The making of BornAgain

Jan Burle, Céline Durniak, Jonathan Fisher,
Marina Ganeva, David Li , Walter Van Herck,
Gennady Pospelov, Joachim Wuttke

Scientific Computing Group at MLZ
Jülich Center for Neutron Science

JCNS Jülich Centre for Neutron Science

MLZ Heinz Maier-Leibnitz Zentrum

SINE2020 Coordination Meeting, Villigen PSI, April 5, 2016

The making of BornAgain

Jan Burle, Céline Durniak, Jonathan Fisher,
Marina Ganeva, David Li , Walter Van Herck,
Gennady Pospelov, Joachim Wuttke

Scientific Computing Group at MLZ
Jülich Center for Neutron Science

Jülich Centre for Neutron Science

Heinz Maier-Leibnitz Zentrum

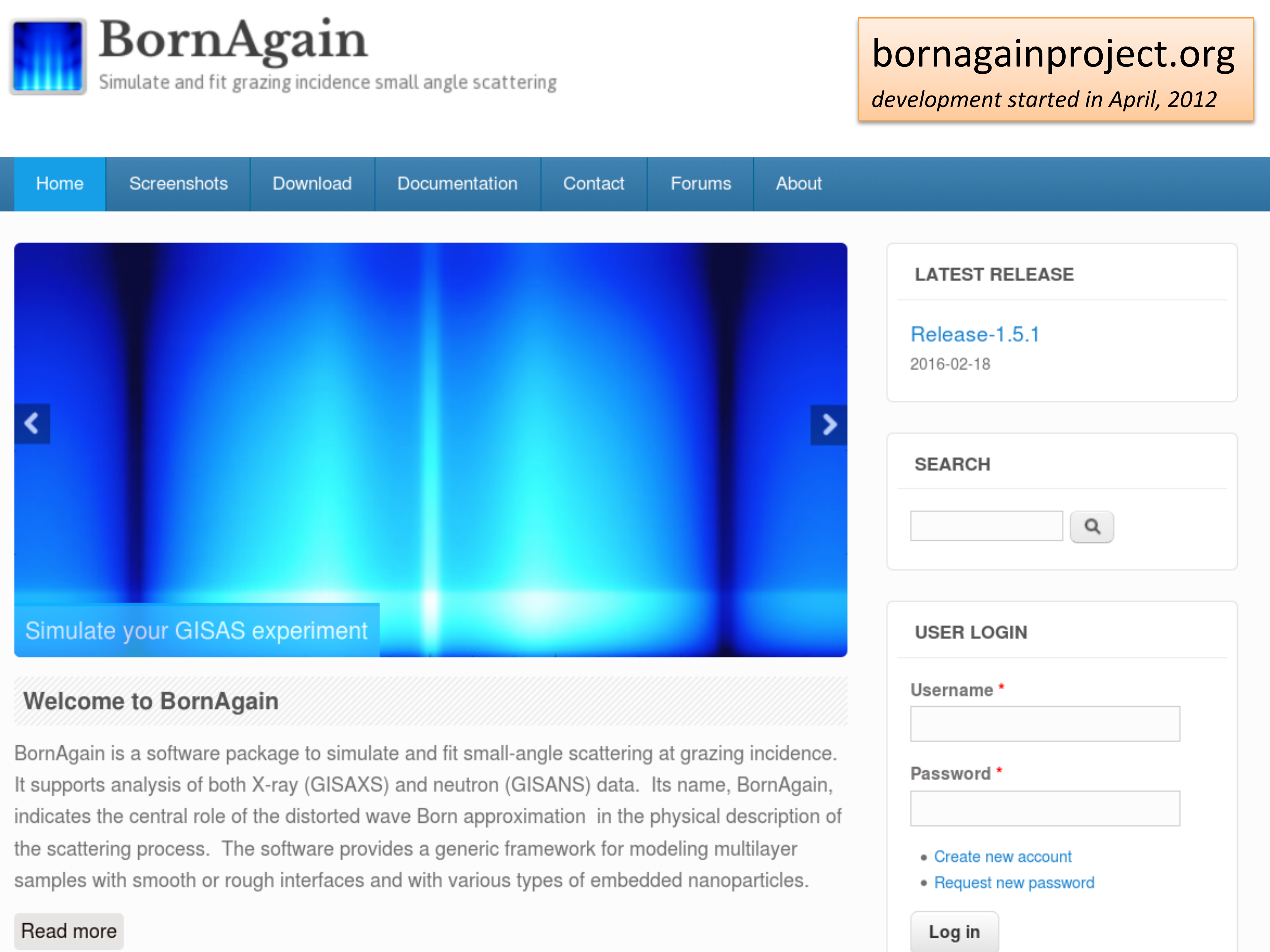
SINE2020 Coordination Meeting, Villigen PSI, April 5, 2016

Outline

- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

bornagainproject.org

development started in April, 2012



BornAgain

Simulate and fit grazing incidence small angle scattering

Simulate your GISAS experiment

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.

LATEST RELEASE

Release-1.5.1

2016-02-18

SEARCH

USER LOGIN

Username *

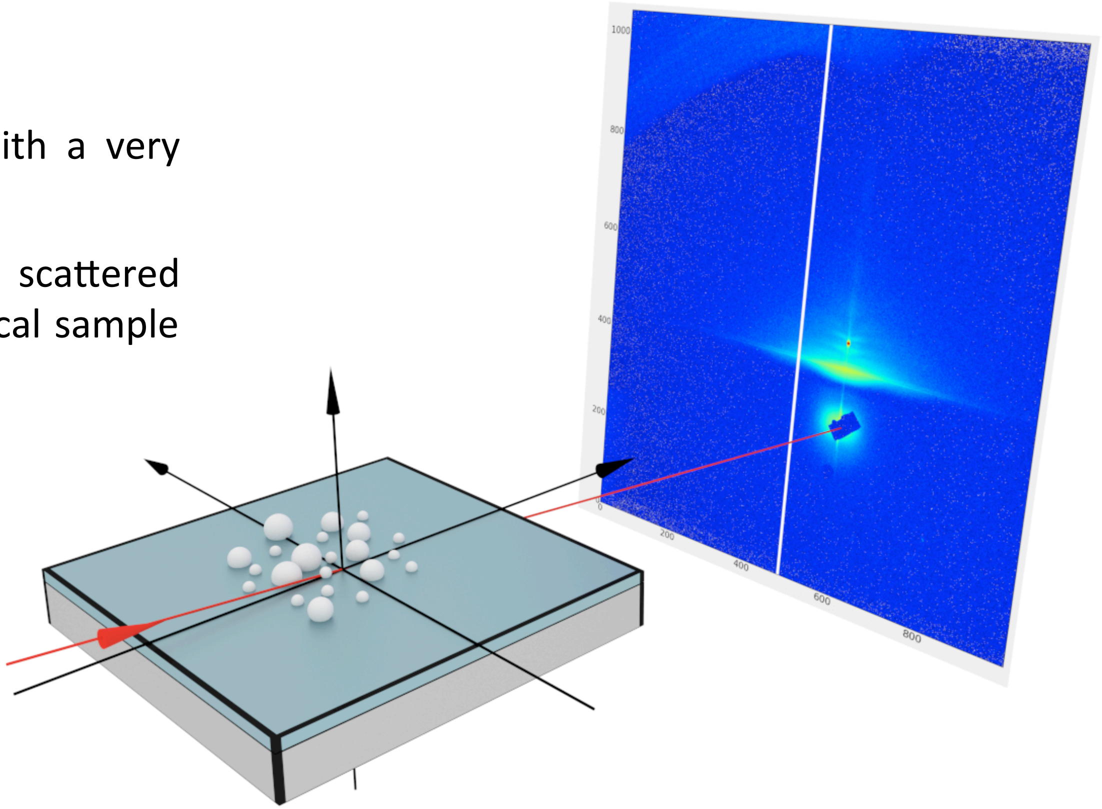
Password *

- Create new account
- Request new password

Grazing Incidence Small Angle Scattering

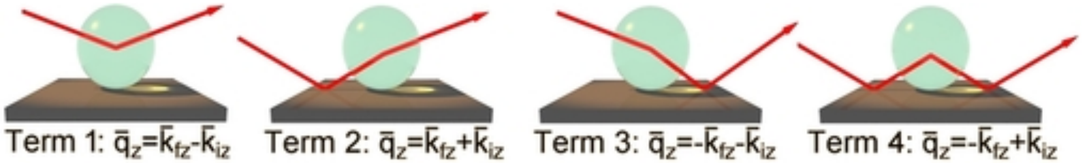
Experiment

- 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



Term 1: $\bar{q}_z=\bar{k}_{fz}-\bar{k}_{iz}$ Term 2: $\bar{q}_z=\bar{k}_{fz}+\bar{k}_{iz}$ Term 3: $\bar{q}_z=-\bar{k}_{fz}-\bar{k}_{iz}$ Term 4: $\bar{q}_z=-\bar{k}_{fz}+\bar{k}_{iz}$

$$\frac{d\sigma}{d\Omega} = \left\langle \left|F_{DWBA}\right|^2 \right\rangle S(q_\parallel)$$

Motivation

Support for specific instruments at MLZ

- ○ Serve our users, support in house research, at Maria and REFSANS instruments

Limited functionality of existing software

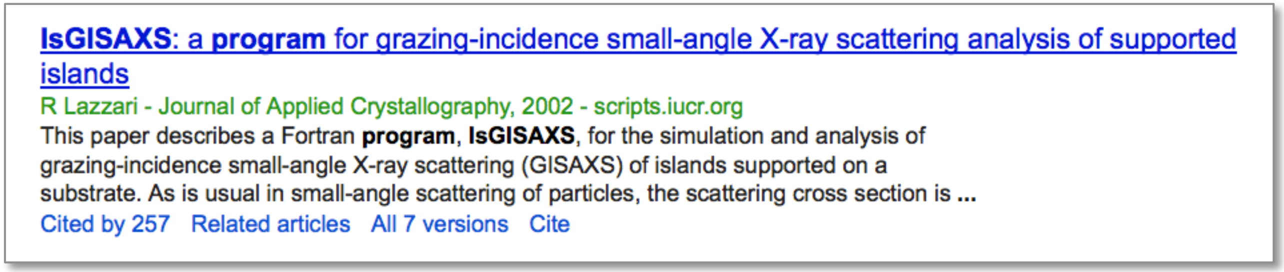
- ○ No support for polarized neutrons, limitations in sample geometry
- ○ Usability issues, lack of support

High Data Rate Processing and Analysis initiative (HDRI)

- ○ Call to create simulation software for non-expert users for GISAS field
- ○ Provide functionality/extensibility for broader usage

IsGISAXS as an example

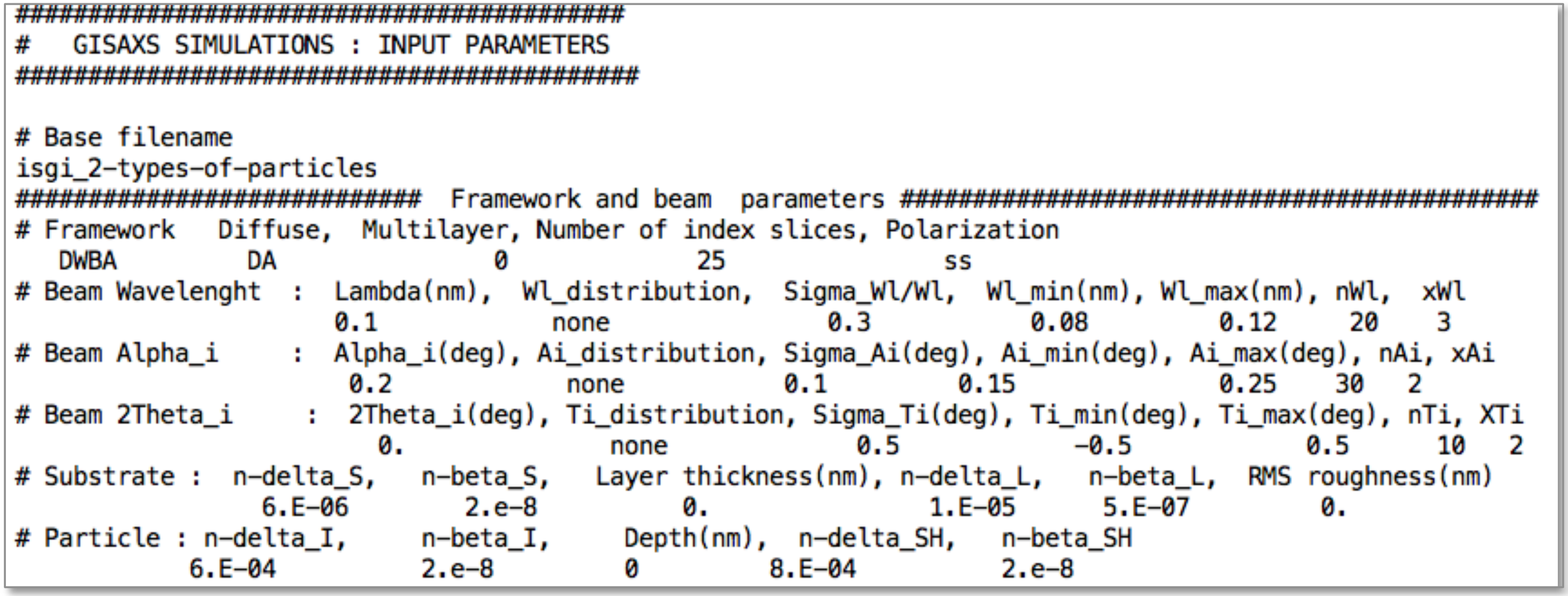
- Successful software which is a de facto standard in the user community



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

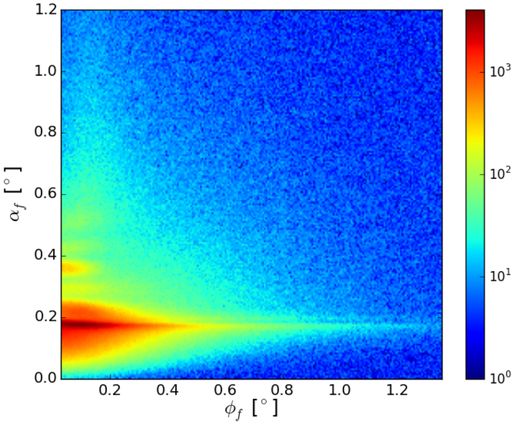
IsGISAXS parameter file

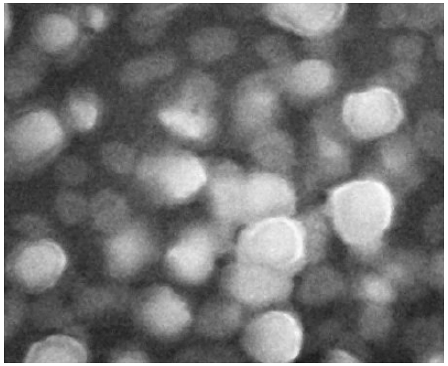


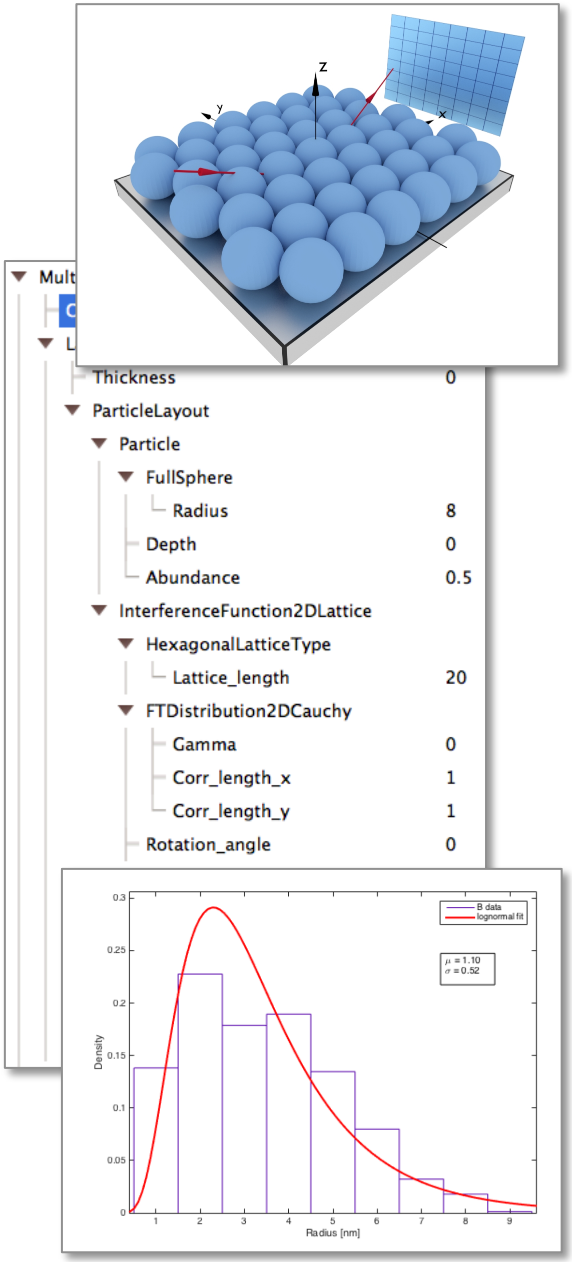
```
###########################################
#   GISAXS SIMULATIONS : INPUT PARAMETERS
###########################################

# Base filename
isgi_2-types-of-particles
############################  Framework and beam  parameters ###########################################
# Framework   Diffuse,  Multilayer, Number of index slices, Polarization
   DWBA         DA              0              25                ss
# Beam Wavelenght  :  Lambda(nm),  Wl_distribution,  Sigma_Wl/Wl,  Wl_min(nm), Wl_max(nm), nWl,  xWl
                      0.1              none              0.3          0.08         0.12      20    3
# Beam Alpha_i     :  Alpha_i(deg), Ai_distribution, Sigma_Ai(deg), Ai_min(deg), Ai_max(deg), nAi, xAi
                       0.2              none          0.1          0.15              0.25     30    2
# Beam 2Theta_i     :  2Theta_i(deg), Ti_distribution, Sigma_Ti(deg), Ti_min(deg), Ti_max(deg), nTi, XTi
                         0.              none              0.5            -0.5          0.5      10    2
# Substrate :  n-delta_S,    n-beta_S,    Layer thickness(nm), n-delta_L,    n-beta_L,  RMS roughness(nm)
                 6.E-06         2.e-8         0.                1.E-05       5.E-07          0.
# Particle : n-delta_I,      n-beta_I,      Depth(nm),  n-delta_SH,     n-beta_SH
            6.E-04          2.e-8          0           8.E-04          2.e-8
```

