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

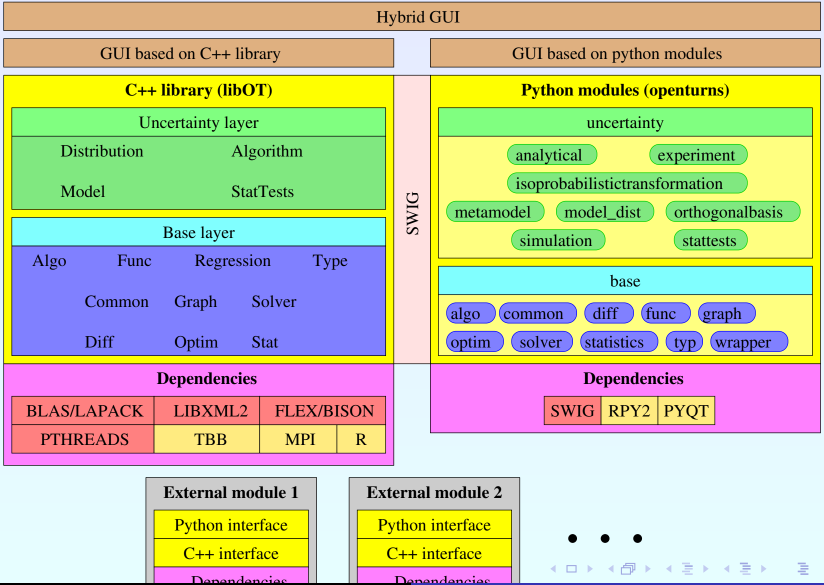
2015 Vectorial kriging, HMat support

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 spped-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 interation 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 adress 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 newcommer (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 curently 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.

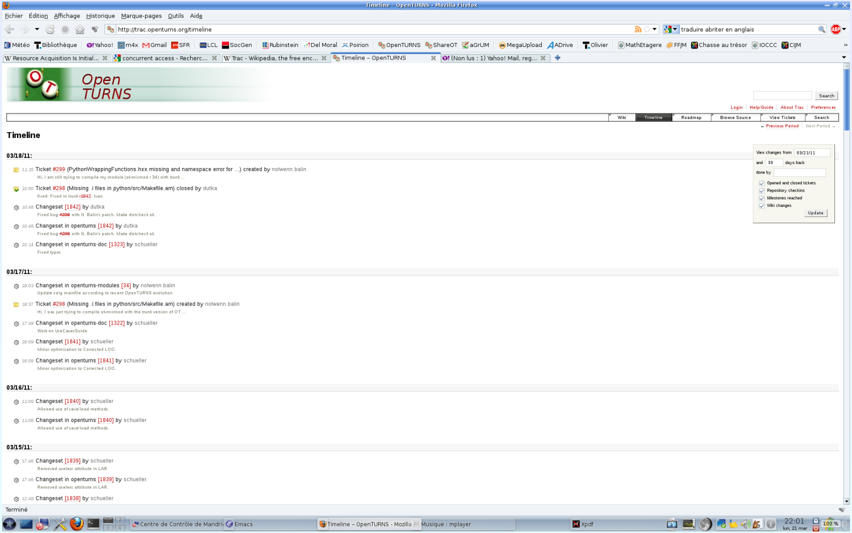
The development infrastructure

Modules repository

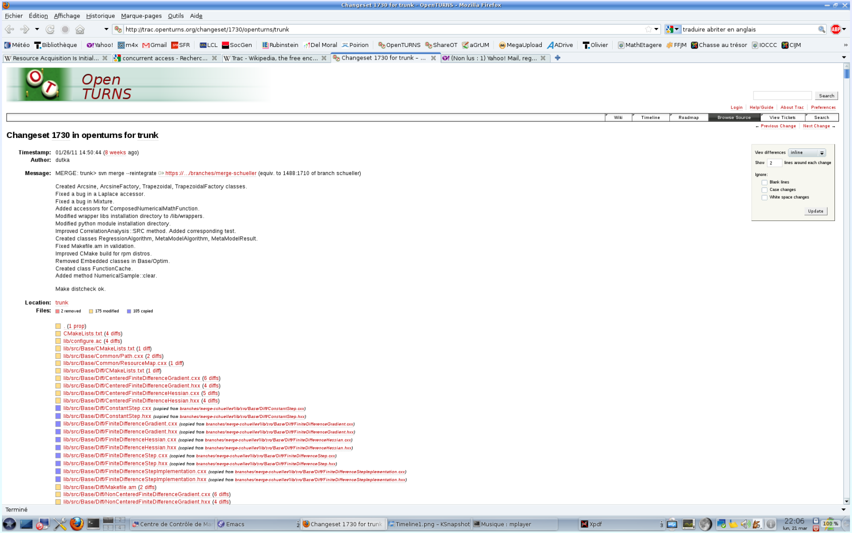
This repository is dedicated to the development of extra functionalities that have not yet been integrated into the main library, due to the lack of maturity of their dependencies for example. This repository is organized in a different way than the previous repositories:

- 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.

