

OpenTURNS Developer Training: the platform overview

Trainer : Régis LEBRUN
Airbus
regis.lebrun@airbus.com

Developers training



Platform overview

- 1 The story
- 2 The platform
- 3 The development infrastructure
- 4 Conclusion

Introduction

Objectives

The objectives of this course is to give a broad overview of the OpenTURNS project and the resulting platform. We will cover the following points:

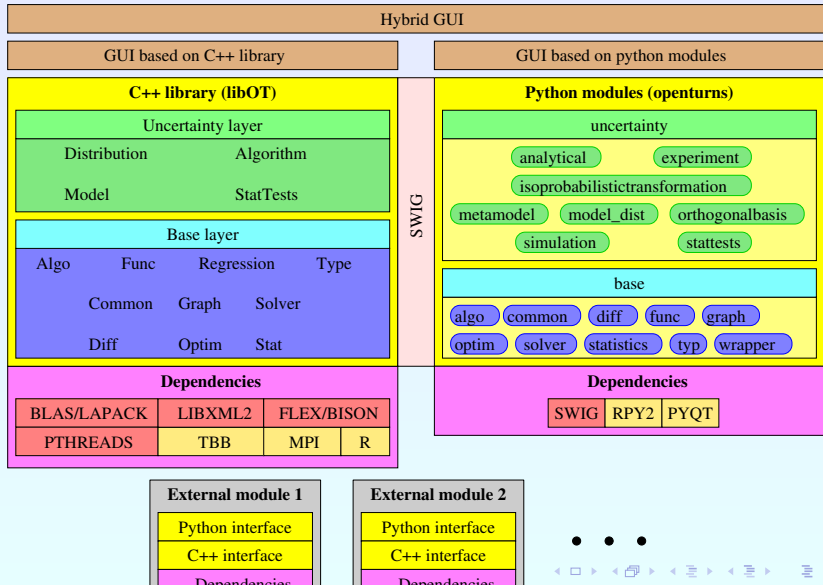
- The history of the project,
- The global organization of the platform,
- The multi-layer view of the library,
- The several usages of the platform.

History

2005-2019: 14 years of partnership

- 2005 Conception
- 2007 First release the 10th of May. Python bindings.
- 2009 Multithreaded wrappers, polynomial chaos expansion.
- 2010 First windows port, parallelization.
- 2011 Sparse chaos implementation.
- 2012 v1.0, Stochastic processes
- 2013 Bayesian updating, matplotlib viewer
- 2014 Kriging, native windows support
- 2015 Vectorial kriging, HMat support
- 2016 Karhunen-Loeve process decomposition, NLopt bindings
- 2017 Canonical format low-rank tensor approximation, field functions
- 2018 Domains arithmetic, asymptotic Sobol' estimators, new simulation algorithms

The platform at a glance, developer's view



The platform at a glance, developer's view

The big parts

- The **core** of the OpenTURNS platform is a **C++ library**, made of about 500 classes of various size. The library has a multi-layered architecture that is materialized by both the namespace hierarchy and the source tree.
- The **main user interface** is the **python module**, automatically generated from the C++ library using the wrapping software SWIG. It allows for a usage of OptnTURNS through python scripts of any level of complexity.
- The library relies on relatively **few dependencies** and most of them are optional.
- A service of **modules** is provided in order to extend the capabilities of the platform from the outside.
- Several **GUIs** have already been built on top of the C++ library or the Python module.

The C++ library

A multilayered library

The two main layers in the C++ library are the **Base** layer and the **Uncertainty** layer.

- **Base** layer: it contains all the classes not related to the probabilistic concepts. It covers the elementary data types (vectors as NumericalPoint, samples as NumericalSamples), the concept of models (NumericalMathFunction), the linear algebra (Matrix, Tensor) and the general interest classes (memory management, resource management);
- **Uncertainty** layer: it contains all the classes that deal with probabilistic concepts. It covers the probabilistic modelling (Distribution, RandomVector), the stochastic algorithms (MonteCarlo, FORM), the statistical estimation (DistributionFactory), the statistical testing (FittingTest)

A class in the Uncertainty layer can use any class in the Base or the Uncertainty layer.
A class in the Base layer can ONLY USE classes in the Base layer.

The C++ library

A monolythic library?