- Introduction
- **Software architecture**
- Demonstration
- Under the hood
- Closing remarks

User needs

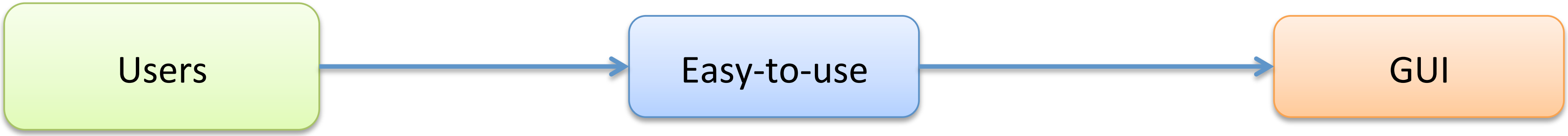




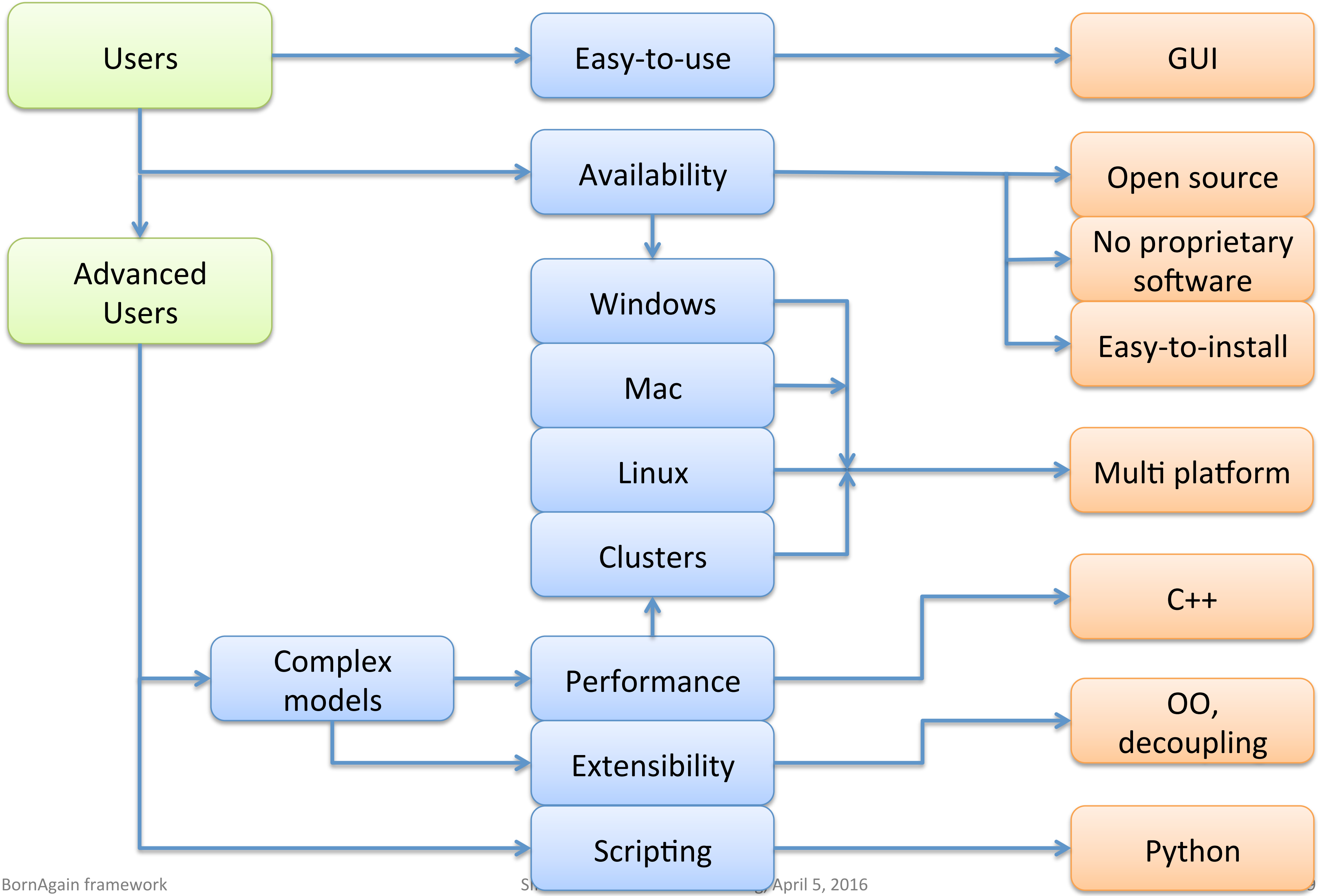


- Mult
  - C
  - L
    - Thickness 0
    - ParticleLayout
      - Particle
        - FullSphere
          - Radius 8
        - Depth 0
        - Abundance 0.5
      - InterferenceFunction2DLattice
        - HexagonalLatticeType
          - Lattice_length 20
        - FTDistribution2DCauchy
          - Gamma 0
          - Corr_length_x 1
          - Corr_length_y 1
        - Rotation_angle 0

User needs



Requirements



Users

Easy-to-use

GUI

Availability

Open source

No proprietary software

Easy-to-install

Advanced Users

Windows

Mac

Linux

Clusters

Multi platform

Complex models

Performance

C++

Extensibility

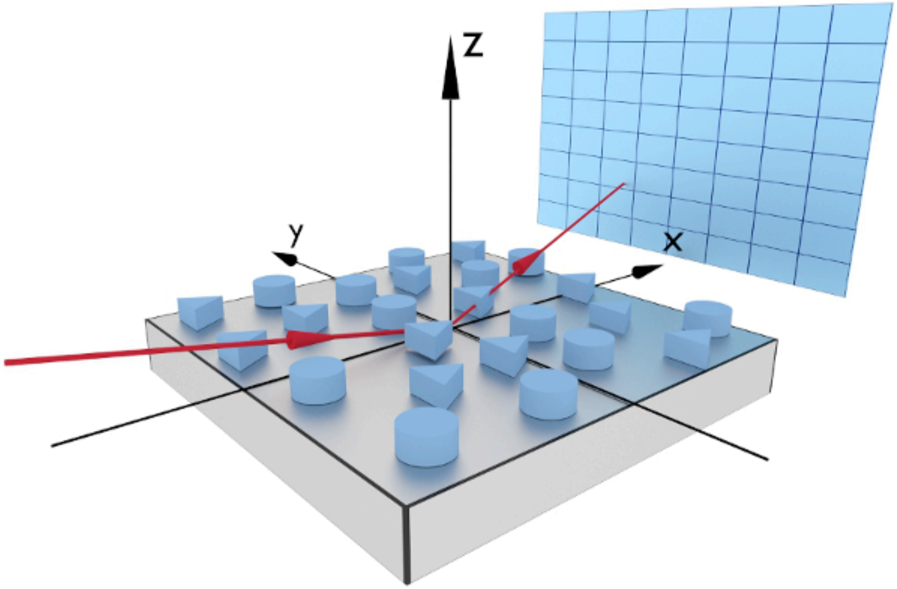
OO, decoupling

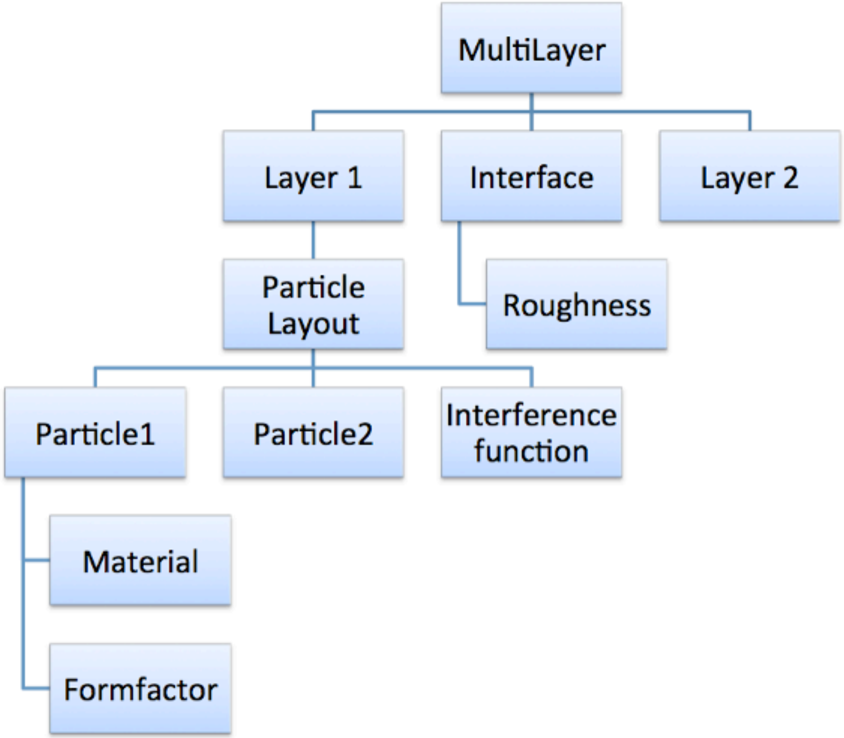
Scripting

Python

Software Architecture

- Open-source framework written in C++, interfaced with Python
  - distributed under GPL3 license
- Multi-platform
  - Unix flavors, source code
  - Windows, binary installer package
  - Mac OS, binary installer package
- Object-oriented approach for sample description





MultiLayer

Layer 1

Interface

Layer 2

Particle Layout

Roughness

Particle1

Particle2

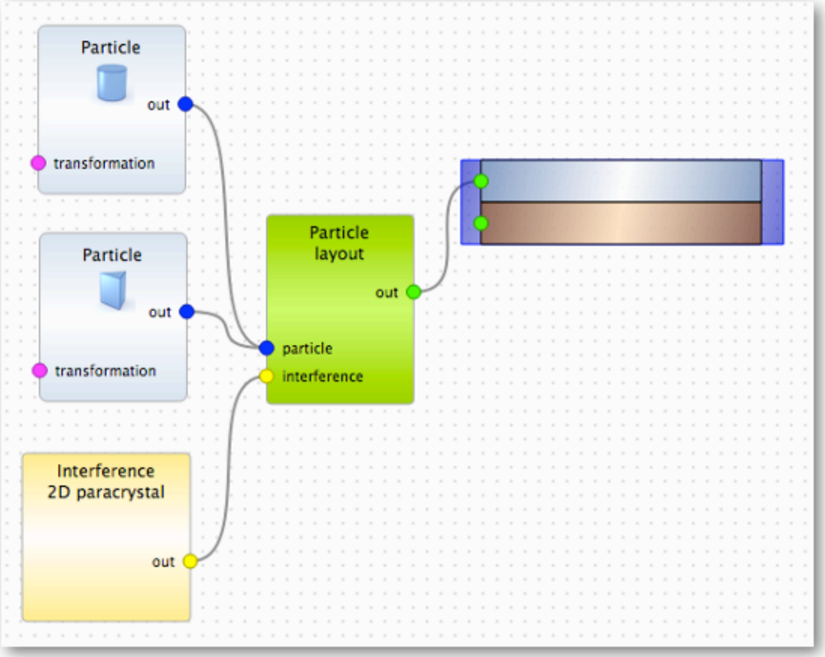
Interference function

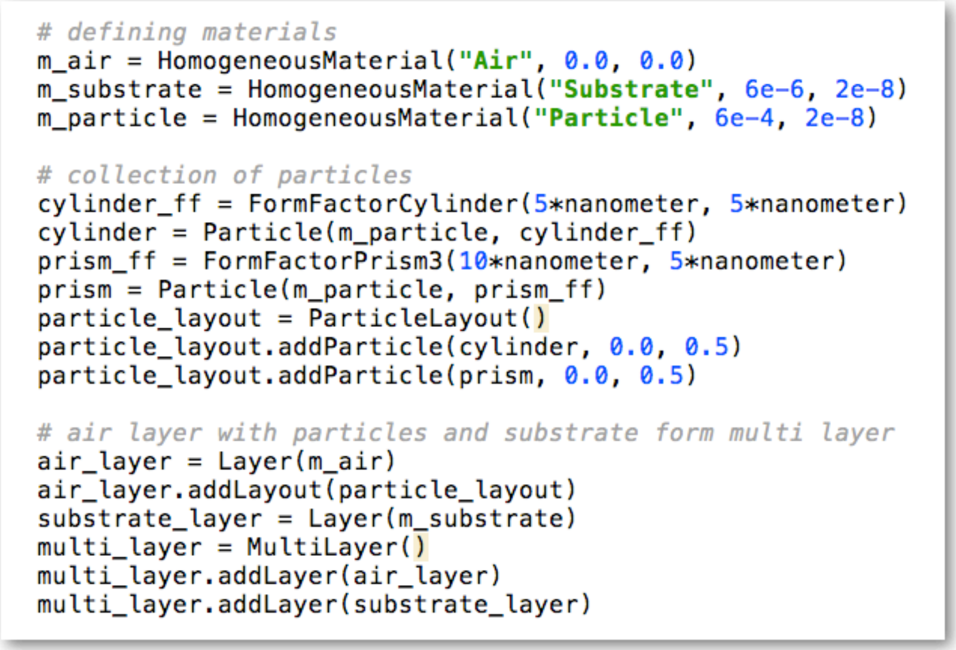
Material

Formfactor

Software Architecture

- Open-source framework written in C++, interfaced with Python
  - distributed under GPL3 license
- Multi-platform
  - Unix flavors, source code
  - Windows, binary installer package
  - Mac OS, binary installer package
- Object-oriented approach for sample description

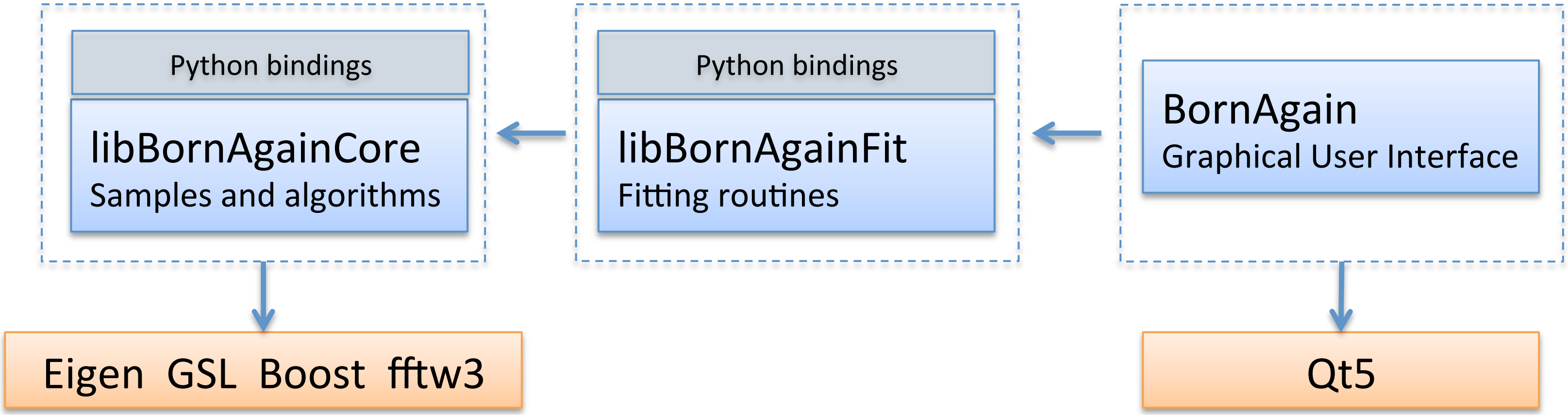




```
# defining materials
m_air = HomogeneousMaterial("Air", 0.0, 0.0)
m_substrate = HomogeneousMaterial("Substrate", 6e-6, 2e-8)
m_particle = HomogeneousMaterial("Particle", 6e-4, 2e-8)

# 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)
```

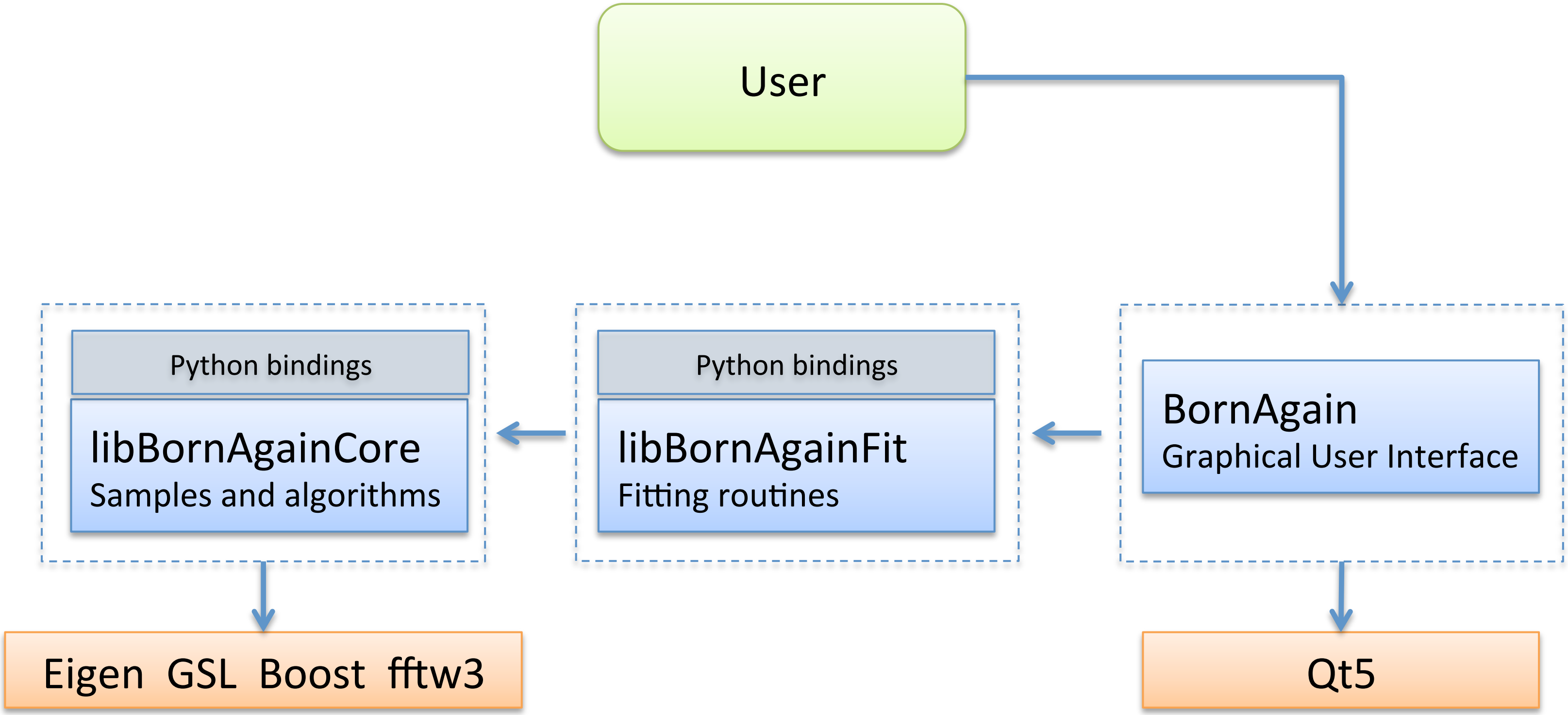
Package structure

- C++ kernel for simulation description and fitting
- Python bindings
- Graphical User Interface
- Well established Open Source libraries as external dependencies
- CMake based