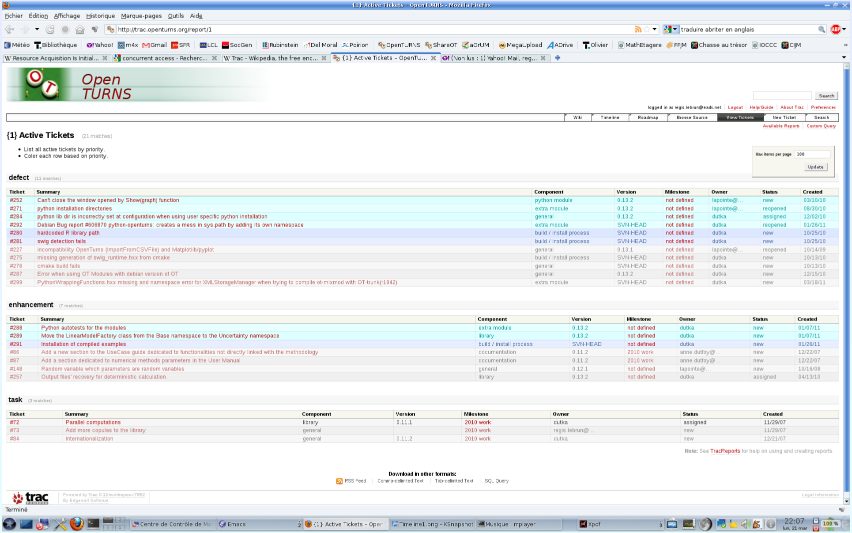
Trac interface: the timeline



Trac interface: the source navigator



Trac interface: the bug tracking (1/2)



{1} Active Tickets (21 matches)

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

defect (11 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#252	Can't close the window opened by Show(graph) function	python module	0.13.2	not defined	lapointe@…	new	03/10/10
#271	python installation directories	extra module	0.13.2	not defined	lapointe@…	reopened	08/30/10
#284	python lib dir is incorrectly set at configuration when using user specific python installation	general	0.13.2	not defined	dutka	assigned	12/02/10
#292	Debian Bug report #606870 python-openturns: creates a mess in sys.path by adding its own namespace	extra module	SVN-HEAD	not defined	dutka	reopened	01/26/11
#280	hardcoded R library path	build / install process	SVN-HEAD	not defined	dutka	new	10/25/10
#281	swig detection fails	build / install process	SVN-HEAD	not defined	dutka	new	10/25/10
#227	incompatibility OpenTurns (ImportFromCSVFile) and Matplotlib/pyplot	general	0.13.1	not defined	lapointe@…	reopened	10/14/09
#275	missing generation of swig_runtime.hxx from cmake	build / install process	SVN-HEAD	not defined	dutka	new	10/13/10
#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	12/15/10
#299	PythonWrappingFunctions.hxx missing and namespace error for XMLStorageManager when trying to compile ot-mixmod with OT-trunk(r1842)	extra module	SVN-HEAD	not defined	dutka	new	03/18/11

enhancement (7 matches)

Ticket	Summary	Component	Version	Milestone	Owner	Status	Created
#288	Python autotests for the modules	extra module	0.13.2	not defined	dutka	new	01/07/11
#289	Move the LinearModelFactory class from the Base namespace to the Uncertainty namespace	library	0.13.2	not defined	dutka	new	01/07/11
#291	Installation of compiled examples	build / install process	SVN-HEAD	not defined	dutka	new	01/26/11
#86	Add a new section to the UseCase guide dedicated to functionalities not directly linked with the methodology	documentation	0.11.2	2010 work	anne.dutfoy@…	new	12/22/07
#87	Add a section dedicated to numerical methods parameters in the User Manual	documentation	0.11.2	2010 work	anne.dutfoy@…	new	12/22/07
#148	Random variable which parameters are random variables	general	0.12.1	not defined	lapointe@…	new	10/16/08
#257	Output files' recovery for deterministic calculation.	library	0.13.2	not defined	dutka	assigned	04/13/10

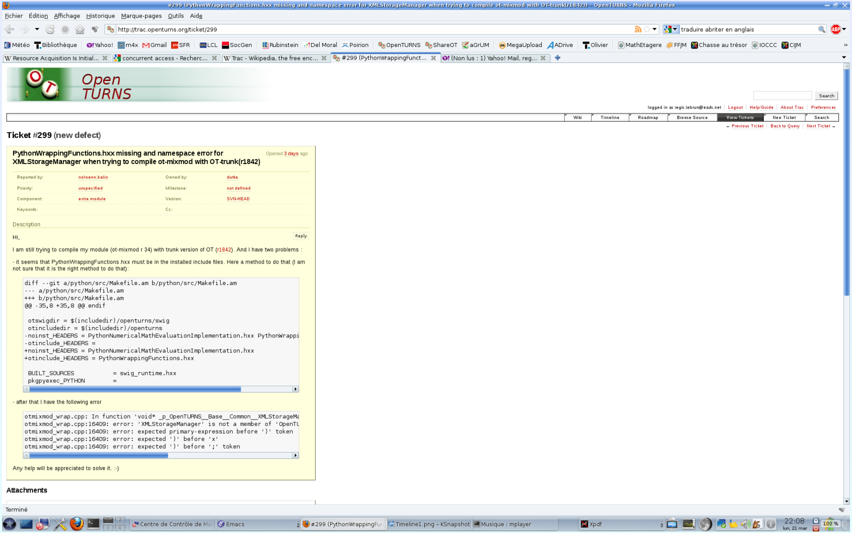
task (3 matches)

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 copulas to the library	general		2010 work	regis.lebrun@…	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 | Comma-delimited Text | Tab-delimited Text | SQL Query

Trac interface: the bug tracking (2/2)



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!