Software architecture
- Demonstration
- Closing remarks
Motivation

Heinz Maier-Leibnitz Zentrum
Garching by Munich, Germany

FRM II
20 MW neutron source
8×10^{14} neutrons cm^{-2}s^{-1}

More than 30 instruments
Including MARIA, REFANS, NREX reflectometers

Scientific Computing group at MLZ
- Develop, adapt, and maintain software for data reduction and analysis
  - Software for non-expert users at GISAS field
  - With polarized-neutrons support
  - Extensibility for broader usage
Grazing Incidence Small Angle Scattering

**Experiment**

- the beam is directed on a surface with a very small incident angle
- 2D detector records the intensity of scattered wave giving access to lateral and vertical sample structure information

**Simulation**

- Intensity is calculated from known sample structure using **Distorted Wave Born Approximation**

\[
\frac{d\sigma}{d\Omega} = \left|\langle F_{\text{DWBA}} \rangle\right|^2 S(qz)
\]
Grazing Incidence Small Angle Scattering

Experiment

- the beam is directed on a surface with a very small incident angle
- 2D detector records the intensity of scattered wave giving access to lateral and vertical sample structure information

Simulation

- Intensity is calculated from known sample structure using Distorted Wave Born Approximation

\[
\frac{d\sigma}{d\Omega} = \left| F_{\text{DWBA}} \right|^2 S(q)\]
- Motivation
- **Software architecture**
- Demonstration
- Closing remarks
User needs

Users → Easy-to-use → GUI
Requirements

Users

- Easy-to-use

  Availability

  - Windows
  - Mac
  - Linux

  GUI

  Open source

  No proprietary software

  Easy-to-install

  Multi platform

BornAgain framework  SXNS14, July 10-14, 2016
Requirements

Users

Easy-to-use

GUI

Availability

Open source
No proprietary software
Easy-to-install

Advanced Users

Windows

Multi platform

Mac

Clusters

C++

Linux

OO, decoupling

Complex models

Performance

Python

Extensibility

Scripting
Welcome to BornAgain

BornAgain is a software package to simulate and fit small-angle scattering at grazing incidence. It supports analysis of both X-ray (GISAXS) and neutron (GISANS) data. Its name, BornAgain, indicates the central role of the distorted wave Born approximation in the physical description of the scattering process. The software provides a generic framework for modeling multilayer samples with smooth or rough interfaces and with various types of embedded nanoparticles.

Read more
BornAgain architecture

- Open-source, GPL3 license
- Multi-platform (Windows, Mac OS, Linux)
- C++ kernel for simulation and fitting
- Python bindings
- Well established open-source libraries as external dependencies
- Standalone Graphical User Interface

**Diagram:**

- **Python bindings**
  - **C++ kernel**
  - **External dependencies:** Eigen, fftw3, GSL

- **Standalone GUI**
  - **External dependencies:** Qt5
Working with BornAgain

- Using Graphical User Interface
- Running Python script with simulation description
Functionality

- X-rays, non-polarized and polarized neutrons
- Arbitrary number of layers
- Rough interfaces
- Simple and composite particles
- Correlated positions
- Nanoparticle assemblies
- Off-specular geometry, beam divergence
o Motivation

o Software architecture

o Demonstration

o Closing remarks
Project infrastructure

Software management
- Version control (Git)
- Issue tracking (Redmine)
- Documentation (Doxygen, Drupal)

Quality assurance
- Unit tests (googletest, QtTests)
- Functional tests
- Static code analysis

Continuous integration
- Nightly builds (buildbot, vagrant, docker)
- Release procedure

Number of lines of code in BornAgain git repository vs time
Validation

Validation against existing software

BornAgain

IsGISAXS

difference

$10^{-10}$

Validation against experimental data
Plans

- In our agenda
  - Particles crossing interfaces
  - Specular peak, transmission
  - Adapting BornAgain for reflectometry
  - Magnetic domains and magnetic roughness
  - Real space visualization

The order of appearance of new features depends from the user community. Tell us what you want next!