Working with BornAgain

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



User

Python bindings

libBornAgainCore

Samples and algorithms

Python bindings

libBornAgainFit

Fitting routines

BornAgain

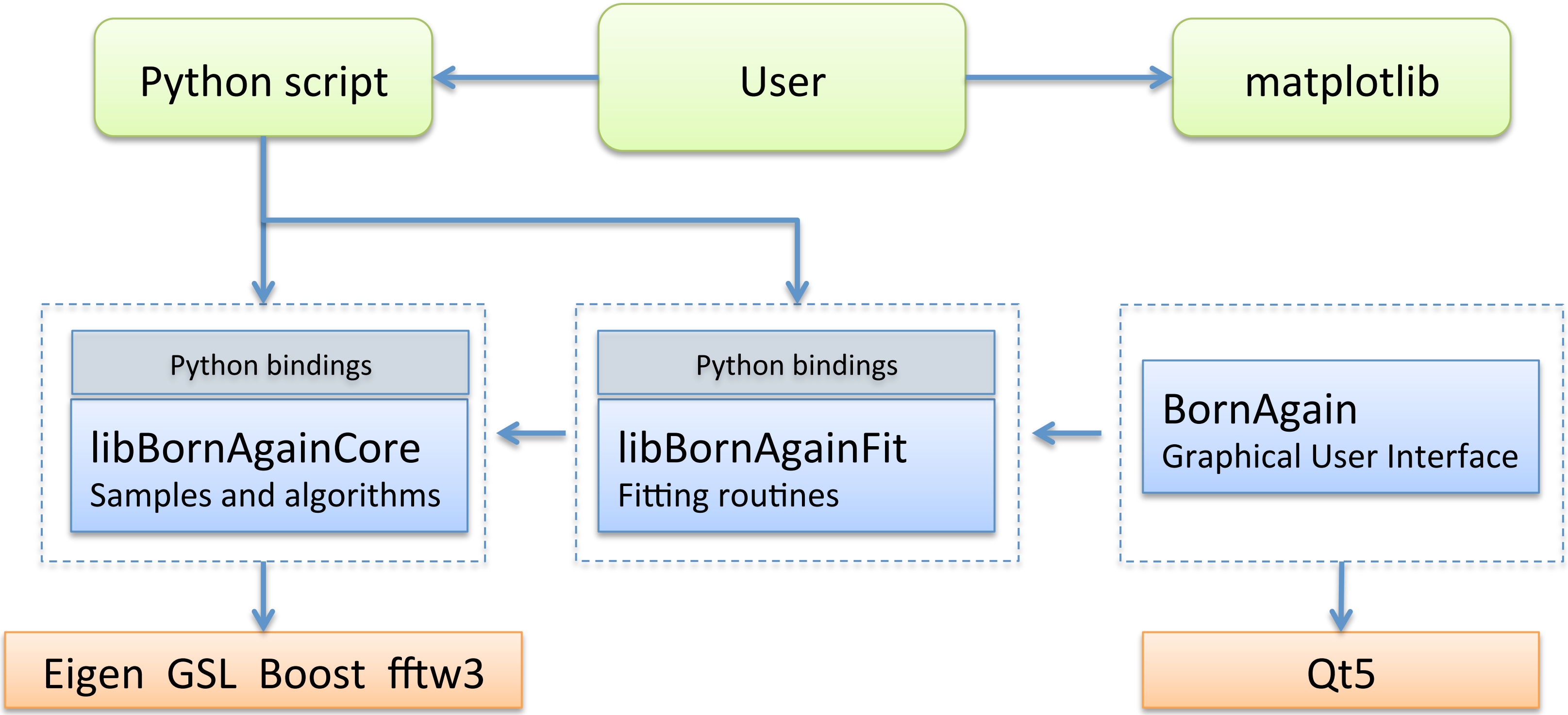
Graphical User Interface

Eigen GSL Boost fftw3

Qt5

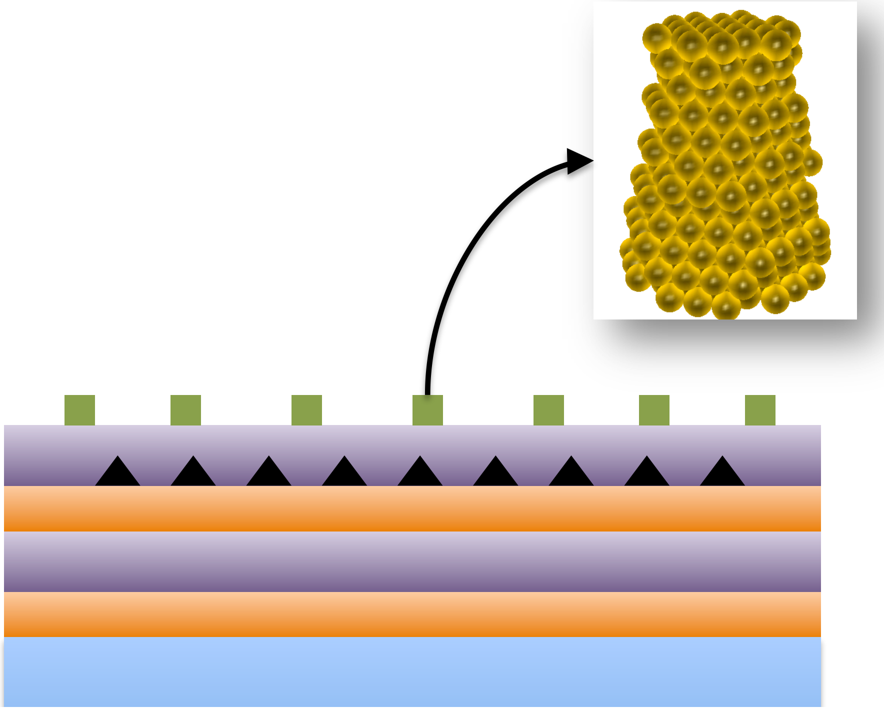
Working with BornAgain

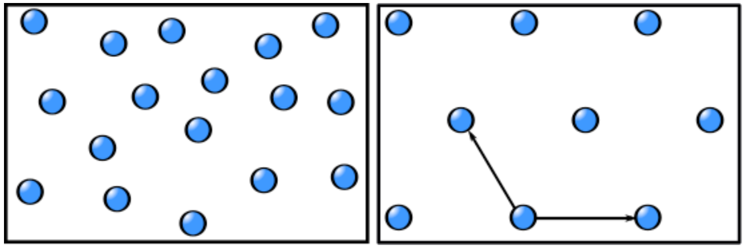
- Using Graphical User Interface
- Running Python script with simulation description

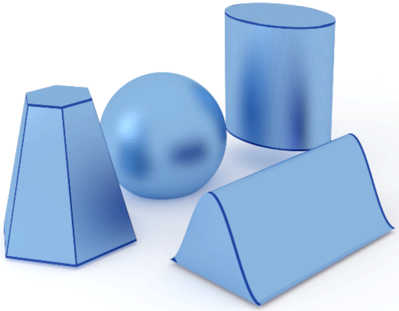


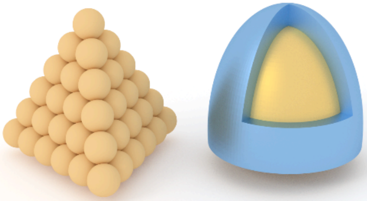
Functionality

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







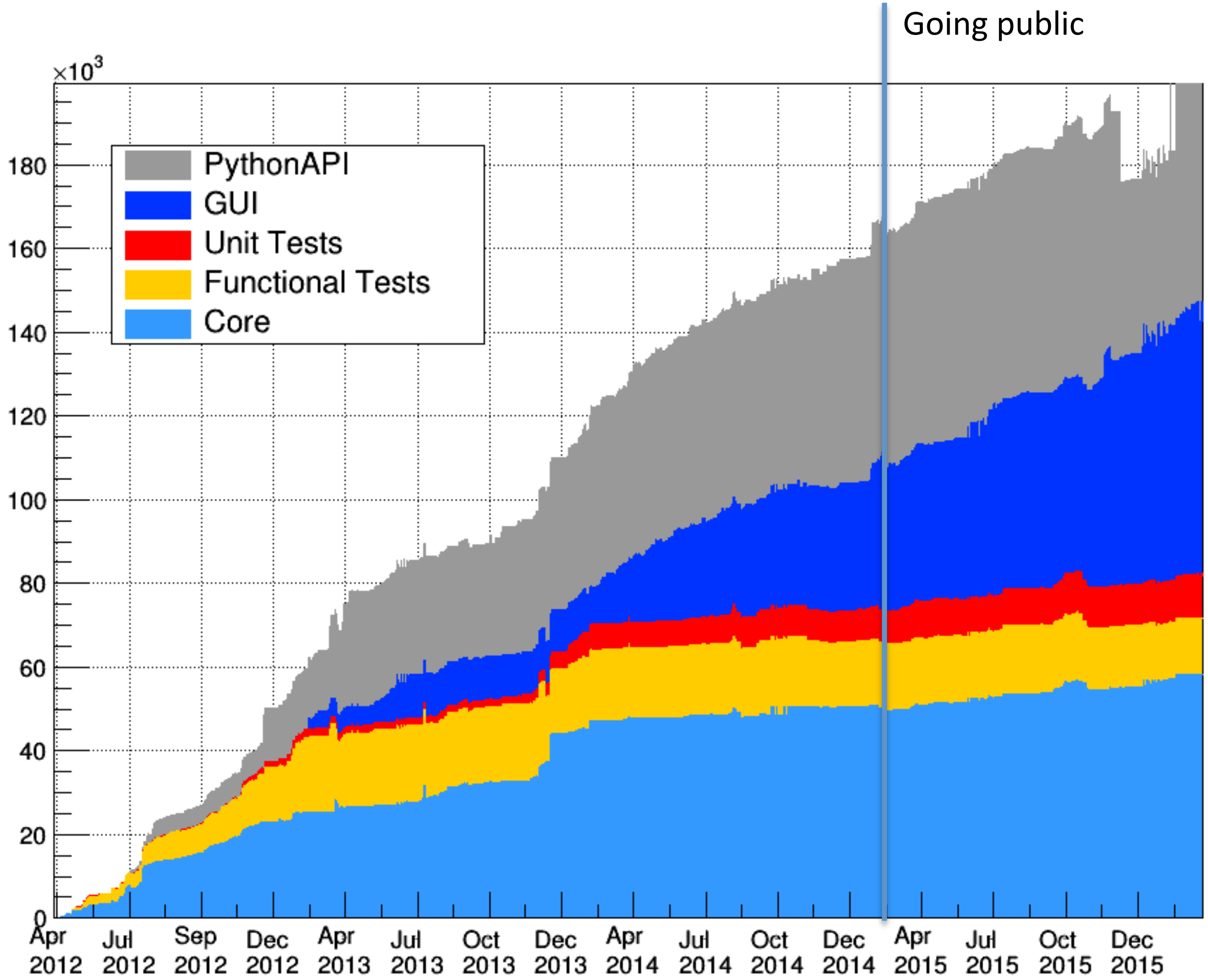


- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

Under the hood

Number of lines of code vs time



Development tools

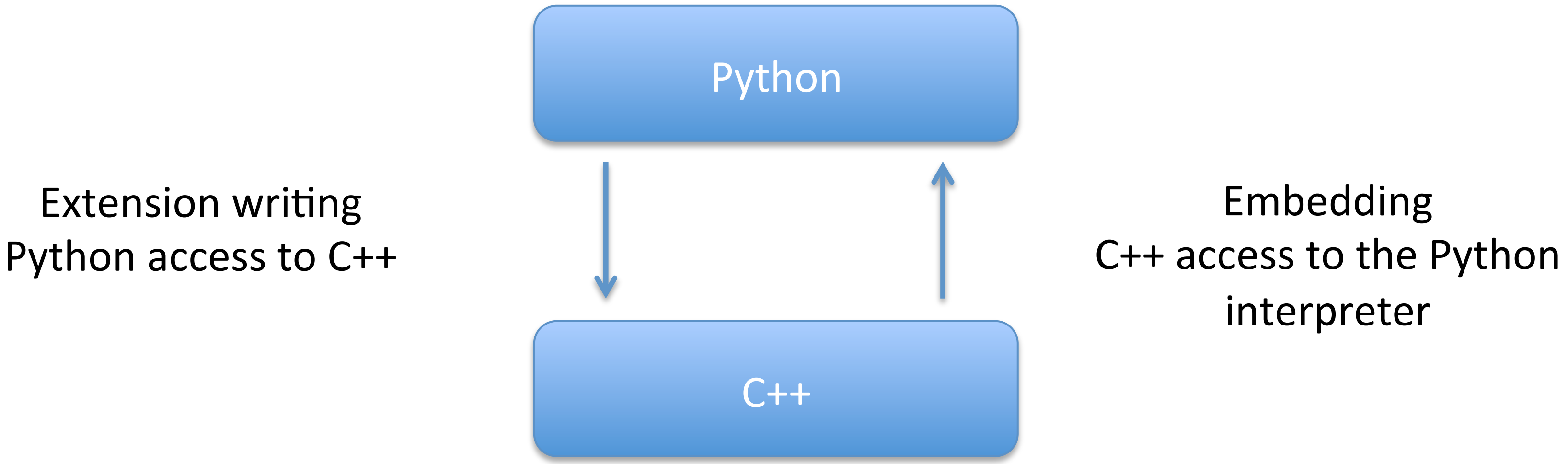
- Version control system (`git`)
- Issue tracking (`redmine`)
- Nightly build, CI (`teamcity -> docker + vagrant + buildbot`)
- Unit tests (`googletest, QtTest`)
- Functional tests
- Release procedure

- Other
  - Google analytics
  - slack
  - Doxygen
  - Valgrind, Coverity, MacOS/Instruments
  - Blender/Inkscape

- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

Python bindings

C++/Python relationship



Python

C++

Extension writing
Python access to C++

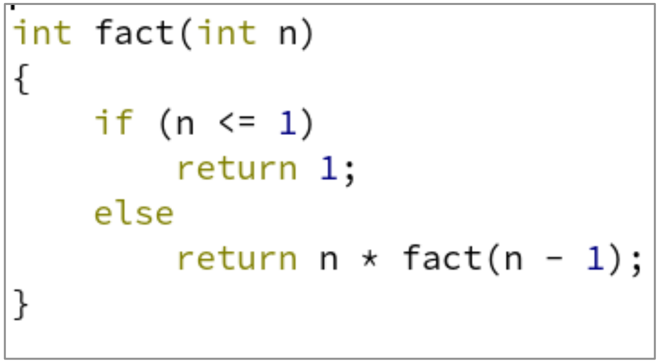
Embedding
C++ access to the Python interpreter

Python bindings

Wrapper function

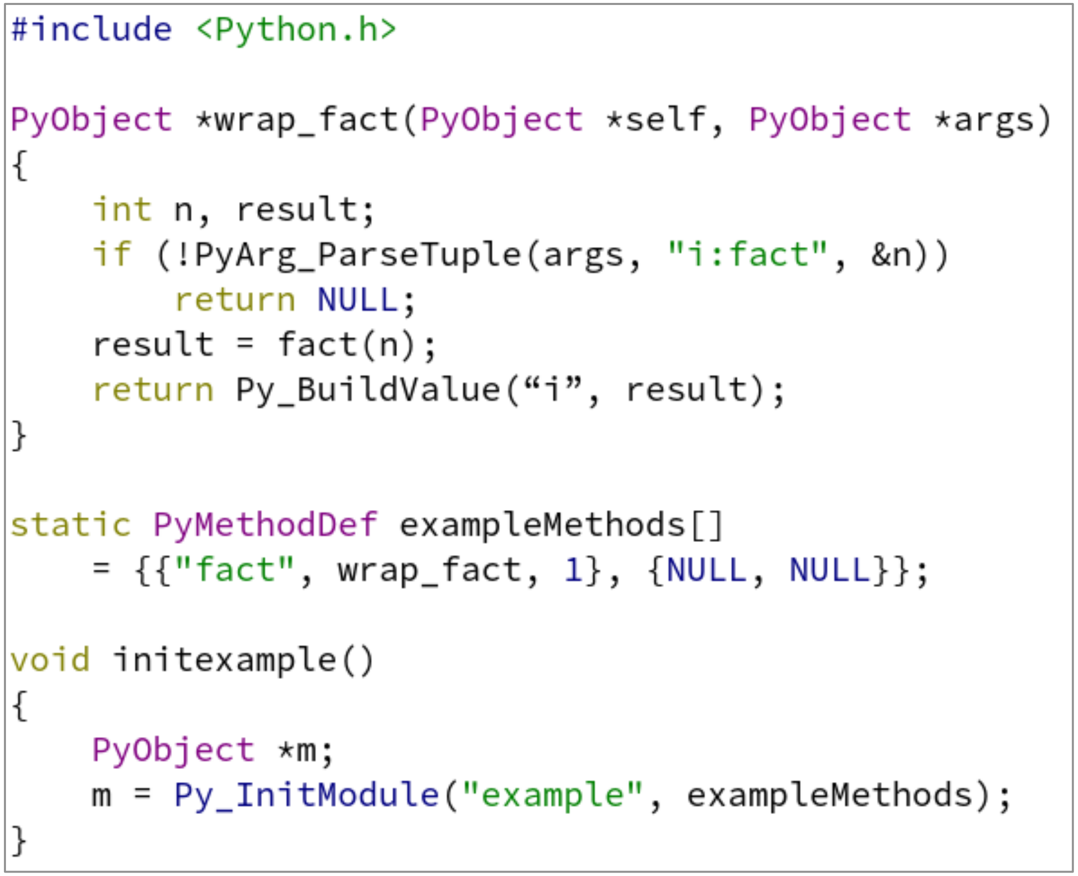
- Converts function arguments from Python to C, returns results in Python expected form
- Has to be registered for Python interpreter

function.c



```
int fact(int n)
{
    if (n <= 1)
        return 1;
    else
        return n * fact(n - 1);
}
```

wrapper.c



```
#include <Python.h>

PyObject *wrap_fact(PyObject *self, PyObject *args)
{
    int n, result;
    if (!PyArg_ParseTuple(args, "i:fact", &n))
        return NULL;
    result = fact(n);
    return Py_BuildValue(“i”, result);
}

static PyMethodDef exampleMethods[]
    = {{"fact", wrap_fact, 1}, {NULL, NULL}};

void initexample()
{
    PyObject *m;
    m = Py_InitModule("example", exampleMethods);
}
```

Python bindings

Choosing technology to wrap a complex C/C++ application

- External dependencies?
- What is the performance?
- Build system integration?
- Is wrapping code on Python side or on C++ side?
- How much code should be written additionally?
- Should I affect or duplicate existing C++ code?
- How big is the community?
- Is it possible to fully automate wrappers generation?
- Do I need bindings with another languages?