The C++ library is provided as a unique object file (libOT.so) created using the libtool technology. As such, it is a monolythic library (of about 8Mo stripped), but internally it is made of numerous sub-libraries, one for each folder in the source tree. A future objective is to modularize this library in order to speed-up both the compilation time and the loading time.

A parallel library

Some of the most time-consuming algorithms have been parallelized using the Thread Building Block technology (INTEL), a C++ library that allows for a parallelization of C++ code in a shared memory model. One of the objectives of OpenTURNS is the ability to execute external simulation softwares on large simulation models for a large amount of independent data sets. As such, OpenTURNS provides basic functionalities to distribute the executions of these simulations on a multiprocessors (cores) infrastructure using the low-level pthread technology or the TBBs.

The python module

Interfacing python and C++ libraries: SWIG

In order to provide a convenient interface to the user, the C++ library can be manipulated as a set of Python modules (18) that are organized through a hierarchy, on top of which stands the openturns module. The binding of the library is done almost automatically by SWIG (Simplified Wrapper Interface Generator) through a set of SWIG interface files.

An additional significant work has been made in order to ease the interaction between the OpenTURNS objects and the native Python objects.

A unique feature of the Python interface is the ability to wrap a Python function into an OpenTURNS concept of mathematical function, namely the NumericalMathFunction, using a specific class: the OpenTURNSPythonFunction. Using this mechanism, it is possible to address virtually all the scenarios of coupling with an external simulation software.

The target OSeS

A linux platform that works on windows

The historic platform of OpenTURNS is linux. The first stage of developments have been made on 32 bits Intel Linux platforms (debian, mandriva), then the (minor) adaptations have been made in order to run on 64 bits Intel platforms as well. Currently, the platform works on the following platforms:

- Debian 32 bits / 64 bits (Hetch to current);
- Mandriva 32 bits / 64 bits (2005 to current);
- Windows 32 bits / 64 bits (XP, Vista, seven).

The development platform remains Linux, the Windows version is obtain by cross compilation under Linux (using the mingw tools). A native Windows version is planed.

The development infrastructure

Compilation infrastructure

Two alternative compilation infrastructures are present: the historic one (autotools + libtools) and the newcomer (CMake). For now, only the first infrastructure is running smoothly. It covers:

- The detection and configuration aspects of the platform;
- The dependency management of the sources;
- The generation of parallel makefiles;
- The regression tests;
- The library packaging.

The use of the autotools also greatly simplify the Linux packaging (.deb, .rpm). The CMake infrastructure is still in a beta state, but should greatly improve the configuration and compilation steps, and provide a way to compile the Windows version using Microsoft compilers, in order to easily reuse the C++ library in native Windows projects.

The development infrastructure

Versioning system

The versioning system used for the development of the whole platform is Subversion (SVN), on top of which stands a TRAC website.

- **Apache Subversion** is a software versioning and a revision control system issued from the Apache project. It is used for both the sources of the platform and the documentation, as well as for the development of modules.
- **Trac** is an open source, web-based project management and bug-tracking tool. Trac allows hyperlinking information between a bug database, revision control and wiki content. It also serves as a web interface to several revision control systems including Subversion.

Repositories

Three repositories are used for the development of the platform, its documentation and its modules. This choice has been made for the following reasons:

- The time scale is not the same for these three activities;
- The teams are different partly in term of people, but mainly in term of expertise.

The development infrastructure

Platform repository

This repository is in charge of both the C++ library source code and the python interface. It has the following organization, which is quite standard:

- A trunk that stores the source code of the upcoming version of the platform. The rule is to have only source code that pass with success all the tests embedded with both the library and the python module.
- Several development branches, dedicated to contributors or to specific developments. There are currently 6 active branches.
- A branch for the tags with one entry for each release candidate or official releases. 17 releases have been tagged so far.

The usage of this infrastructure is described in the Contribution Guide, one of the several documents that come with the platform. In particular, a specific role is assigned to an **integrator**, in charge of the merges from the different branches into the trunk.

The development infrastructure

Documentation repository

This repository is in charge of the whole documentation of the platform (10 different documents). It has been separated from the main repository by the middle of 2009 in order to allow for more frequent updates of the documentation with respect to the whole platform. It shares the same structure as the main repository:

- A trunk that stores the source code of the upcoming version of the documentation. The rule is to have only LaTeX source code able to generate the PDF version of all the documents.
- Several documentation branches, dedicated to contributors. There are currently 3 active branches.
- A branch for the tags with one entry for each release of the documentation. 4 releases have been tagged since July 2009.

