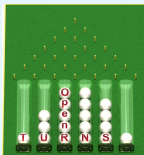


OpenTURNS Developer Training: the platform overview

Trainer : Régis LEBRUN
EADS/IW/SE/AM
regis.lebrun@eads.net

Developers training



Objectives

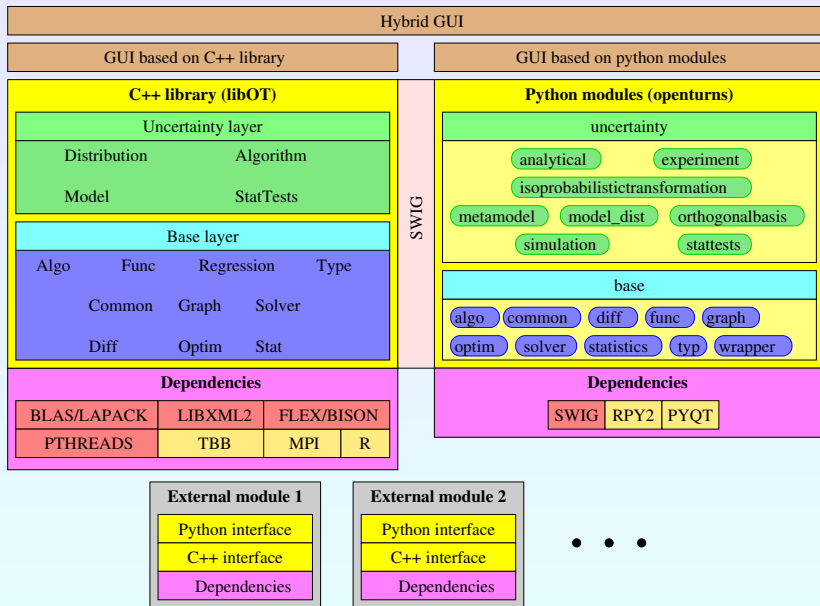
The objectives of this course is to give a broad overview of the OpenTURNS project and the resulting platform. We will cover the following points:

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

2005-2011: 7 years of partnership

- 2005 Conception, setup of the compilation infrastructure, setup of the development environment, first base classes.
- 2006 main development of the C++ library
- 2007 python binding. First release the 10th of May (release 0.9.0). Web site. 6 additional releases (0.9.1, 0.9.2, 0.10.0, 0.11.0, 0.11.1, 0.11.2).
- 2008 4 releases (0.11.3, 0.12.0, 0.12.1, 0.12.2, 0.12.3). Main new features: more distributions, more wrapping facilities.
- 2009 2 releases (0.13.0, 0.13.1). Main new features: multithreaded wrappers, new algorithms, polynomial chaos expansion.
- 2010 1 release (0.13.2). First windows port. Modularization of the python binding. Better integration of the data model with python native structures. Work on the parallelization of the algorithms. First work on stochastic processes and random fields.
- 2011 Sparse chaos implementation. Coming soon: first development of stochastic processes.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 shows the Trac interface for OpenTURNS, displaying a timeline of changes and tickets. The browser window title is "Timeline - OpenTURNS - Mozilla Firefox". The address bar shows "http://trac.openturns.org/timeline". The page has a navigation bar with links like "Wiki", "Timeline", "Roadmap", "Browse Source", "View Tickets", and "Search". The "Timeline" tab is selected. The main content area is titled "Timeline" and shows a list of changes and tickets, grouped by date ranges. A sidebar on the right allows filtering changes by date range and type (Opened and closed tickets, Regulatory changes, Milestones reached, Wiki changes). The bottom status bar shows the system clock as 22:01 on June 25, 2011.