1St BornAgain School and User Meeting
21-22 November, Munich

Satellite of GISAXS2016 workshop
Bornagainproject.org
making DWBA manageable

Thank you!
Backup
Motivation

Why GISAS?

- Support for specific instruments at MLZ
  - Maria and REFSANS reflectometers
  - Absence of the program to simulate polarized neutrons at grazing incidence

- High Data Rate Processing and Analysis initiative (HDRI)
  - Call to create simulation software for non-expert users
  - Grazing incidence as main target

Start community project for data analysis and simulation

- Create generic framework covering simulations of x-rays, polarized and non-polarized neutrons at grazing incidence
- Provide functionality/extensibility for broader usage
User needs
Distorted Wave Born Approximation (DWBA)

- accounts for reflection-refraction effects close to the critical angle
- interference between scattering from direct and reflected beam

\[
\frac{d\sigma}{d\Omega} = \left| F_{\text{DWBA}} \right|^2 S(q_z)
\]

Scalar case

\[
F_{\text{DWBA}}^i(k_i, k_f, R_z) \equiv T_i T_f F^i(k_i - k_f) e^{i(k_i - k_f)R_z} + R_i T_f F^i(\tilde{k}_i - k_f) e^{i(-k_i - k_f)R_z} + T_i R_f F^i(k_i - \tilde{k}_f) e^{i(k_i + k_f)R_z} + R_i R_f F^i(\tilde{k}_i - \tilde{k}_f) e^{i(-k_i + k_f)R_z}
\]

Scalar case

\[
f(\theta, \phi) \sim \langle \Psi_{\text{out}} | V | \Psi_{\text{in}} \rangle \rightarrow 2 \cdot 2 = 4 \text{ terms}
\]

Scalar case

\[
f(\theta, \phi) \sim \langle \Psi_{\text{out}} | V | \Psi_{\text{in}} \rangle \rightarrow 4 \cdot 4 = 16 \text{ terms (}2 \times 2\text{) polarization}
\]
Working with BornAgain

- Object oriented approach in sample description
Working with BornAgain

- Object oriented approach in sample description

```python
# defining materials
m_air = HomogeneousMaterial("Air", 0.0, 0.0)
m_substrate = HomogeneousMaterial("Substrate", 6e-6, 2e-8)
m_particle = HomogeneousMaterial("Particle", 6e-4, 2e-8)

# collection of particles

cylinder_ff = FormFactorCylinder(5*nanometer, 5*nanometer)
cylinder = Particle(m_particle, cylinder_ff)
prism_ff = FormFactorPrism3(10*nanometer, 5*nanometer)
prism = Particle(m_particle, prism_ff)
particle_layout = ParticleLayout()
particle_layout.addParticle(cylinder, 0.0, 0.5)
particle_layout.addParticle(prism, 0.0, 0.5)

# air layer with particles and substrate form multi layer
air_layer = Layer(m_air)
air_layer.addLayout(particle_layout)
substrate_layer = Layer(m_substrate)
multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)
```
Interface roughness

- Multilayered sample with correlated roughness
- Roughness described by
  - rms roughness of interface
  - Hurst parameter
  - lateral correlation length
  - cross-correlation length

For each interface:

For multilayer:
Approximations

- **Decoupling approximation**
  - No correlation between type and interparticle distances

\[
\frac{d\sigma}{d\Omega}(q) = I_d(q) + S(q) \cdot \left| \langle F_\alpha(q) \rangle_\alpha \right|^2
\]

- **Local Monodisperse Approximation**
  - Incoherent superposition of different domains, each with their own type and interference function

\[
\frac{d\sigma}{d\Omega}(q) = \left\langle S_\alpha(q) \cdot |F_\alpha(q)|^2 \right\rangle_\alpha
\]

- **Size-Space coupling approximation**
  - Interparticle distance parameter in interference function depends on sizes of the two particles considered
Interference functions

- Finite size effects are modeled by convoluting the reciprocal lattice with a Fourier transformed distribution.
- For paracrystals positional disorder is cumulative.