The usage of this infrastructure is the same as for the main repository.

The development infrastructure

Modules repository

This repository is dedicated to the development of extra functionalities that have not yet been integrated into the main library, due to the lack of maturity of their dependencies for example. This repository is organized in a different way than the previous repositories:

- There is a branch for each module
- Within these branches, **the same organization as the main repository is proposed** but the developers are free to organize the things the way they want. Nevertheless, the commitment to the standard organization should ease the future integration into the mainstream development.
- For now, there are 2 active modules and one template.

Trac interface: the timeline

The screenshot displays the Trac interface for OpenTURNS. The top navigation bar includes links for Wiki, Timeline, Roadmap, and Browse Source. The main content area shows a timeline of development activity, organized by version ranges. The timeline entries include ticket numbers, descriptions, dates, and authors. For example, under the 0.10.0/11 section, there are tickets for missing functions, missing files, and changes to the documentation. The sidebar on the right provides a filter for changes from version 0.10.5/11, with checkboxes for various criteria like 'opened and closed tickets' and 'milestones reached'.

Trac interface: the source navigator

The image is a screenshot of a web browser window displaying the OpenTurns project page on SourceForge. The browser's address bar shows the URL 'http://trac.openturns.org/changest/1730/openturns/trunk'. The page features the OpenTurns logo at the top left. Below the logo, there is a search bar and a navigation menu with links like 'Home', 'Trac', 'Wiki', 'Downloads', 'FAQ', 'Contact', 'Sponsors', 'Partners', 'Links', 'About', 'FAQ', 'Contact', 'Sponsors', 'Partners', 'Links'. The main content area is titled 'Changeset 1730 in openturns for trunk' and shows a list of recent changes. Each change entry includes a timestamp, author, and a brief description of the changes. For example, one change by 'MORSE' on 12/26/11 at 14:59:44 lists several modifications to the 'Acoustic' module. To the right of the changes list, there is a sidebar with a 'View Differences' section showing a comparison of two versions of a file. At the bottom of the page, there is a 'Files' section displaying a directory tree of the project files. The browser's status bar at the bottom shows the current page is 'Changeset 1730 for trunk - OpenTurns - Mozilla Firefox'.

Trac interface: the bug tracking (1/2)

OpenTURNS - OpenTURNS - Mozilla Firefox

OpenTURNS

logged in as regis.l@open-tURNS.net

WII Timeline Roadmap Browse Source New Tickets New Ticket Search

How items per page: 100

Update

(1) Active Tickets (21 matches)

- List all active tickets by priority.
- Order each row based on priority.

defect (11 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#252	Can't close the window opened by Showgraph() function	python module	0.13.2	not defined	laporte@...	new	09/10/16
#271	python installation directories	extra module	0.13.2	not defined	laporte@...	reopened	08/30/16
#284	python file de la incorrectly set at configuration when using user specific python installation	general	0.13.2	not defined	ducha	assigned	12/03/16
#292	Custom Bug report #2018476 python-apt-get: creates a mess in sys path by adding its own namespace	extra module	SVN-HEAD	not defined	ducha	reopened	01/26/11
#293	hardcoded R library path	build / install process	SVN-HEAD	not defined	ducha	new	10/25/16
#261	swig detection fails	build / install process	SVN-HEAD	not defined	ducha	new	10/25/16
#227	incompatibility OpenTURNS (mpartFromCVF4) and Matlab/SymPy	general	0.13.1	not defined	laporte@...	reopened	10/14/16
#275	missing generation of swig runtime.h from cmake	build / install process	SVN-HEAD	not defined	ducha	new	10/13/16
#276	cmake build fails	general	SVN-HEAD	not defined	ducha	new	10/13/16
#287	Error when using OT module with debian version of OT	general	0.13.2	not defined	ducha	new	10/13/16
#289	PythonWrappingFunctions.hax missing and namespace error for SWI.StorageManager when trying to compile at-extended with OT trunk[1842]	extra module	SVN-HEAD	not defined	ducha	new	09/30/11