After careful consideration we have chosen

boost::python

Python bindings

Choosing technology to wrap a complex C/C++ application

- External dependencies?
- What is the performance?
- Build system integration?
- Is wrapping code on Python side or on C++ side?
- How much code should be written additionally?
- Should I affect or duplicate existing C++ code?
- How big is the community?
- Is it possible to fully automate wrappers generation?
- Do I need bindings with another languages?

Difficulties with C++11, accompanying code generators (gccxml, Py++) were obsolete.

After careful consideration we have chosen
boost::python

3 years later...
After careful consideration we have switched to
SWIG

Python bindings

SWIG bindings in BornAgain (starting from next release 1.6)

- Binding generation is governed by a SWIG interface file

```
%{
#include "ISample.h"
%}
%include "ISample.h"
%feature("director") ISample;
```

`libBornAgainCore.i`

- Interface file can be fine-tuned to ignore certain methods of classes or tweak existing one
  - No change to the original C++ code is required

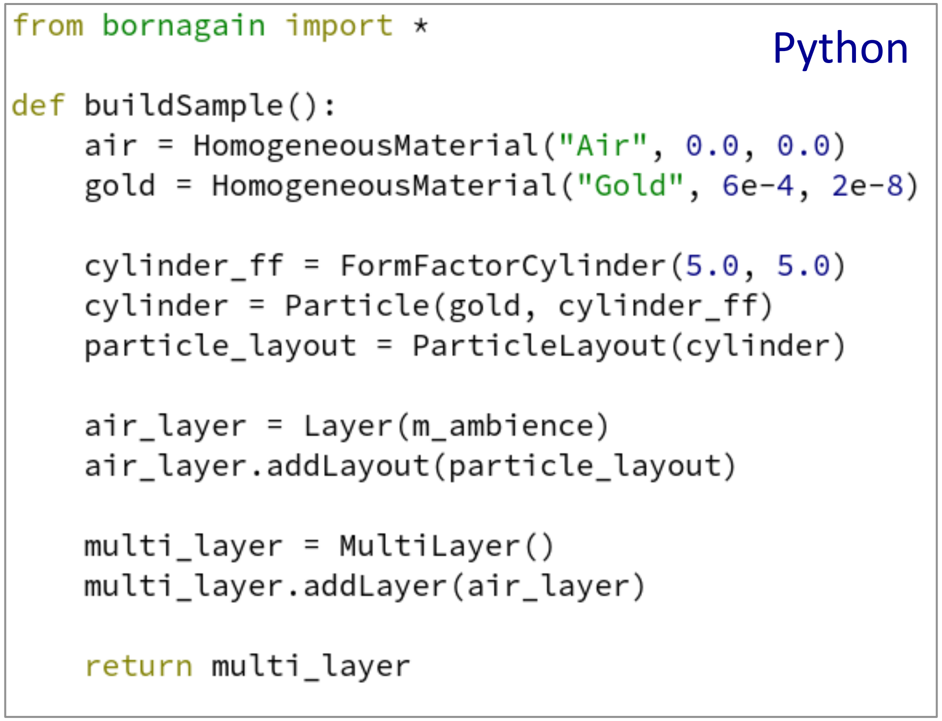
- Generation of bindings is done via `swig` executable

```
$ swig libBornAgainCore.i
```

Produces additionally 130k lines of C++,
25k lines of Python

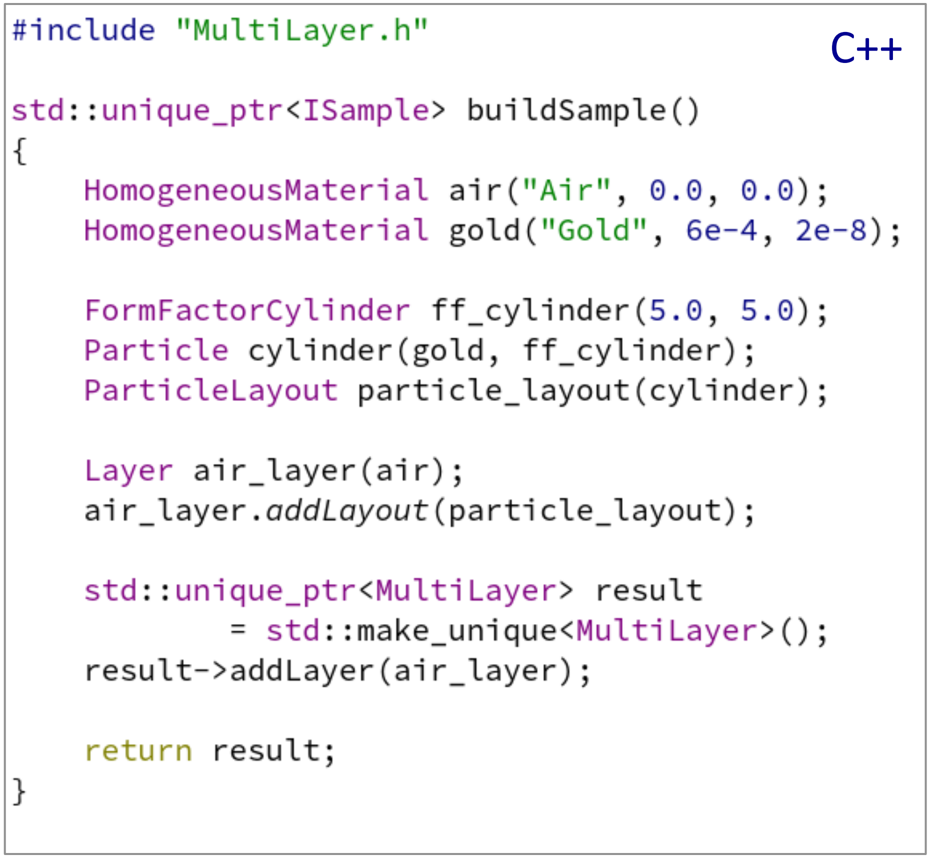
Python bindings

Achieved results



Python

```python
from bornagain import *

def buildSample():
    air = HomogeneousMaterial("Air", 0.0, 0.0)
    gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

    cylinder_ff = FormFactorCylinder(5.0, 5.0)
    cylinder = Particle(gold, cylinder_ff)
    particle_layout = ParticleLayout(cylinder)

    air_layer = Layer(m_ambience)
    air_layer.addLayout(particle_layout)

    multi_layer = MultiLayer()
    multi_layer.addLayer(air_layer)

    return multi_layer
```



C++

```cpp
#include "MultiLayer.h"

std::unique_ptr<ISample> buildSample()
{
    HomogeneousMaterial air("Air", 0.0, 0.0);
    HomogeneousMaterial gold("Gold", 6e-4, 2e-8);

    FormFactorCylinder ff_cylinder(5.0, 5.0);
    Particle cylinder(gold, ff_cylinder);
    ParticleLayout particle_layout(cylinder);

    Layer air_layer(air);
    air_layer.addLayout(particle_layout);

    std::unique_ptr<MultiLayer> result
            = std::make_unique<MultiLayer>();
    result->addLayer(air_layer);

    return result;
}
```

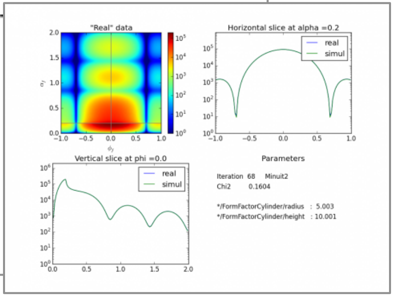
Python bindings

Achieved results

- Supports both Python 2.7 and 3
- Generated code is portable (compiles with gcc, clang and Visual Studio)
- Supports shared ownership, transfer of ownership
- Automatic conversion between many C++ types/containers and those on Python side
  - `std::string/Python string, std::vector/Python list, std::map/Python dict`
- Allows custom conversions
  - `vector<vector<double>> -> Numpy array`
- Python docstring is made out of C++ doxygen comments
- Cross-language polymorphism

```
class IFitObserver {
    virtual void update(FitSuite *suite);
};

class FitSuite {
    void attach(IFitObserver *observer);
    void runFit() {
        observer->update(this);
    }
};
```

```
class DrawObserver(IFitObserver):
    def __init__(self):
        IFitObserver.__init__(self)

    def update(self, fit_suite):
        pyplot.imshow(fitSu

observer = DrawObserver()

fitSuite = FitSuite()
fitSuite.attach(observer)
```



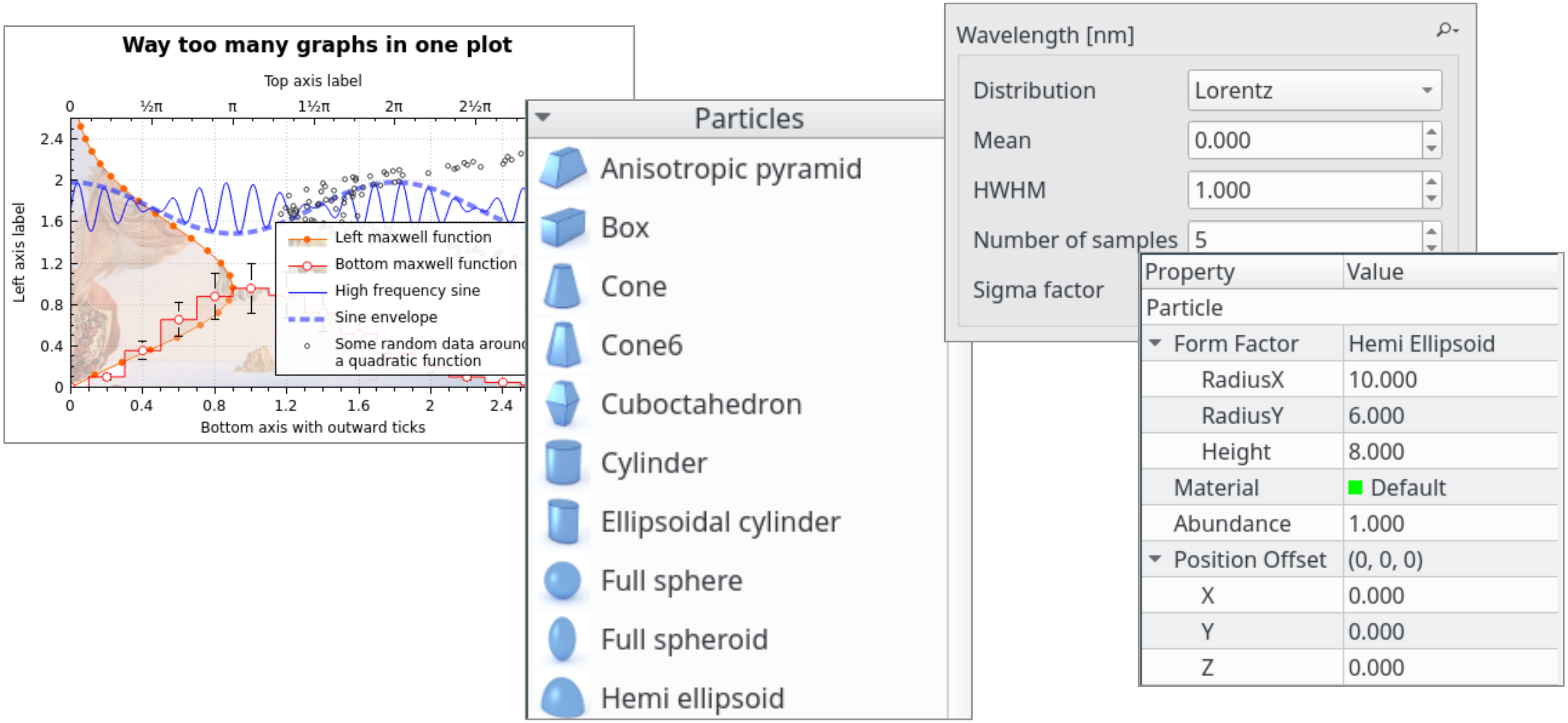
Iteration 68 Minuit2
Chi2 0.1604

*/FormFactorCylinder/radius : 5.003
*/FormFactorCylinder/height : 10.001

- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

GUI main features

- ○ 60k lines of code, Qt5 based, C++
- ○ Additional 3rd party code (included in source tree)
  - QCustomPlot (scientific graphics)
  - Qt-manhattan-style (few styles/widgets borrowed from Qt creator code)
  - Qt-propertybrowser-framework (dynamic property editors generation)



Wavelength [nm]

Distribution: Lorentz

Mean: 0.000

HWHM: 1.000

Number of samples: 5

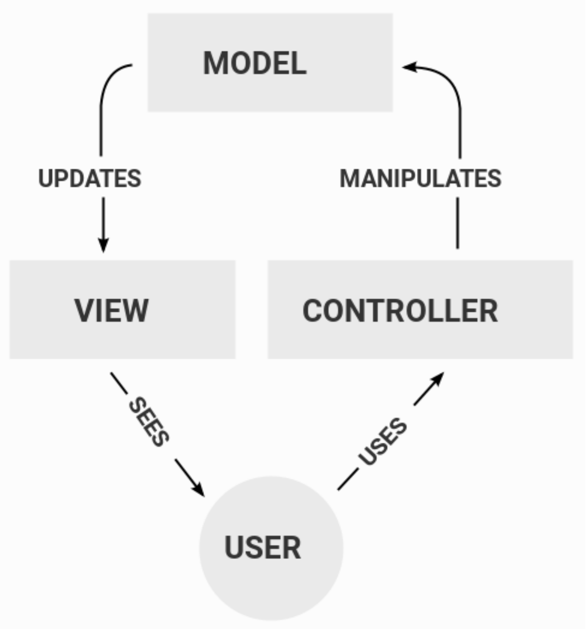
Sigma factor

Property	Value
Particle	
Form Factor	Hemi Ellipsoid
RadiusX	10.000
RadiusY	6.000
Height	8.000
Material	🟩 Default
Abundance	1.000
Position Offset	(0, 0, 0)
X	0.000
Y	0.000
Z	0.000

GUI main features

The Model/View architecture

- The data (model), user interface (view) and interactions (controller) are separated



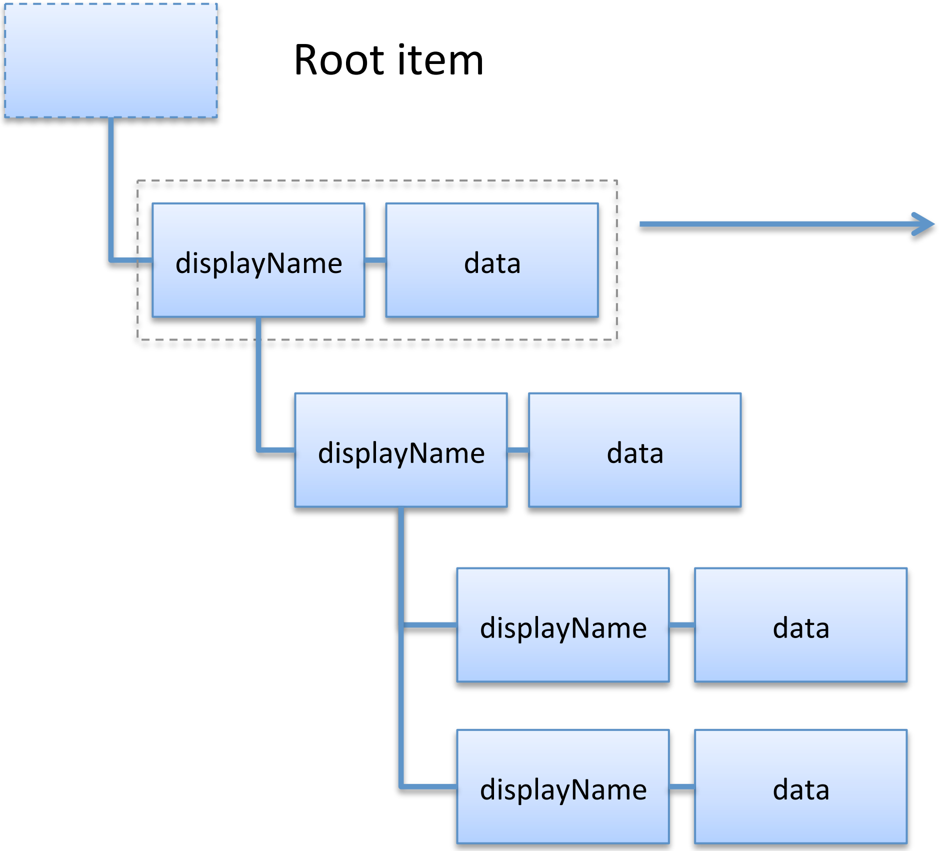
Advantages

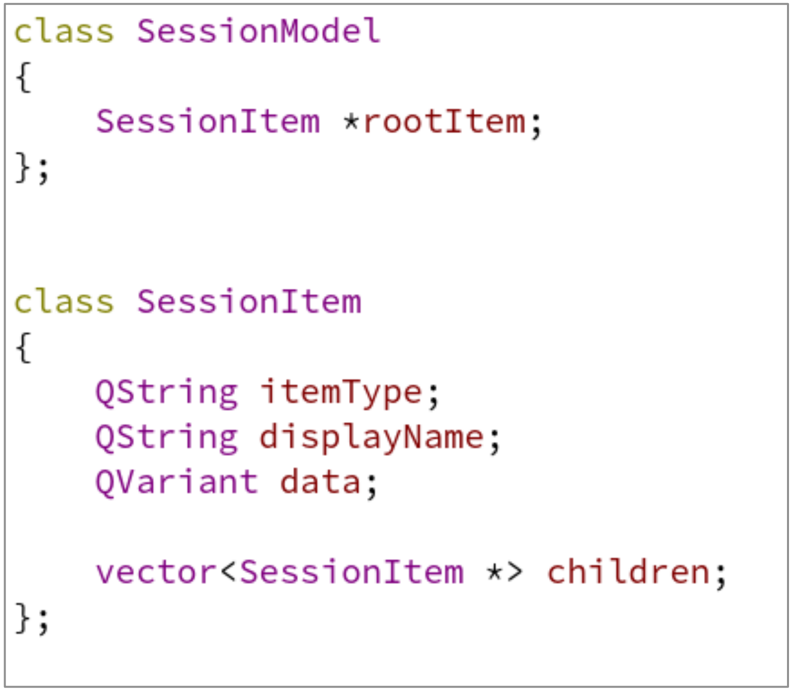
- Same data can be displayed in many views
- Increased flexibility and reuse
- Possibility to unit-test GUI logic outside of GUI context