- 1D lattice
  ![1D lattice diagram]
- 1D paracrystal
  ![1D paracrystal diagram]
- 2D lattice
  ![2D lattice diagram]
- 2D paracrystal
  ![2D paracrystal diagram]
Interference functions

- Single cylinder form factor
- Only interference function
- Cylinder form factor combined with interference function
○ Every shape can be rotated in all 3 directions (form factors are complex in q)
Complex shapes

Core shell particles

Particles with size distribution
With possibility to link parameters

Particle compositions
collection of particles with fixed inter-particle distance
coherent interference

All can be rotated
Complex shapes (2)

- Mesocrystals
  - Outer shape of the mesocrystal
  - Lattice vectors
  - Lattice basis (consisting of ‘regular’ particles and their positions)

\[
S_{\text{total}}(r) = S_{MC}(r) \cdot \sum_{r_i \in \Lambda} S_{NP}(r) \otimes \delta(r - r_i)
\]

\[
F_{\text{total}}(q) \propto F_{MC}(q) \otimes \left\{ F_{NP}(q) \cdot \sum_{q_i \in \Lambda^*} \delta(q - q_i) \right\}
\]
Very large particles

Large particles gives rise to a problem, known in communication theory as aliasing: Rapidly oscillating signal measured at fixed points shows up as slow sinusoid.

In GISAS simulation
Rapidly oscillating form factor of large particles leads to a significant variation of intensity over the detector bin.
Very large particles

Small cylinders
height = 10 nm
radius = 20 nm

Large cylinders
height = 1000 nm
radius = 2000 nm
Experimental setup

**Basic GISAS setup**
- beam is defined via wavelength and incidence angles
- detector parameters are set using range of angles and number of detector bins
- It is possible to define beam divergence and detector resolution function

![Beam divergence OFF](image1.png)  
**beam divergence OFF**

![Beam divergence ON](image2.png)  
**beam divergence ON**
Experimental setup (2)

- Specular and off-specular experimental setups already exist in BornAgain.
- We need more feedback from users to improve existing functionality.
Extensibility

Fast prototyping with BornAgain

- it is possible to extend C++ kernel with Python code
- for example, user can define new form factor on Python side and pass it to the simulation

```python
class CustomFormFactor(IFormFactor Born):
    # A custom defined form factor
    def __init__(self, L, H):
        IFormFactorBorn.__init__(self)
        # parameters describing the form factor
        self.L = L
        self.H = H

    def evaluate_for_q(self, q):
        qzhH = 0.5*q.z()*self.H
        qxhL = 0.5*q.x()*self.L
        qyhL = 0.5*q.y()*self.L
        return 0.5*self.H*self.L**2*
        cmath.exp(complex(0.1)*qzhH)*sinc(qzhH)*
        (sinc(0.5*qyhL)*(sinc(qxhL) - 0.5*sinc(0.5*qxhL))
        + sinc(0.5*qxhL)*sinc(qyhL))
```
Toy fitting example

\[ f(x, y; p) = p_0 + p_1 \cdot \text{sinc}(x - p_2) \cdot \text{sinc}(y - p_3) \]

There you are!
Fitting in BornAgain

User Data
- Sample description
- Experimental setup
- Fitting parameters
- Reference data
- Minimizer settings

BornAgain
- New sample description
- New values of fit parameters
- Minimization
- Chi2 value

Reference data
- Simulation
- Simulated intensity
- Chi2 calculations

Optimal sample parameters

Intensity plot
**Fitting in BornAgain: main features**

- Variety of minimization algorithms
  
<table>
<thead>
<tr>
<th>Library</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minuit2</td>
<td>Migrad, Simplex, Fumili, Scan</td>
</tr>
<tr>
<td>GSL</td>
<td>Fletcher-Reeves Conjugate Gradient Algorithm</td>
</tr>
<tr>
<td></td>
<td>BFGS Conjugate Gradient Algorithm</td>
</tr>
<tr>
<td></td>
<td>Levenberg-Marquardt Algorithm</td>
</tr>
<tr>
<td></td>
<td>Simulated Annealing Algorithm</td>
</tr>
<tr>
<td>TMVA</td>
<td>Genetic Algorithm</td>
</tr>
</tbody>
</table>