enhancement (7 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#286	Python wrappers for the modules	extra module	0.13.2	not defined	ducha	new	01/07/11
#289	Move the UncertainFactory class from the Base namespace to the Uncertainty namespace	library	0.13.2	not defined	ducha	new	01/07/11
#261	Installation of complex examples	build / install process	SVN-HEAD	not defined	ducha	new	01/26/11
#65	Add a new section to the UserGuide guide dedicated to functionalities not directly linked with the methodology	documentation	0.11.2	2010 work	anna.duffy@...	new	12/22/07
#67	Add a section dedicated to numerical methods parameters in the User Manual	documentation	0.11.2	2010 work	anna.duffy@...	new	12/22/07
#148	Random variable which parameters are random variables	general	0.12.1	not defined	laporte@...	new	10/16/08
#257	Output files' recovery for deterministic calculation.	library	0.13.2	not defined	ducha	assigned	04/13/13

task (3 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#72	Parallel computations	library	0.11.1	2010 work	ducha	assigned	11/29/07
#73	Add more examples to the library	general	2010 work		regis.l@open-t...	new	11/29/07
#54	Internationalization	general	0.11.2	2010 work	ducha	new	12/21/07

Note: See [TracReports](#) for help on using and creating reports

Download in other formats: RSS Feed | Camera deleted text | Tab-delimited Text | SQL Query

Powered by Trac 0.12.4multiplex/1082 by Egroupware Software

Logout Information

OpenTURNS

Centre de Contrôle de M... Enacs

(1) Active Tickets - OpenTURNS

Timeline1.png - KSnapshot

Husique - mplayer

22:07

Trac interface: the bug tracking (2/2)

The screenshot displays the Trac web interface for OpenTURNS. At the top, there's a navigation bar with links like 'OpenTURNS', 'Documentation', 'Downloads', 'Contact', and 'Help'. Below this, a search bar and a 'New Ticket' button are visible. The main content area shows 'Ticket #299 (new defect)' with the title 'PythonWrappingFunctions.hxx missing and namespace error for XMLStorageManager when trying to compile ot-mixedmod with OT trunk (1842)'. The ticket is reported by 'richard.babin' and assigned to 'della'. The description includes a code snippet for a Makefile.am and a list of error messages from the compiler. The ticket is marked as 'unassigned' and 'not defined'. The bottom of the page shows a Windows taskbar with various open applications like 'Centre de Contrôle de M...', 'Emacs', and 'PythonWrappingFunctions...'.

Ticket #299 (new defect)

PythonWrappingFunctions.hxx missing and namespace error for XMLStorageManager when trying to compile ot-mixedmod with OT trunk (1842)

Reported by: richard.babin
Priority: unassigned
Component: extra module
Keywords: EC

Assigned to: della
Status: not defined
Version: 2.10.0

Description

Hi,

I am still trying to compile my module (ot-mixedmod) with trunk version of OT (1842). And I have two problems :

- It seems that PythonWrappingFunctions.hxx must be in the installed include files. Here a method to do that (I am not sure that it is the right method to do that).

```
diff --git a/python/src/Makefile.am b/python/src/Makefile.am
... a/python/src/Makefile.am
+++ b/python/src/Makefile.am
@@ -35,8 +35,8 @@ endif

otwsgdr = $(includedir)/openturns/swig
otwinclude_dir = $(includedir)/openturns
...notinst_HEADERS = PythonNumericalMathEvaluationImplementation.hxx PythonWrapping
...otinclude_HEADERS =
...notinst_HEADERS = PythonNumericalMathEvaluationImplementation.hxx
+otinclude_HEADERS = PythonWrappingFunctions.hxx

BUILT_SOURCES = swig_runtime.hxx
pkgpyexec_PYTHON =
```

... after that I have the following error

```
otmixedmod_wrap.cpp: In function 'void* _p_OpenTURNS_Base_Common_XMLStorageM
otmixedmod_wrap.cpp:16409: error: 'XMLStorageManager' is not a member of 'OpenT
otmixedmod_wrap.cpp:16409: error: expected primary-expression before '}' token
otmixedmod_wrap.cpp:16409: error: expected ';' before ':' token
otmixedmod_wrap.cpp:16409: error: expected '}' before ':' token
```

Any help will be appreciated to solve it. :)

Attachments

Terminal

Conclusion

- A quite mature project: 6 full years of development
- A structured (rigid ;-)? development process
- A working infrastructure to help the developer in his/her hard job!