GUI main features

Presentation Model

- Holds all the data (sample parameters, presentation attributes, widgets status)
- Every row in the model corresponds to `SessionItem`



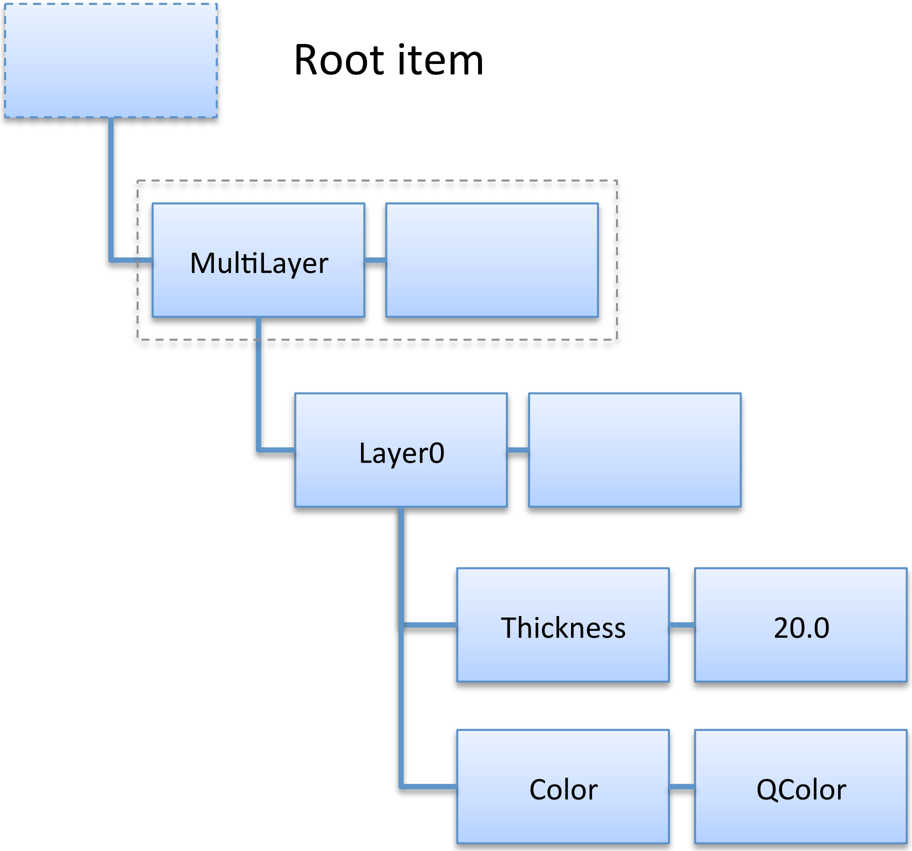


```
class SessionModel
{
    SessionItem *rootItem;
};


class SessionItem
{
    QString itemType;
    QString displayName;
    QVariant data;

    vector<SessionItem *> children;
};
```

GUI main features

Presentation Model

- Holds all the data (sample parameters, presentation attributes, widgets status)
- Every row in the model corresponds to `SessionItem`



Root item

MultiLayer

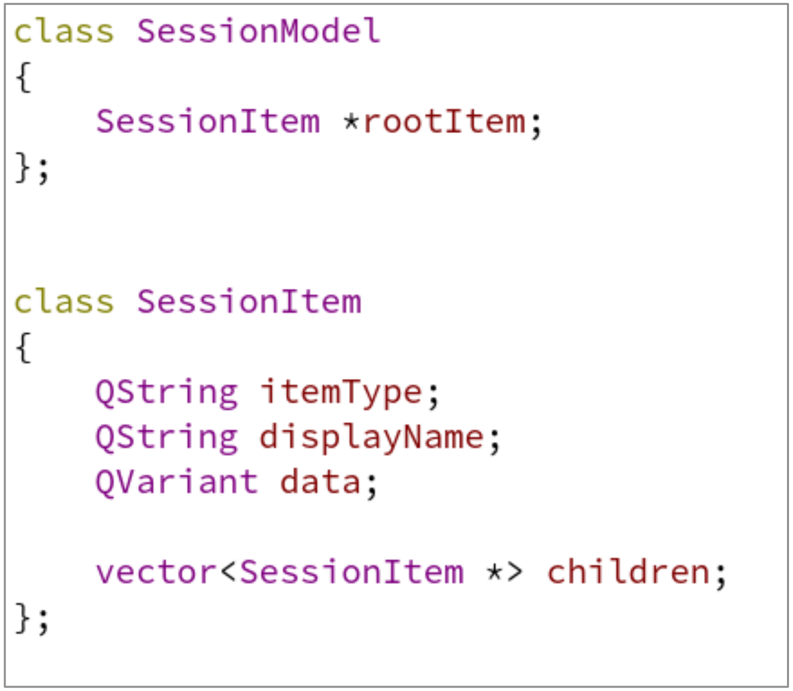
Layer0

Thickness

20.0

Color

QColor

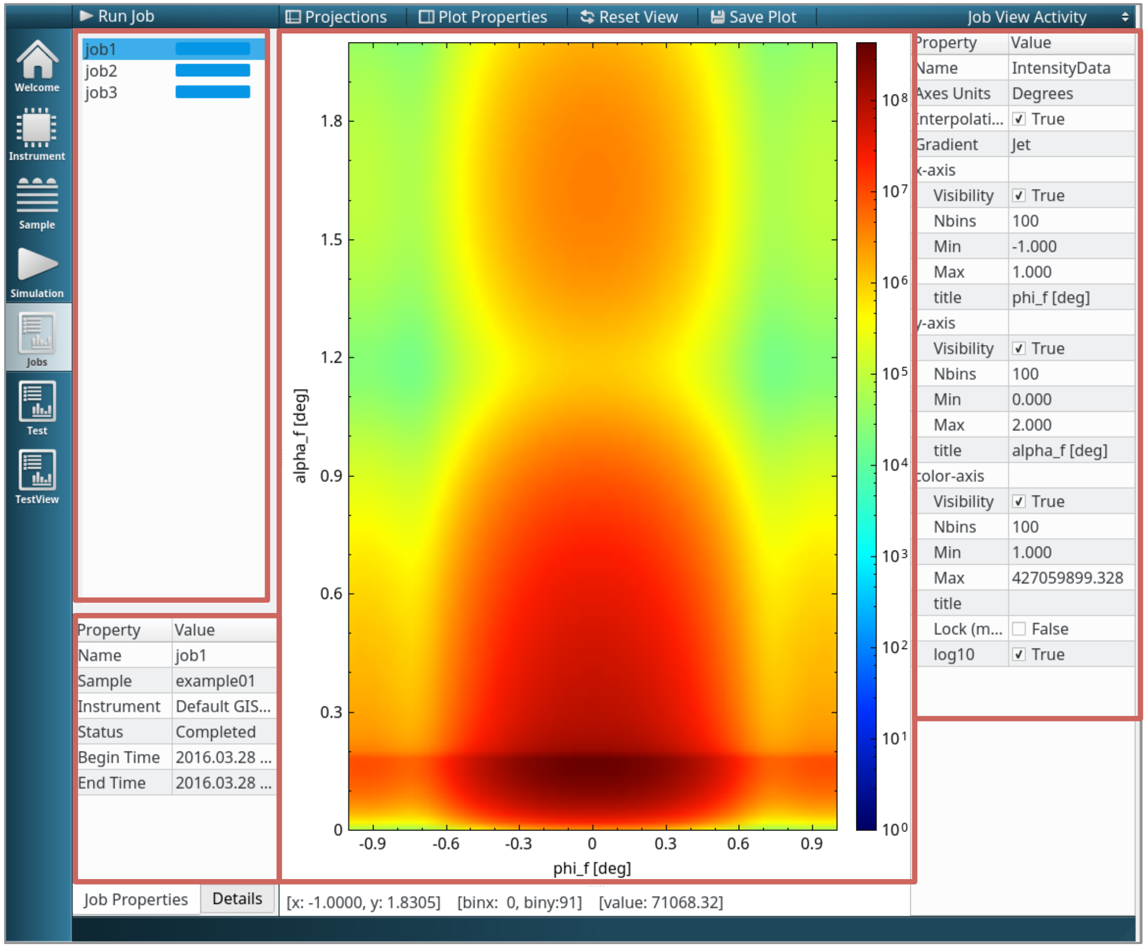


```
class SessionModel
{
    SessionItem *rootItem;
};


class SessionItem
{
    QString itemType;
    QString displayName;
    QVariant data;

    vector<SessionItem *> children;
};
```

GUI main features

Presentation Model and its Views

- Part of presentation model related to job results
- Job views representing different items of job model

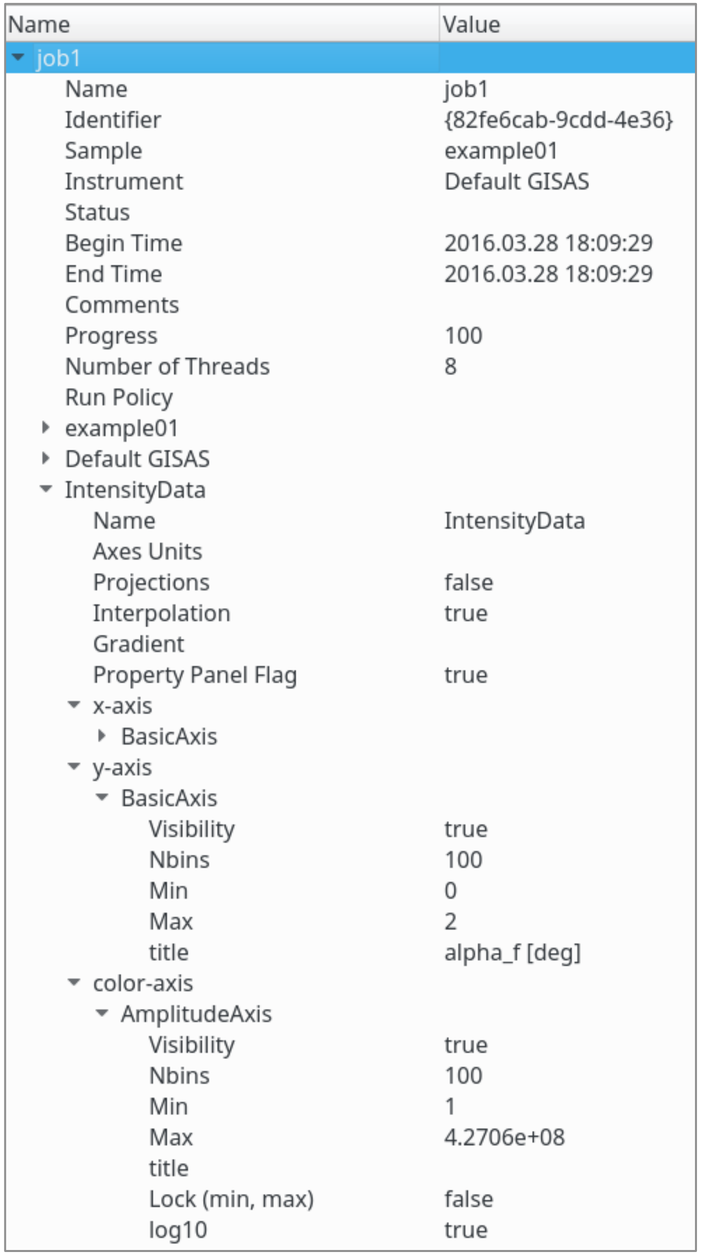


job1
job2
job3

Property	Value
Name	job1
Sample	example01
Instrument	Default GIS...
Status	Completed
Begin Time	2016.03.28 ...
End Time	2016.03.28 ...

Property	Value
Name	IntensityData
Axes Units	Degrees
Interpolati...	☑ True
Gradient	Jet
x-axis	
Visibility	☑ True
Nbins	100
Min	-1.000
Max	1.000
title	phi_f [deg]
y-axis	
Visibility	☑ True
Nbins	100
Min	0.000
Max	2.000
title	alpha_f [deg]
color-axis	
Visibility	☑ True
Nbins	100
Min	1.000
Max	427059899.328
title	
Lock (m...	☐ False
log10	☑ True

[x: -1.0000, y: 1.8305] [binx: 0, biny:91] [value: 71068.32]

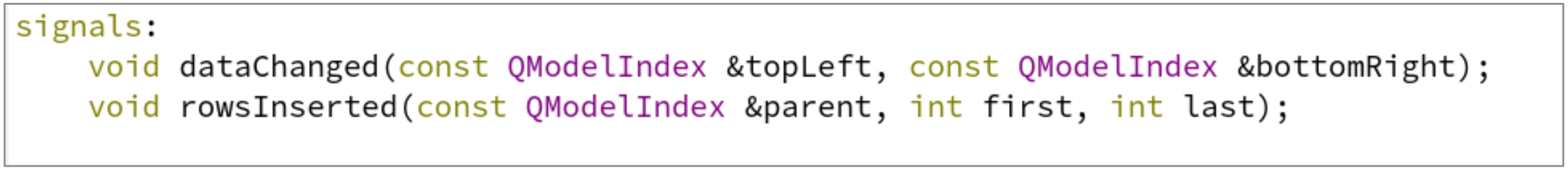


Name	Value
job1	
Name	job1
Identifier	{82fe6cab-9cdd-4e36}
Sample	example01
Instrument	Default GISAS
Status	
Begin Time	2016.03.28 18:09:29
End Time	2016.03.28 18:09:29
Comments	
Progress	100
Number of Threads	8
Run Policy	
example01	
Default GISAS	
IntensityData	
Name	IntensityData
Axes Units	
Projections	false
Interpolation	true
Gradient	
Property Panel Flag	true
x-axis	
BasicAxis	
y-axis	
BasicAxis	
Visibility	true
Nbins	100
Min	0
Max	2
title	alpha_f [deg]
color-axis	
AmplitudeAxis	
Visibility	true
Nbins	100
Min	1
Max	4.2706e+08
title	
Lock (min, max)	false
log10	true

GUI main features

Presentation Model

- ○ Conform to `QAbstractItemModel` interface

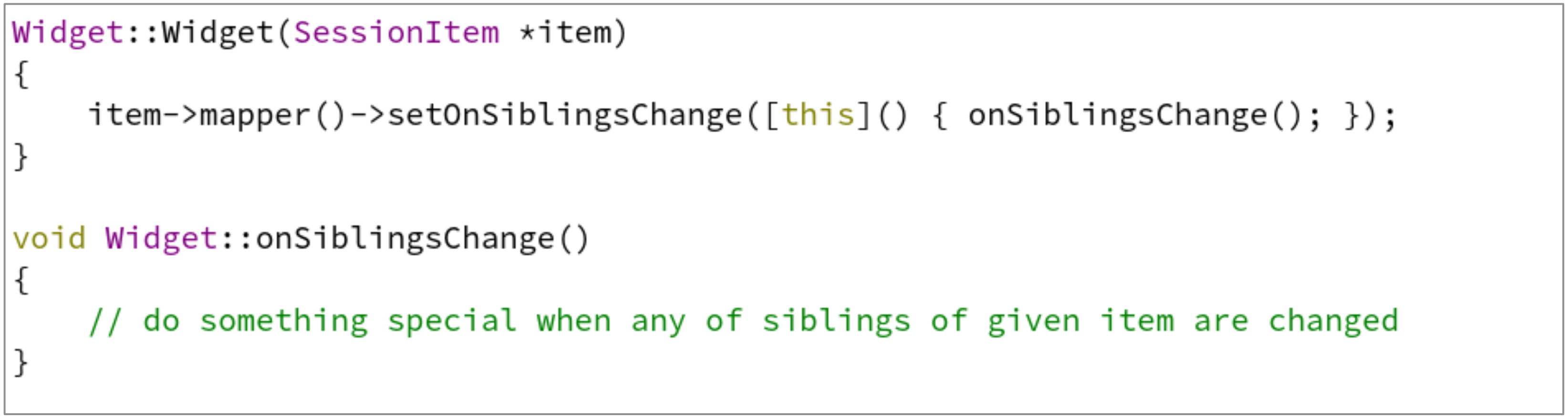


```
signals:
    void dataChanged(const QModelIndex &topLeft, const QModelIndex &bottomRight);
    void rowsInserted(const QModelIndex &parent, int first, int last);
```

- ○ Various proxy models allows to hide certain model parts from the view

- ○ Serialization is done via XML stream
  - Saving the model in file, drag-and-drop, copying/cloning across the model

- ○ Additional machinery allows non-Qt objects to be notified on SessionItem change

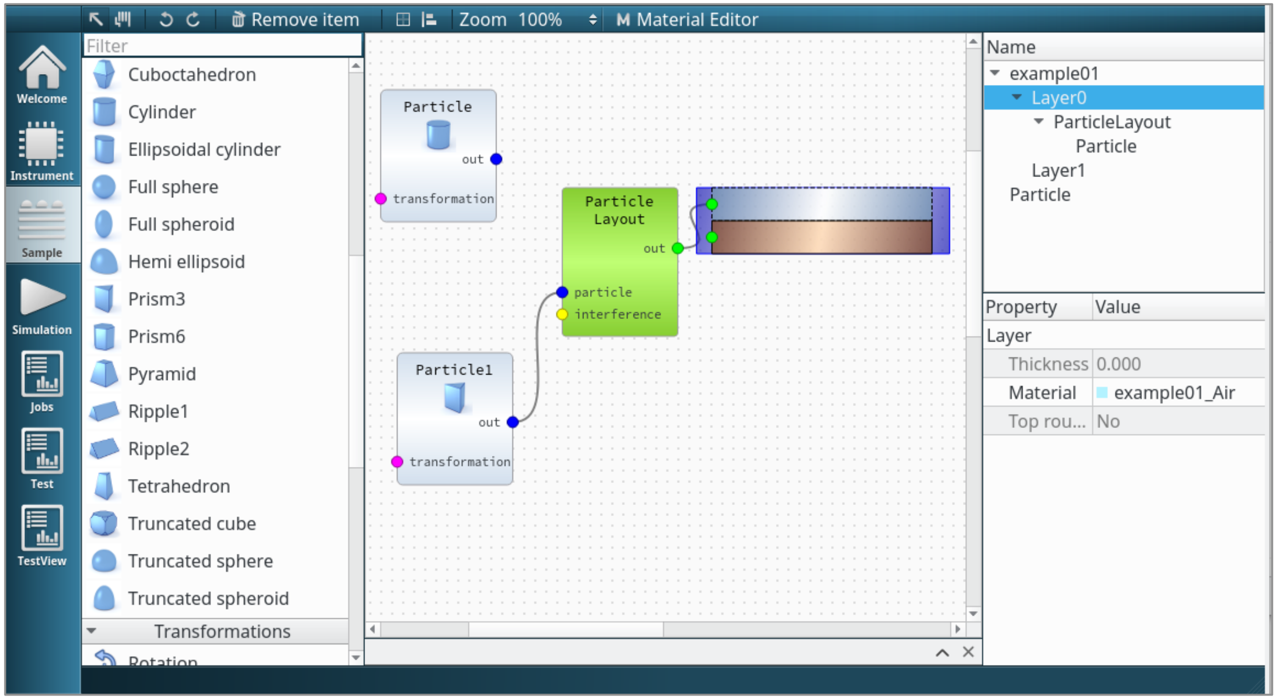


```
Widget::Widget(SessionItem *item)
{
    item->mapper()->setOnSiblingsChange([this]() { onSiblingsChange(); });
}

void Widget::onSiblingsChange()
{
    // do something special when any of siblings of given item are changed
}
```