- Possibility to fit every sample parameter or their combination

```java
FitParameter("par1", 8.0*nm, limited(5.0, 15.0))
radius = fun1(par1); lattice_length = fun2(par1)
```

- Various fit strategies (e.g. fix/release parameters)
Fitting in BornAgain: main features (2)

- Organizing different minimization algorithms into the chain
  - Genetic minimizer explores large parameter space, Levenberg-Marquardt finalize location of minima

- Simultaneous fit of two datasets
  - Two experimental images obtained for different incident angles can be fit with one sample model

- Fitting along slices, masking certain areas of the detector image
Testing

Unit tests
- Core library (google-test, 330 tests), GUI models (QtTest, 60 tests)

Functional tests
- Runs simulation for certain geometry, produces intensity plot
- Compares the plot with the reference
  - simulation from previous day
  - simulation through different chain (Core/GUI/Python)
  - simulation of identical samples obtained in different way

Create particle composition from two hemi spheres
- Assign same material to them
- Compare with normal full sphere, same material, same radius
- Scattering intensities should be identical
## Functional tests for Core/GUI/Python domains

- When new functionality is implemented the corresponding standard simulation is added to the factory
- Corresponding intensity data is generated and saved for future reference.

```
makes check launches test simulations for all 3 domains
```

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>139/146</td>
<td>GUI Suite/BoxCompositionRotateZandY</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>140/146</td>
<td>GUI Suite/BoxStackComposition</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>141/146</td>
<td>GUI Suite/BoxStackComposition</td>
<td>Passed</td>
<td>0.23 sec</td>
</tr>
<tr>
<td>142/146</td>
<td>GUI Suite/RectDetectorGeneric</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>143/146</td>
<td>GUI Suite/RectDetectorPerpToSample</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>144/146</td>
<td>GUI Suite/RectDetectorPerpToDirectBeam</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>145/146</td>
<td>GUI Suite/RectDetectorPerpToReflectedBeam</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>146/146</td>
<td>GUI Suite/RectDetectorPerpToReflectedBeamDpos</td>
<td>Passed</td>
<td>0.06 sec</td>
</tr>
</tbody>
</table>

100% tests passed, 0 tests failed out of 146

Total Test time (real) = 58.81 sec

[100%] Built target check
Testing

Functional tests for Core/GUI/Python domains

factory → Core simulation → Intensity Data → Numeric Difference? → Reference Data
Functional tests for Core/GUI/Python domains

- Core simulation
- Python script
- Embedded Python
- Intensity Data
- Numeric Difference?

factory

Generate text of Python script

Run
Testing

Functional tests for Core/GUI/Python domains

1. **Core simulation**
   - Generate text of Python script
   - Reference Data

2. **Reference Data**
   - Run

3. **Intensities**
   - Run
   - Numeric Difference?

4. **Core simulation**
   - Run

5. **GUI model**
   - Generate GUI representation

6. **Python script**
   - Factory
   - Generate text of Python script

7. **Embedded Python**
   - Run
IsGISAXS as a starting point:

- Successful software which is a de facto standard in the user community

  **IsGISAXS**: a program for grazing-incidence small-angle X-ray scattering analysis of supported islands  
  R Lazzari - Journal of Applied Crystallography, 2002 - scripts.iucr.org  
  This paper describes a Fortran program, IsGISAXS, for the simulation and analysis of grazing-incidence small-angle X-ray scattering (GISAXS) of islands supported on a substrate. As is usual in small-angle scattering of particles, the scattering cross section is ...
  Cited by 257  Related articles  All 7 versions  Cite

- Simulation in DWBA
- FORTRAN 90, 13k lines of code
- No longer actively supported
Agile development

- Workflow consists of sprint cycles every 4-6 weeks during which the team create finished portions of product.