Timeline

View changes from: 2012/1/11 and 10 days back done by: ☐ Opened and closed tickets ☐ Regulatory changes ☐ Milestones reached ☐ Wiki changes

2012/1/11:

- 12:25 Ticket #299 (PythonMapping functions have missing and namespace error for ...) created by roblawson
- 12:50 Ticket #298 (Missing .j files in python/scholarship) closed by schueller
- 13:00 ChangeSet [1842] by schueller
- 13:00 ChangeSet in openturns [1842] by schueller
- 13:00 ChangeSet in openturns doc [1829] by schueller

2012/7/11:

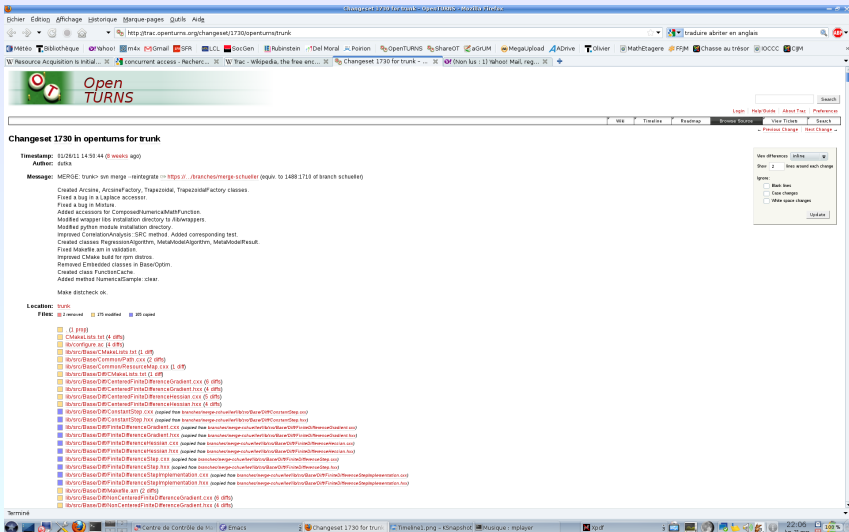
- 13:00 ChangeSet in openturns-modules [34] by roblawson
- 13:00 Ticket #298 (Missing .j files in python/scholarship) created by roblawson
- 13:00 ChangeSet in openturns doc [1829] by schueller
- 13:00 ChangeSet [1841] by schueller
- 13:00 ChangeSet in openturns [1841] by schueller

2012/6/11:

- 12:00 ChangeSet [1840] by schueller
- 12:00 ChangeSet in openturns [1840] by schueller

2012/5/11:

- 12:00 ChangeSet [1839] by schueller
- 12:00 ChangeSet in openturns [1839] by schueller
- 12:00 ChangeSet [1838] by schueller



Trac interface: the bug tracking (1/2)

The screenshot displays the OpenTURNS Trac interface. At the top, there's a navigation bar with links like 'Bilder', 'Edition', 'Offichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below this is a search bar and a list of active tickets. The main content area is divided into three sections: 'defect', 'enhancement', and 'task'. Each section contains a table of tickets with columns for Ticket, Summary, Component, Version, Milestone, Owner, Status, and Created. The 'defect' section shows tickets related to installation and build issues. The 'enhancement' section shows tickets for new features and improvements. The 'task' section shows tickets for documentation and testing. At the bottom, there's a footer with the Trac logo and a list of links.

(1) Active Tickets (11 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	10/20/10
#271	python installation directives	extra module	0.13.2	not defined	laporte@...	reopened	08/20/10
#284	python lib is incorrectly set at configuration when using user specific python installation	general	0.13.2	not defined	dutka	assigned	12/20/10
#292	Distion bug report #80587's python-openturns creates a mess in sys path by adding its own namespace	extra module	SVN-HEAD	not defined	dutka	reopened	10/20/11
#290	hardcoded R library path	build / install process	SVN-HEAD	not defined	dutka	new	10/20/10
#281	swig detection fails	build / install process	SVN-HEAD	not defined	dutka	new	10/20/10
#227	Incompatibility OpenTurns (ImportFromCSVFile) and MatlabbyPython	general	0.13.1	not defined	laporte@...	reopened	10/14/09
#275	missing generation of swig runtime from cmake	build / install process	SVN-HEAD	not defined	dutka	new	10/20/10
#276	cmake build fails	general	SVN-HEAD	not defined	dutka	new	10/13/10
#287	Error when using OT Modules with debian version of OT	general	0.13.2	not defined	dutka	new	12/10/10
#293	PythonWrappingFunctions.hox missing and namespace error for XMLStorageManager when trying to compile ot-merged with OT trunk(1842)	extra module	SVN-HEAD	not defined	dutka	new	10/10/11