GUI main features

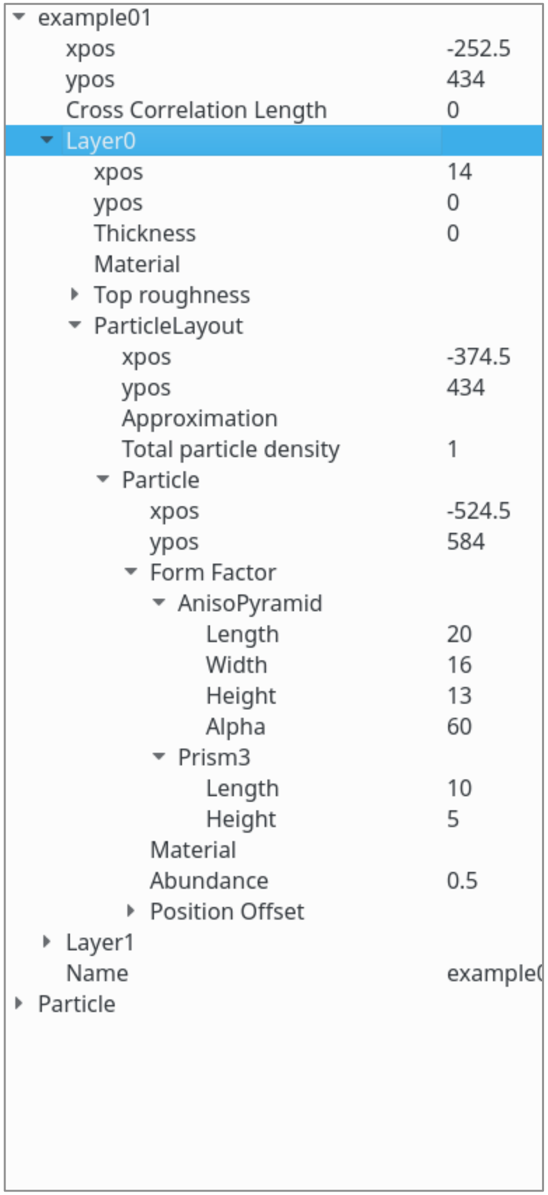
All activities are done through the model

- Drag and Drop action adds an item to the model
  - Graphics scene gets notified and draws new item
- Connection of items through node editor leads to request to change the parent in the model
  - Graphics scene gets notified and draws connection



```
Name
▾ example01
  ▾ Layer0
    ▾ ParticleLayout
        Particle
    Layer1
  Particle
```

Property	Value
Layer	
Thickness	0.000
Material	example01_Air
Top rou...	No



```
▾ example01
    xpos                                 -252.5
    ypos                                 434
    Cross Correlation Length             0
  ▾ Layer0
      xpos                               14
      ypos                               0
      Thickness                          0
      Material
    ▸ Top roughness
    ▾ ParticleLayout
        xpos                             -374.5
        ypos                             434
        Approximation
        Total particle density           1
      ▾ Particle
          xpos                           -524.5
          ypos                           584
        ▾ Form Factor
          ▾ AnisoPyramid
              Length                     20
              Width                      16
              Height                     13
              Alpha                      60
          ▾ Prism3
              Length                     10
              Height                     5
          Material
          Abundance                      0.5
        ▸ Position Offset
  ▸ Layer1
    Name                                 example(
▸ Particle
```

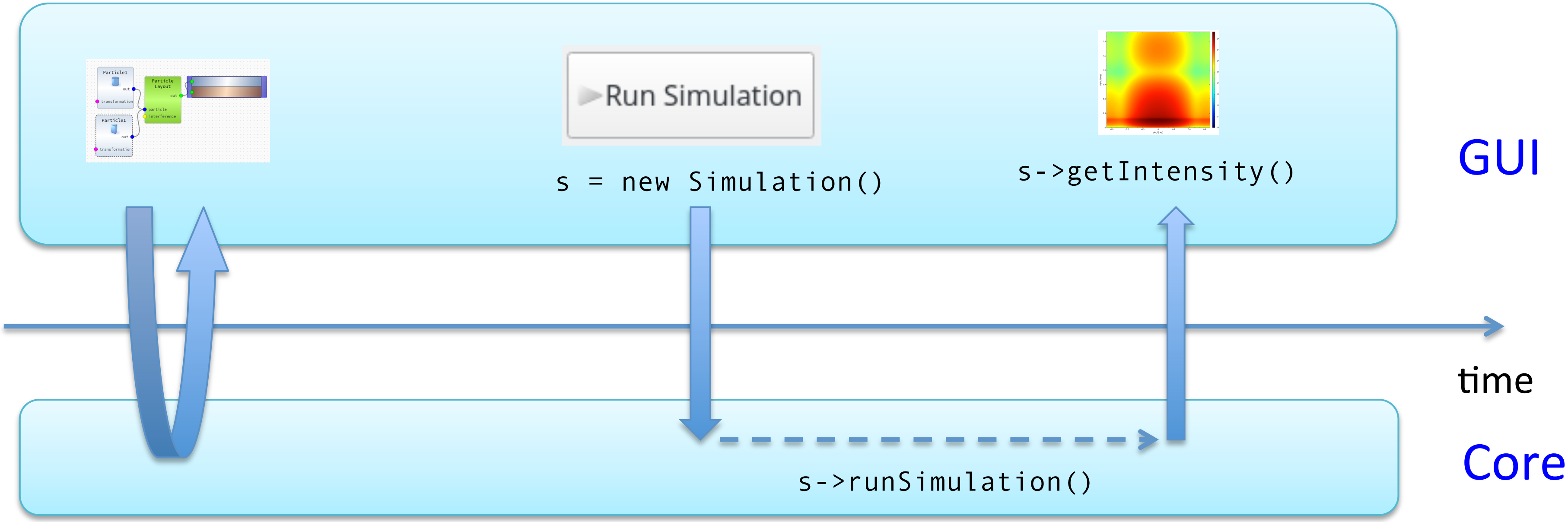
GUI main features

GUI / Core relationship

Converts domain objects (standard samples, library materials etc) into their GUI counterparts

Generates core domain simulation object, runs it in non-GUI thread

Knows how to retrieve simulation results



`s = new Simulation()`

`s->getIntensity()`

`s->runSimulation()`

GUI

time

Core

Core is Qt-independent and fully unaware of GUI existence

- Introduction
- Software architecture
- Demonstration
- Under the hood
- Closing remarks

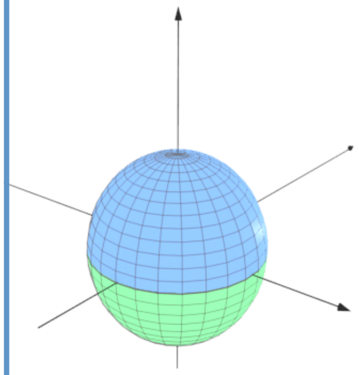
Testing

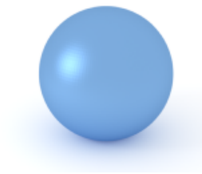
Unit tests

- ○ 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





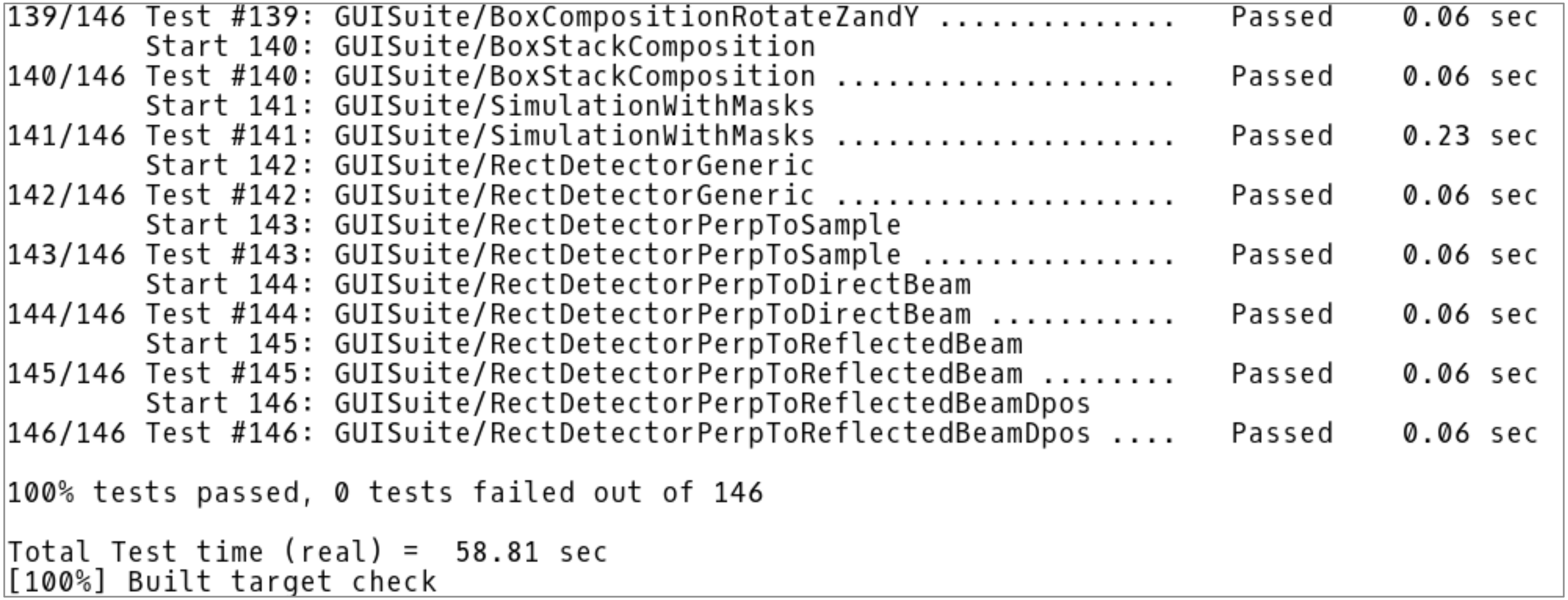
- ○ Create particle composition from two hemi spheres
- ○ Assign same material to them
- ○ Compare with normal full sphere, same material, same radius
- ○ Scattering intensities should be identical

Testing

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.

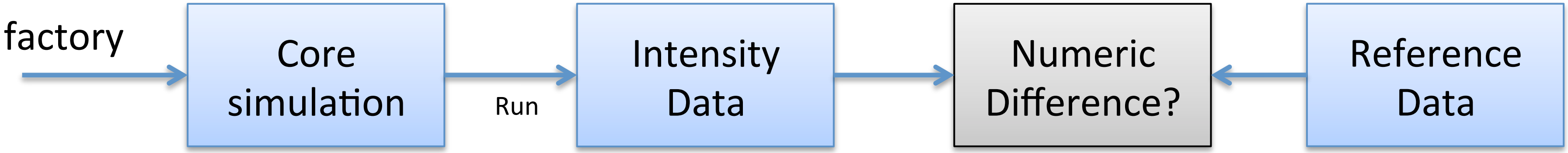
`make check` launches test simulations for all 3 domains



```
139/146 Test #139: GUISuite/BoxCompositionRotateZandY ..............   Passed    0.06 sec
        Start 140: GUISuite/BoxStackComposition
140/146 Test #140: GUISuite/BoxStackComposition ....................   Passed    0.06 sec
        Start 141: GUISuite/SimulationWithMasks
141/146 Test #141: GUISuite/SimulationWithMasks ....................   Passed    0.23 sec
        Start 142: GUISuite/RectDetectorGeneric
142/146 Test #142: GUISuite/RectDetectorGeneric ....................   Passed    0.06 sec
        Start 143: GUISuite/RectDetectorPerpToSample
143/146 Test #143: GUISuite/RectDetectorPerpToSample ...............   Passed    0.06 sec
        Start 144: GUISuite/RectDetectorPerpToDirectBeam
144/146 Test #144: GUISuite/RectDetectorPerpToDirectBeam ...........   Passed    0.06 sec
        Start 145: GUISuite/RectDetectorPerpToReflectedBeam
145/146 Test #145: GUISuite/RectDetectorPerpToReflectedBeam ........   Passed    0.06 sec
        Start 146: GUISuite/RectDetectorPerpToReflectedBeamDpos
146/146 Test #146: GUISuite/RectDetectorPerpToReflectedBeamDpos ....   Passed    0.06 sec

100% tests passed, 0 tests failed out of 146

Total Test time (real) =  58.81 sec
[100%] Built target check
```

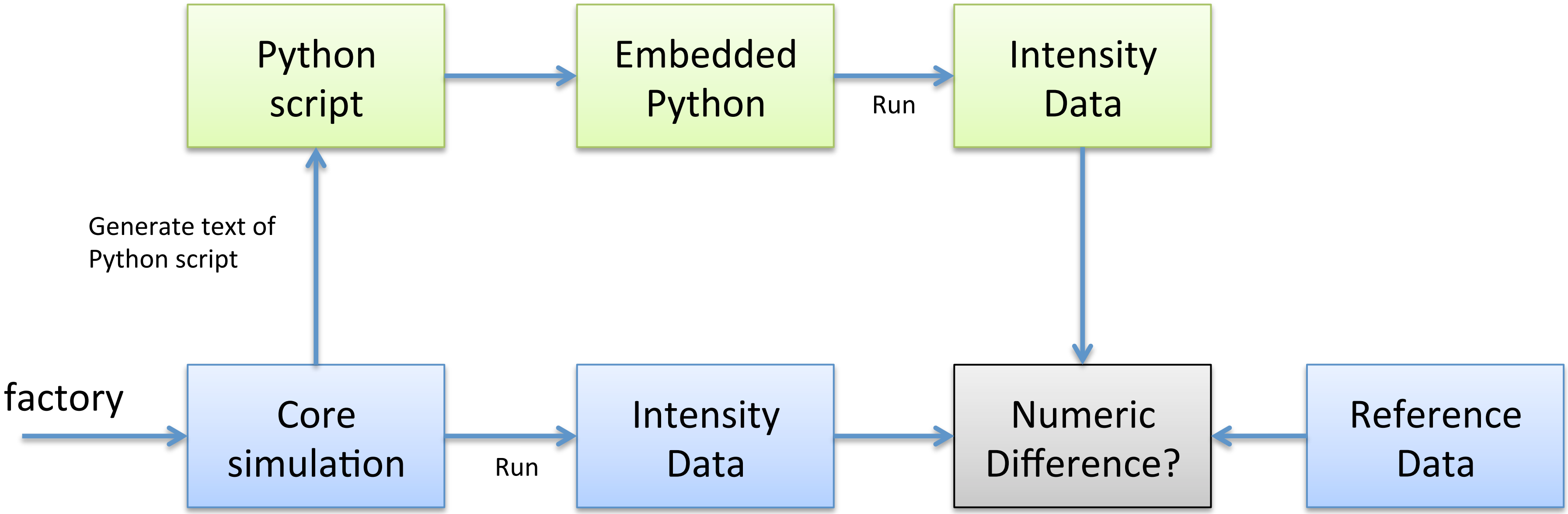
Testing

Functional tests for Core/GUI/Python domains



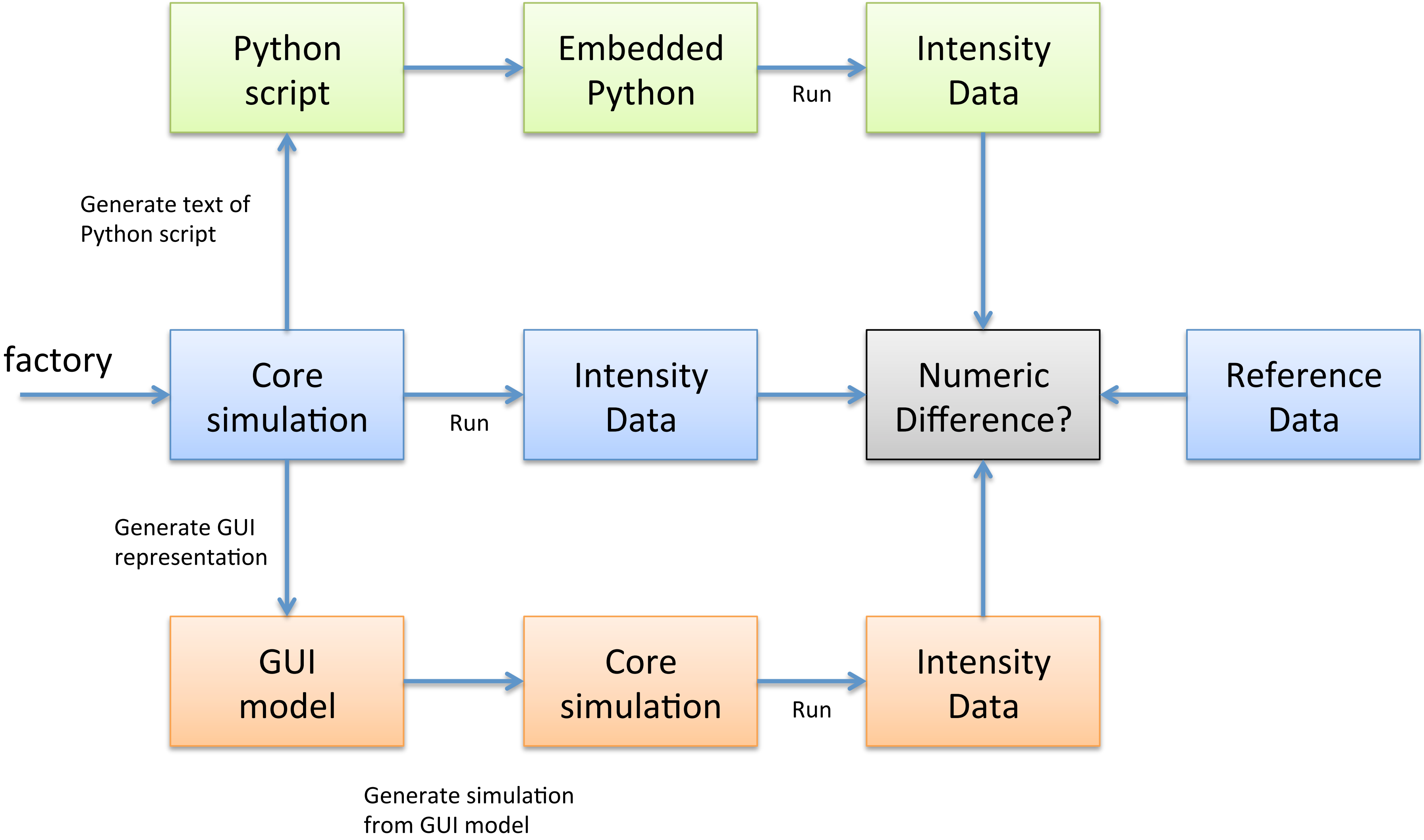
Testing

Functional tests for Core/GUI/Python domains



Testing

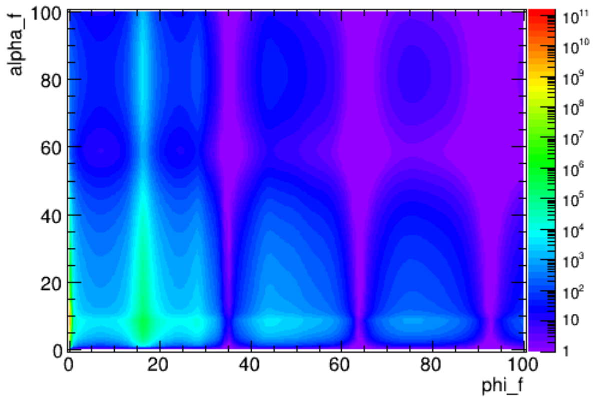
Functional tests for Core/GUI/Python domains