Roadmap

**Sprint 24**

25 days running (15 Sep 2014)
User requests, toward beta of GUI in October.

| 15 closed (49%) | 16 open (52%) |

Related issues:
- Bug #776: GUI: interference function2DParaCrystal rotation angle activation
- Bug #780: Windows: pmmap of item being dragged is not displayed on DesignerScene
- Bug #788: Fix release script to process CHANGELOG correctly
- Bug #791: LLDataTest, DataAssignment Unit test failure
- Bug #823: Remove interference function approximations from GUI
- Bug #826: cmake fails under Debian/testing: problem with Python
- Bug #829: CMake is not able to find right Python version when there is a Python2 and Python
- Feature #393: Create Mac installer
- Feature #586: Investigate chi2-like objective functions
- Feature #677: Provide validation of GUI sample for correctness and corresponding info widget
- Feature #758: Provide recording of stack trace in crashing GUI application
- Feature #759: Integrate QuickSimulationView into JobView
- Feature #769: Remove SimuleonDataModel
- Feature #775: Windows installer: implement add/remove BornAgain desktop icon
- Feature #785: Design BornAgain main application icon
- Feature #784: Revise workspace behaviour in DesignerScene
- Feature #802: Implement correct handling of simulation failure in JobItem
- Feature #805: Implement simple crash handler widget to report bugs
- Feature #806: Implement crash handler manager to launch external executable in platform independent stack trace retrieval
- Feature #914: Implement exceptions catching in the Core to report exception from a thread
to the GUI
- Feature #920: Implement reset of JobItem's sample and instrument models to the original.
- Feature #822: Revise submit job logic
- Feature #823: Allow multiple Layout objects per layer
- Feature #825: Update default behaviour of OutputDataWidget
- Feature #928: Trivial form factor for demonstration purposes
- Documentation #487: Provide screenshots for project homepage
- Documentation #781: Provide short description of GUI functionality
- Refactoring #786: Remove unnecessary calls to getOutCoefficients
- Refactoring #915: Review SimulationParameters