enhancement (7 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#288	Python subscripts for the modules	extra module	0.13.2	not defined	dutka	new	10/17/11
#289	Move the LinearModelFactory class from the Base namespace to the Uncertainty namespace	library	0.13.2	not defined	dutka	new	10/17/11
#291	Installation of complex examples	build / install process	SVN-HEAD	not defined	dutka	new	01/26/11
#261	Add a new section to the UserCase guide dedicated to functionalities not directly linked with the methodology	documentation	0.11.2	2010 work	anna.dutka@...	new	12/22/07
#167	Add a section dedicated to numerical methods parameters in the User Manual	documentation	0.11.2	2010 work	anna.dutka@...	new	12/22/07
#148	Random variable which parameters are random variables	general	0.12.1	not defined	laporte@...	new	10/14/09
#257	Output files' recovery for deterministic calculation.	library	0.13.2	not defined	dutka	assigned	04/13/13

task (3 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#72	Parallel computations	library	0.11.1	2010 work	dutka	assigned	11/29/07
#73	Add more capsize to the library	general	2010 work	2010 work	regis.kohn@...	new	11/29/07
#84	Internationalization	general	0.11.2	2010 work	dutka	new	12/21/07

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

Download in other formats: RSS Feed | Camera-optimized Text | Tab-separated Text | SQL Query

Powered by Trac 0.12.4multipage-1082 by Eloquent Software

Terminal

Centre de Contrôle de Vol - Ennac

(1) Active Tickets - OpenTURNS

Timeinet.png - KSnapshot

Musique : rplayer

ipdf

22:07

Trac interface: the bug tracking (2/2)

The screenshot displays the OpenTURNS Trac interface for ticket #299. The browser window title is "PythonWrappingFunctions.hxx missing and namespace error for XMLStorageManager when trying to compile a stream with OT (2018/11/29) - OpenTURNS - Mozilla Firefox". The address bar shows "http://trac.openturns.org/ticket/299". The OpenTURNS logo is at the top left. The ticket title is "Ticket #299 (new defect)" and it is marked as "Closed 3 days ago". The ticket details include: Reported by: richard.bailly, Owned by: baill, Priority: unspecified, Milestone: not defined, Component: extra module, Version: SVN-HEAD, Keywords: C++, and Description: Hi, I am still trying to compile my module (ot-mixed 1.34) 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
@@ -25,8 +25,8 @@ endif

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

BUILT_SOURCES = swig_runtime.hxx
pluginsrc_PYTHON =
```

- after that I have the following error

```
otmixed_wrap.cpp: In function 'void* _p_OpenTURNS_Base_Common_XMLStorageManager':
otmixed_wrap.cpp:16409: error: 'XMLStorageManager' is not a member of 'OpenTURNS'
otmixed_wrap.cpp:16409: error: expected primary-expression before '}' token
otmixed_wrap.cpp:16409: error: expected ';' before 'x'
otmixed_wrap.cpp:16409: error: expected '}' before ':' token
```

Any help will be appreciated to solve it. :)

The interface also shows an "Attachments" section and a "Terminal" window at the bottom with the following content:

```
Terminal
22:08
Centre de Contrôle de la Vitesse
Emacs
PythonWrappingFunctions
KSnapshot
Musique: mplayer
ipdf
```

- 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!