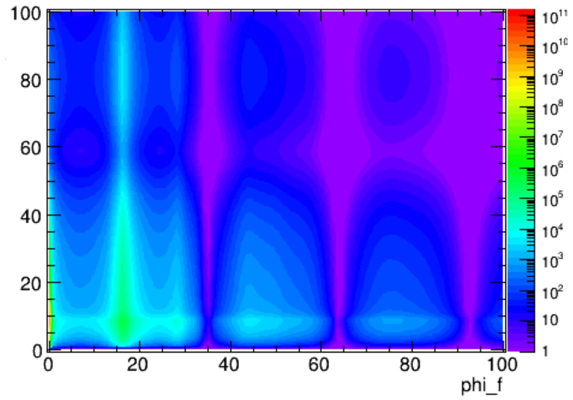
Validation

Validation against existing software

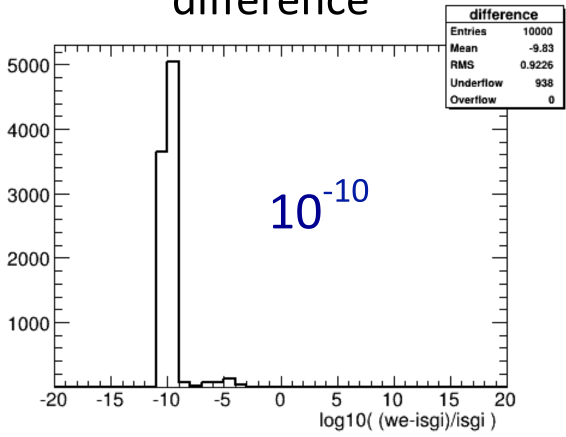
BornAgain



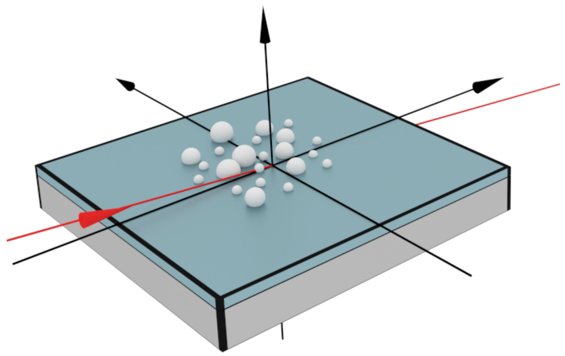
IsGISAXS

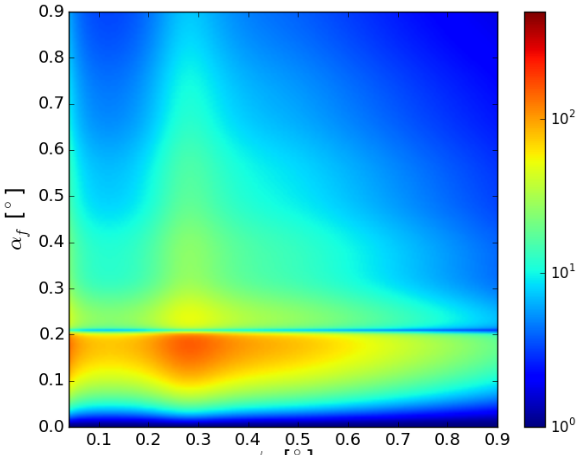


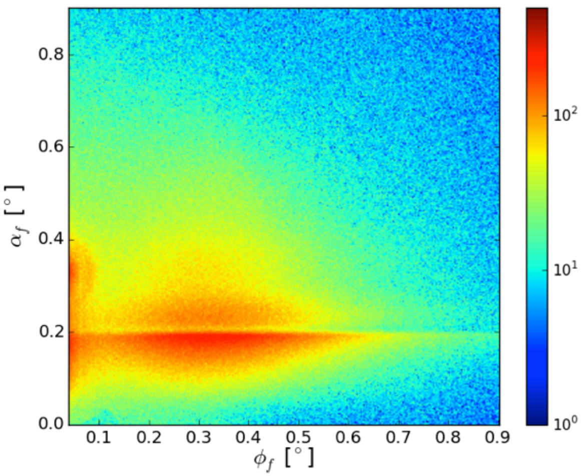
difference



Validation against experimental data







Closing remarks

Horizon 2020 Initiative

- BornAgain as a community project for GISAS and Reflectometry
- Fitting of GISAS, Off-Specular and Specular data in a single framework

Further software development tasks

- Fitting in GUI (prototype in next release)
- Real sample representation using Qt3D
- Switch to Qt installer framework to create MacOS and Windows installers

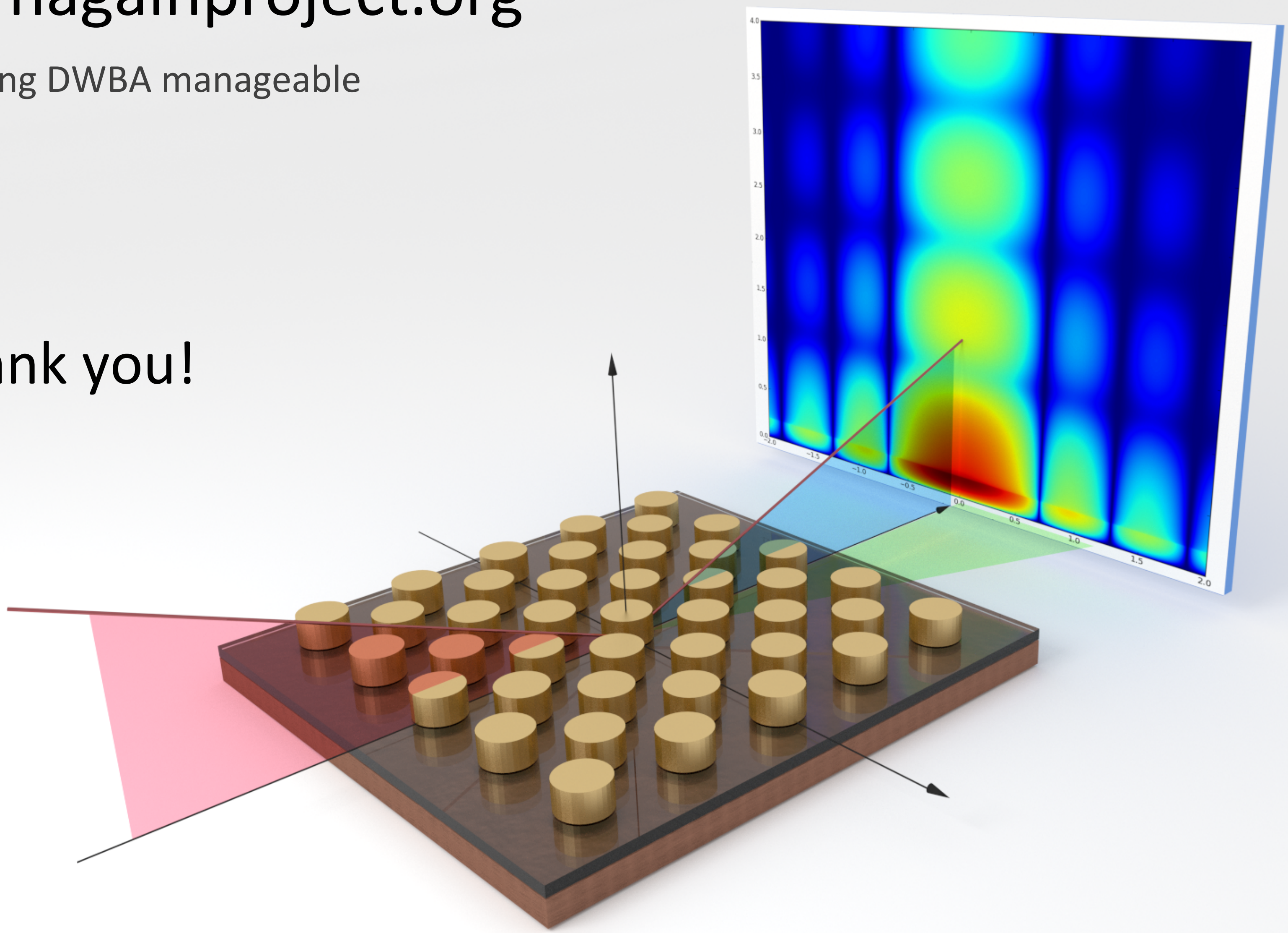
Further kernel development

- Implement specular intensity
- Magnetic roughness and magnetic domains

bornagainproject.org

making DWBA manageable

Thank you!

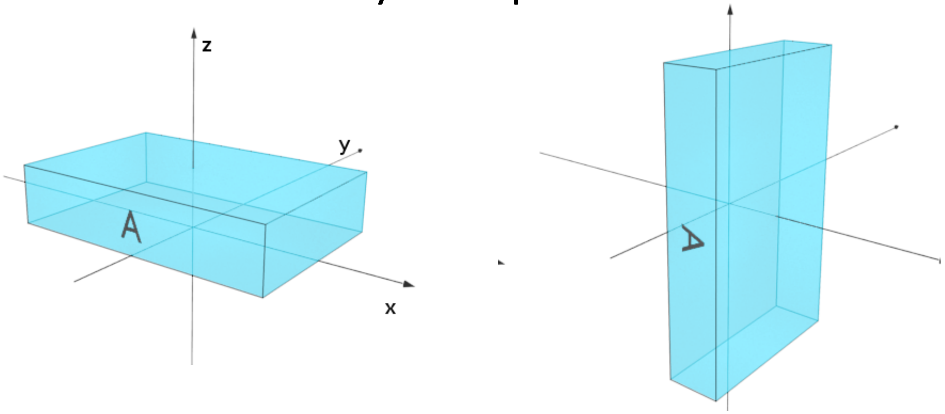


BACKUP

Self validation

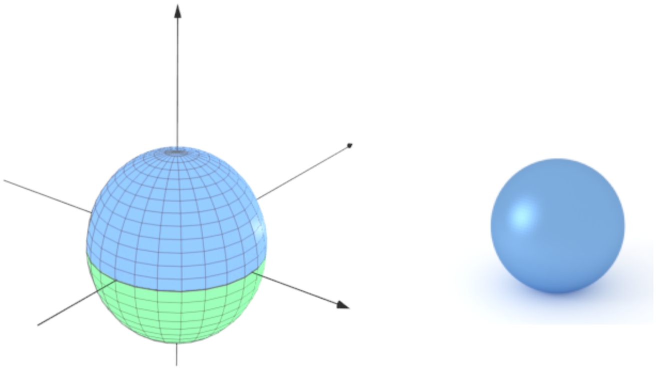
Part of new BornAgain's functionality can be validated via BornAgain itself

- Rotation machinery example



- Create box (30,20,6)
- RotateY by 90 degrees
- Compare with non-rotated box (6,20,30)
- Scattering intensities should be identical

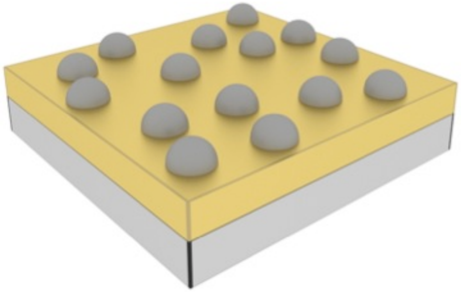
- Particle compositions example

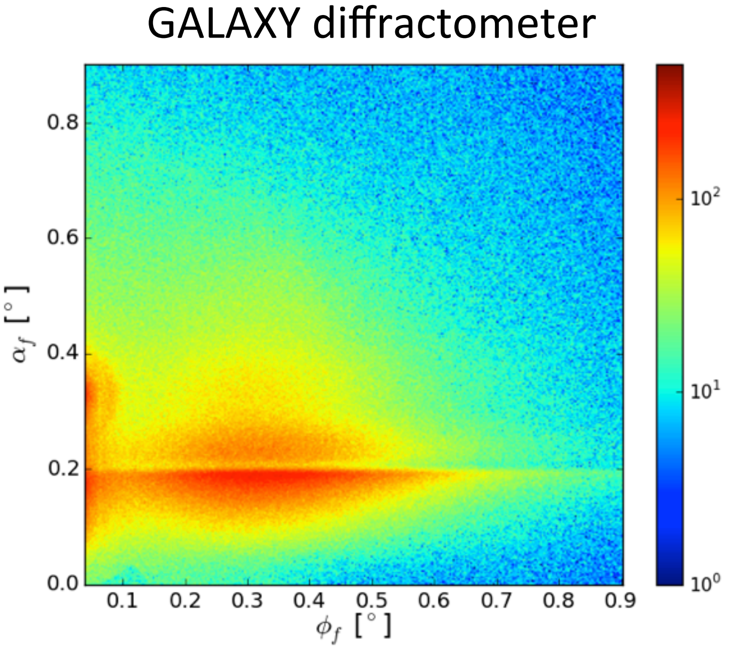


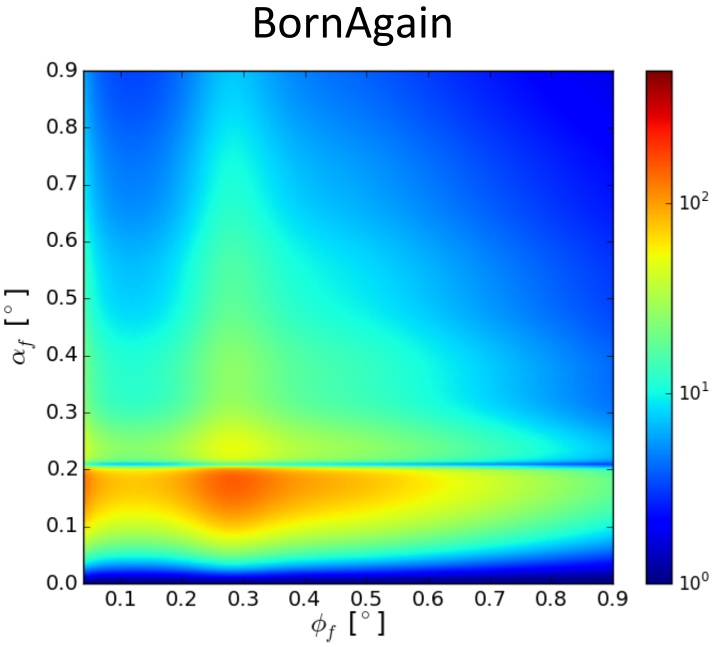
- Create particle composition from two hemi spheres
- Assign same material to them
- Compare with normal full sphere, same material, same radius
- Scattering intensities should be identical