Backlog:
- item121
- item120
- ...
- item95
- item93

Sprint:
- item121
- Item95
- item93

1 month
<table>
<thead>
<tr>
<th>#</th>
<th>Tracker</th>
<th>Status</th>
<th>Priority</th>
<th>Subject</th>
<th>Assignee</th>
<th>Target version</th>
<th>% Done</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>1371</td>
<td>Bug</td>
<td>New</td>
<td>Normal</td>
<td>presence of some boost components not checked by cmake</td>
<td></td>
<td></td>
<td></td>
<td>09 Mar 2016 10:58</td>
</tr>
<tr>
<td>1370</td>
<td>Bug</td>
<td>Sprint</td>
<td>Normal</td>
<td>Fix numerous &quot;features&quot; introduced by latest major GUI refactoring</td>
<td>david</td>
<td>Sprint 31</td>
<td></td>
<td>08 Mar 2016 17:24</td>
</tr>
<tr>
<td>1366</td>
<td>Refactoring</td>
<td>Backlog</td>
<td>Normal</td>
<td>Revise boost libraries usage</td>
<td></td>
<td></td>
<td></td>
<td>04 Mar 2016 13:55</td>
</tr>
<tr>
<td>1363</td>
<td>Envelope task</td>
<td>In Progress</td>
<td>Urgent</td>
<td>Unix build tasks</td>
<td></td>
<td></td>
<td></td>
<td>03 Mar 2016 13:56</td>
</tr>
<tr>
<td>1362</td>
<td>Envelope task</td>
<td>In Progress</td>
<td>Normal</td>
<td>Mac build tasks</td>
<td></td>
<td></td>
<td></td>
<td>03 Mar 2016 13:56</td>
</tr>
<tr>
<td>1361</td>
<td>Envelope task</td>
<td>In Progress</td>
<td>Normal</td>
<td>Win build tasks</td>
<td></td>
<td></td>
<td></td>
<td>03 Mar 2016 13:56</td>
</tr>
<tr>
<td>1360</td>
<td>Documentation</td>
<td>New</td>
<td>Normal</td>
<td>reequilibrates hierarchy levels in online docs</td>
<td></td>
<td></td>
<td></td>
<td>03 Mar 2016 13:28</td>
</tr>
<tr>
<td>1351</td>
<td>Documentation</td>
<td>Sprint</td>
<td>Normal</td>
<td>Drupal: update installation instructions, tutorials for coming release 1.6</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>19 Feb 2016 13:54</td>
</tr>
<tr>
<td>1350</td>
<td>Testing</td>
<td>Sprint</td>
<td>Normal</td>
<td>Buildbot: provide set of configurations for buildbot-based BornAgain's builds</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>19 Feb 2016 13:46</td>
</tr>
<tr>
<td>1349</td>
<td>Testing</td>
<td>Sprint</td>
<td>Normal</td>
<td>Buildbot: provide tutorial how to add new configuration</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>19 Feb 2016 13:38</td>
</tr>
<tr>
<td>1348</td>
<td>Testing</td>
<td>Sprint</td>
<td>Normal</td>
<td>Buildbot: install agent on scgmini and attach Mavericks/Yosemite vagrant boxes</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>19 Feb 2016 13:37</td>
</tr>
<tr>
<td>1342</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>GUI: add Monte-Carlo integration option in the simulation</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>18 Feb 2016 17:30</td>
</tr>
<tr>
<td>1334</td>
<td>Refactoring</td>
<td>Backlog</td>
<td>Normal</td>
<td>Core: remove ProgramOptions from the simulation</td>
<td></td>
<td></td>
<td></td>
<td>11 Feb 2016 17:32</td>
</tr>
<tr>
<td>1333</td>
<td>Refactoring</td>
<td>Sprint</td>
<td>Normal</td>
<td>MSC switches hopefully obsolete</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>11 Feb 2016 14:48</td>
</tr>
<tr>
<td>1308</td>
<td>Feature</td>
<td>Backlog</td>
<td>Normal</td>
<td>GUI: take care about margins in ColorMapPlot</td>
<td></td>
<td></td>
<td></td>
<td>08 Feb 2016 10:31</td>
</tr>
<tr>
<td>1305</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>GUI: Make real time simulation aware of current zoom level to speed up the performance</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>05 Feb 2016 15:07</td>
</tr>
<tr>
<td>1304</td>
<td>Refactoring</td>
<td>New</td>
<td>Normal</td>
<td>Unify treatment of numeric constants</td>
<td></td>
<td></td>
<td></td>
<td>04 Feb 2016 11:21</td>
</tr>
<tr>
<td>1301</td>
<td>Envelope task</td>
<td>In Progress</td>
<td>Urgent</td>
<td>Pre-release actions</td>
<td></td>
<td></td>
<td></td>
<td>02 Feb 2016 19:35</td>
</tr>
<tr>
<td>1296</td>
<td>Documentation</td>
<td>Sprint</td>
<td>Normal</td>
<td>update internal information about performance tests</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>02 Feb 2016 15:02</td>
</tr>
<tr>
<td>1294</td>
<td>Bug</td>
<td>New</td>
<td>Normal</td>
<td>provide substantial unit tests for factor computations</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>02 Feb 2016 14:27</td>
</tr>
<tr>
<td>1293</td>
<td>Bug</td>
<td>Backlog</td>
<td>High</td>
<td>bold math symbols broken under Texlive2015</td>
<td>wuttke</td>
<td></td>
<td></td>
<td>02 Feb 2016 14:22</td>
</tr>
<tr>
<td>1291</td>
<td>Refactoring</td>
<td>Sprint</td>
<td>Normal</td>
<td>core functional test machinery: simplify, or at least explain</td>
<td></td>
<td>Sprint 31</td>
<td></td>
<td>02 Feb 2016 11:47</td>
</tr>
<tr>
<td>1289</td>
<td>Envelope task</td>
<td>In Progress</td>
<td>Normal</td>
<td>Closeup tasks - to keep the code base readable and maintainable</td>
<td></td>
<td></td>
<td></td>
<td>02 Feb 2016 11:29</td>
</tr>
</tbody>
</table>
**Conclusion**

**BornAgain key features**

- Open source, multi platform
- X-rays and polarized neutrons
- Community friendly and community driven
- Software architecture is flexible and allows extension
- Professional approach to the software development