Validation against experimental data

Fitting of 3 layers system with Ag nanoparticles with broad size distribution



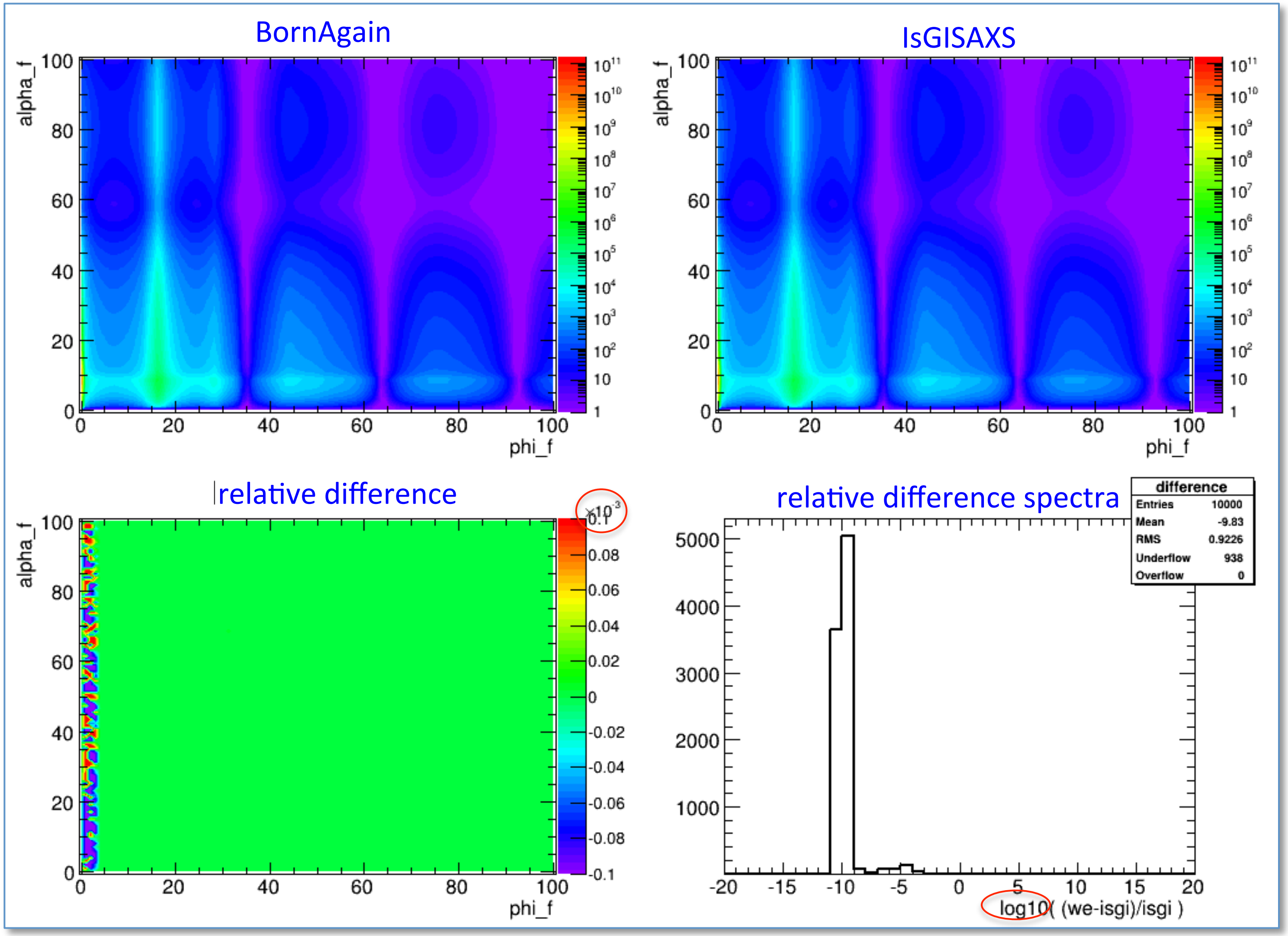




http://apps.jcns.fz-juelich.de/doku/sc/_media/dpg-berlin-talk1.pptx

Validation against IsGISAXS

BornAgain results mostly coincide with IsGisaxs on numerical level

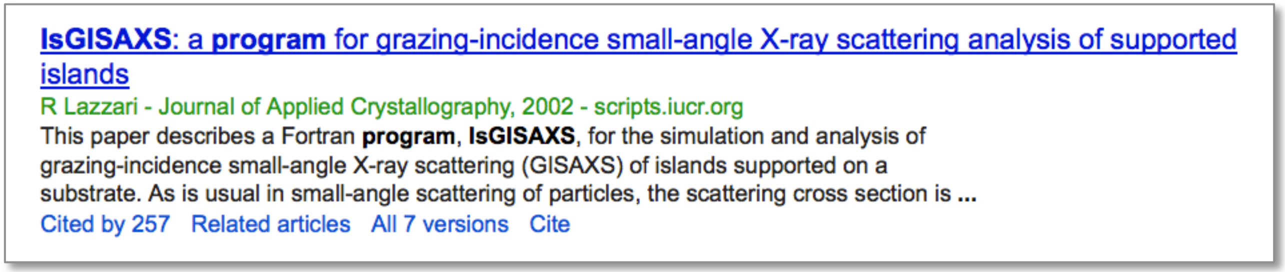


Existing software

Package	Application	Platform	License
IsGISAXS	Nanostructures on surfaces	Windows, Unix	GNU Public
FitGISAXS	Buried nanostructures	IgorPRO	GNU Public + IgorPRO
HipGISAXS	Buried nanostructures	Unix, HPC Computing	Berkeley, non-commercial

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

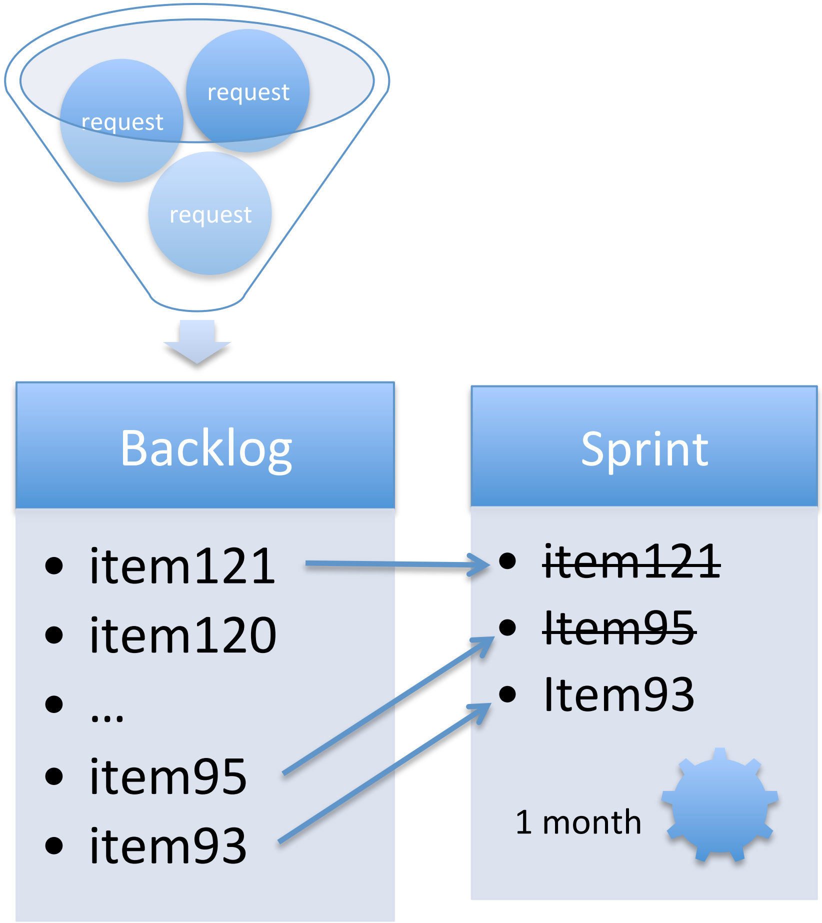
This paper describes a Fortran **program**, **IsGISAXS**, for the simulation and analysis of grazing-incidence small-angle X-ray scattering (GISAXS) of islands supported on a substrate. As is usual in small-angle scattering of particles, the scattering cross section is **...**

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



request

request

request

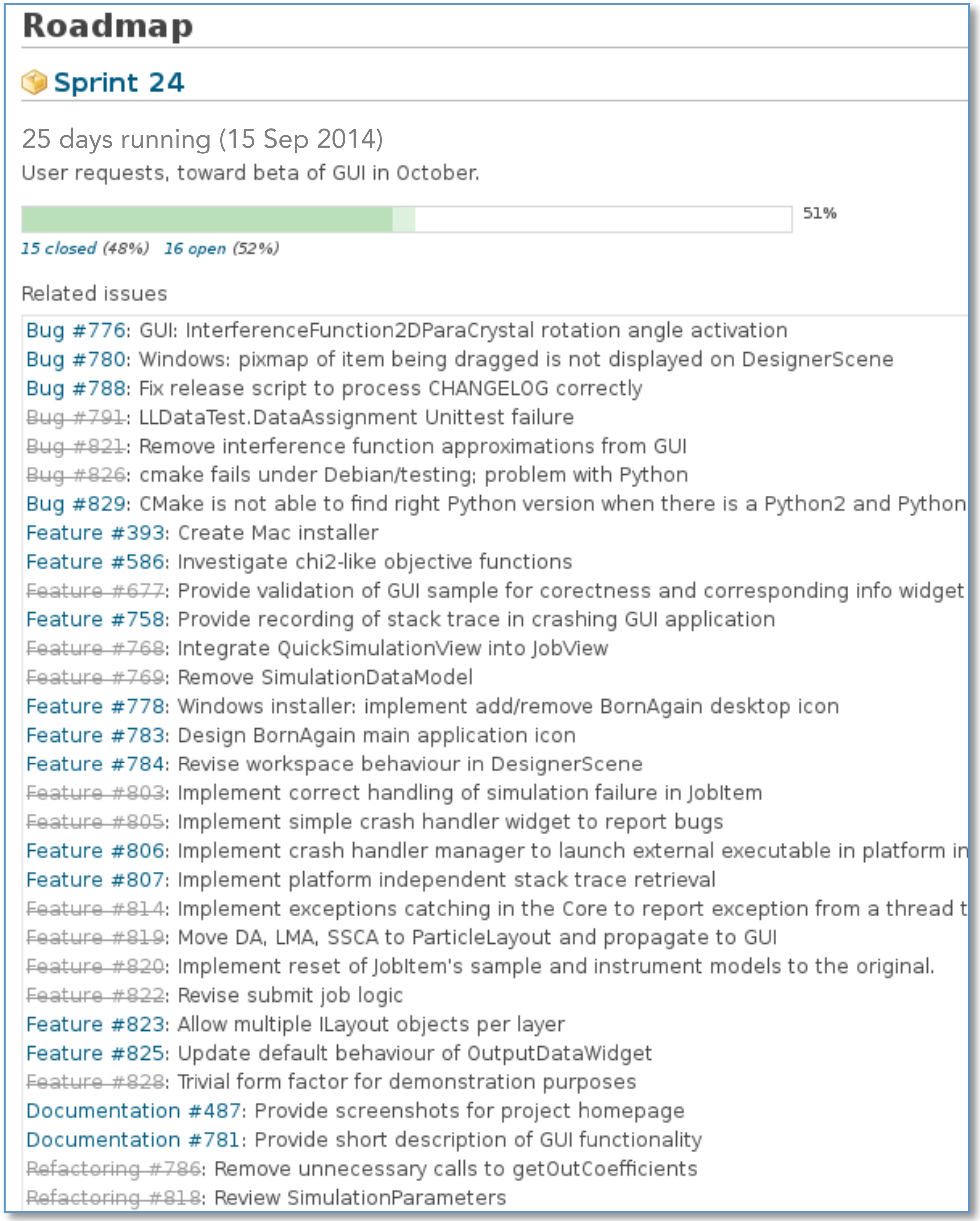
Backlog

- item121
- item120
- …
- item95
- item93

Sprint

- ~~item121~~
- ~~Item95~~
- Item93

1 month



Roadmap

Sprint 24

25 days running (15 Sep 2014)

User requests, toward beta of GUI in October.

51%

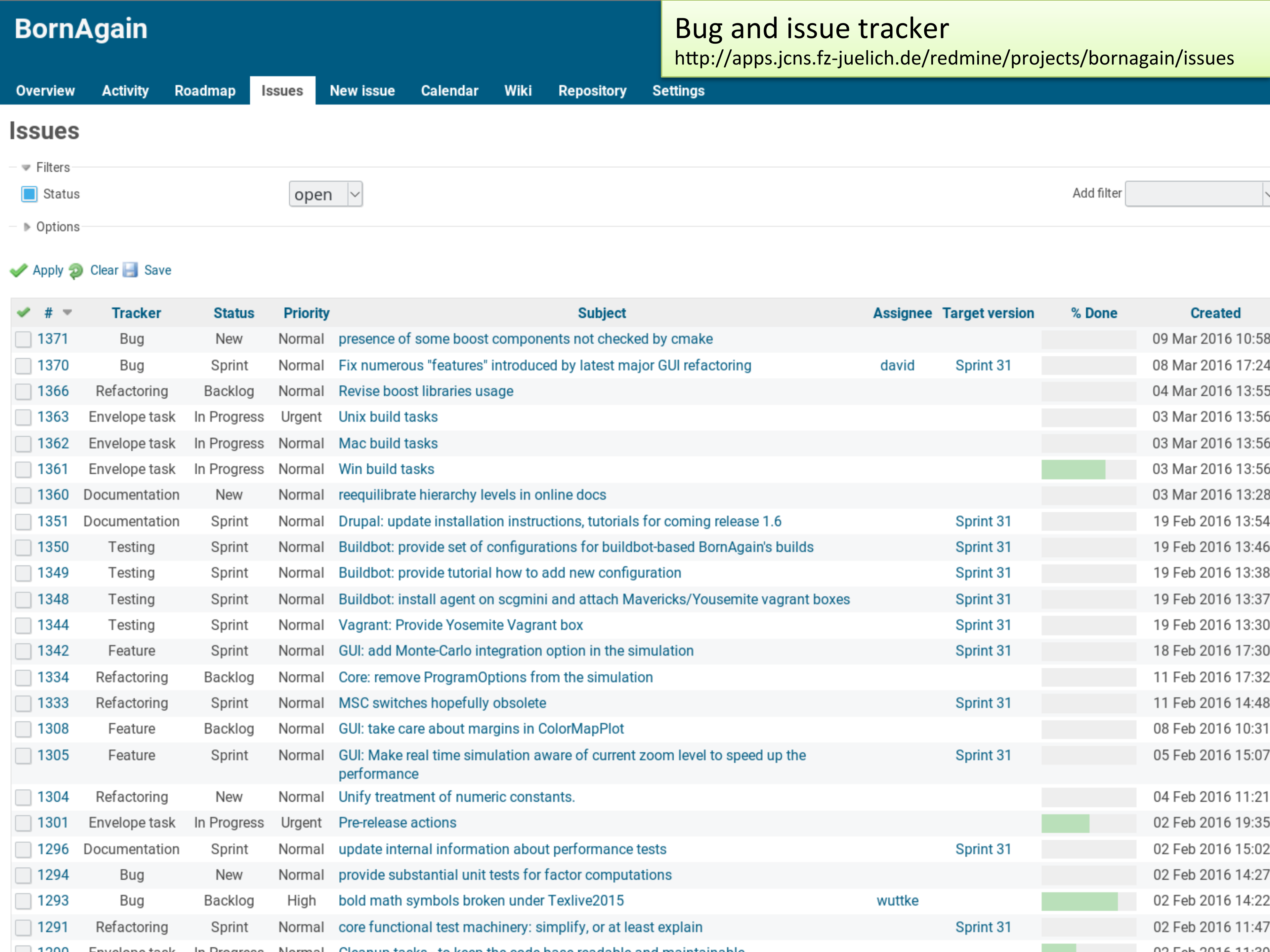
15 closed (48%) 16 open (52%)

Related issues

Bug #776: GUI: InterferenceFunction2DParaCrystal rotation angle activation
Bug #780: Windows: pixmap of item being dragged is not displayed on DesignerScene
Bug #788: Fix release script to process CHANGELOG correctly
~~Bug #791~~: LLDataTest.DataAssignment Unittest failure
~~Bug #821~~: 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 corectness and corresponding info widget
Feature #758: Provide recording of stack trace in crashing GUI application
~~Feature #768~~: Integrate QuickSimulationView into JobView
~~Feature #769~~: Remove SimulationDataModel
Feature #778: Windows installer: implement add/remove BornAgain desktop icon
Feature #783: Design BornAgain main application icon
Feature #784: Revise workspace behaviour in DesignerScene
~~Feature #803~~: 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 in
Feature #807: Implement platform independent stack trace retrieval
~~Feature #814~~: Implement exceptions catching in the Core to report exception from a thread t
~~Feature #819~~: Move DA, LMA, SSCA to ParticleLayout and propagate to GUI
~~Feature #820~~: Implement reset of JobItem's sample and instrument models to the original.
~~Feature #822~~: Revise submit job logic
Feature #823: Allow multiple ILayout objects per layer
Feature #825: Update default behaviour of OutputDataWidget
~~Feature #828~~: 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 #818~~: Review SimulationParameters

Bug and issue tracker

http://apps.jcns.fz-juelich.de/redmine/projects/bornagain/issues



Issues

Filters

☑ Status open

Add filter

Options

Apply Clear Save

✓	#	Tracker	Status	Priority	Subject	Assignee	Target version	% Done	Created
□	1371	Bug	New	Normal	presence of some boost components not checked by cmake				09 Mar 2016 10:58
□	1370	Bug	Sprint	Normal	Fix numerous "features" introduced by latest major GUI refactoring	david	Sprint 31		08 Mar 2016 17:24
□	1366	Refactoring	Backlog	Normal	Revise boost libraries usage				04 Mar 2016 13:55
□	1363	Envelope task	In Progress	Urgent	Unix build tasks				03 Mar 2016 13:56
□	1362	Envelope task	In Progress	Normal	Mac build tasks				03 Mar 2016 13:56
□	1361	Envelope task	In Progress	Normal	Win build tasks				03 Mar 2016 13:56
□	1360	Documentation	New	Normal	reequilibrate hierarchy levels in online docs				03 Mar 2016 13:28
□	1351	Documentation	Sprint	Normal	Drupal: update installation instructions, tutorials for coming release 1.6		Sprint 31		19 Feb 2016 13:54
□	1350	Testing	Sprint	Normal	Buildbot: provide set of configurations for buildbot-based BornAgain's builds		Sprint 31		19 Feb 2016 13:46
□	1349	Testing	Sprint	Normal	Buildbot: provide tutorial how to add new configuration		Sprint 31		19 Feb 2016 13:38
□	1348	Testing	Sprint	Normal	Buildbot: install agent on scgmini and attach Mavericks/Yousemite vagrant boxes		Sprint 31		19 Feb 2016 13:37
□	1344	Testing	Sprint	Normal	Vagrant: Provide Yosemite Vagrant box		Sprint 31		19 Feb 2016 13:30
□	1342	Feature	Sprint	Normal	GUI: add Monte-Carlo integration option in the simulation		Sprint 31		18 Feb 2016 17:30
□	1334	Refactoring	Backlog	Normal	Core: remove ProgramOptions from the simulation				11 Feb 2016 17:32
□	1333	Refactoring	Sprint	Normal	MSC switches hopefully obsolete		Sprint 31		11 Feb 2016 14:48
□	1308	Feature	Backlog	Normal	GUI: take care about margins in ColorMapPlot				08 Feb 2016 10:31
□	1305	Feature	Sprint	Normal	GUI: Make real time simulation aware of current zoom level to speed up the performance		Sprint 31		05 Feb 2016 15:07
□	1304	Refactoring	New	Normal	Unify treatment of numeric constants.				04 Feb 2016 11:21
□	1301	Envelope task	In Progress	Urgent	Pre-release actions				02 Feb 2016 19:35
□	1296	Documentation	Sprint	Normal	update internal information about performance tests		Sprint 31		02 Feb 2016 15:02
□	1294	Bug	New	Normal	provide substantial unit tests for factor computations				02 Feb 2016 14:27
□	1293	Bug	Backlog	High	bold math symbols broken under Texlive2015	wuttke			02 Feb 2016 14:22
□	1291	Refactoring	Sprint	Normal	core functional test machinery: simplify, or at least explain		Sprint 31		02 Feb 2016 11:47
□	1290	Envelope task	In Progress	Normal	Cleanup tasks - to keep the code base readable and maintainable				02 Feb 2016 11:30