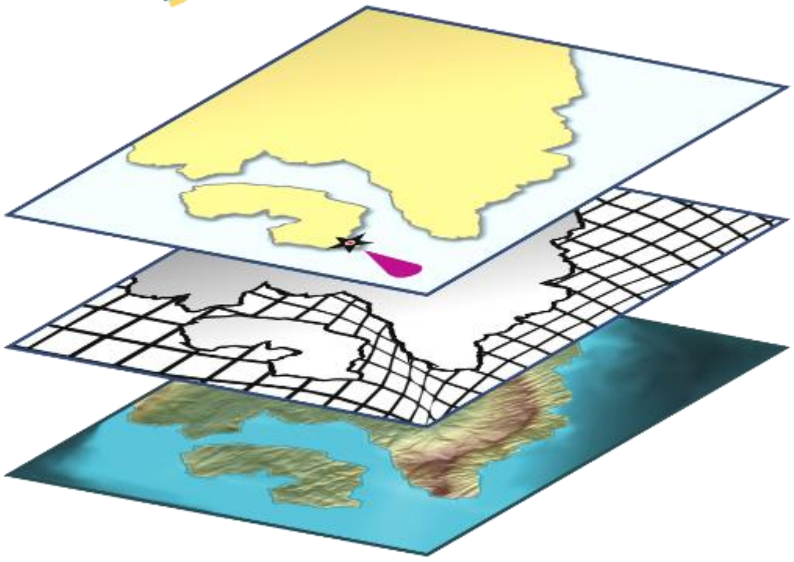
WHITE PAPER

HydrOffice



Giuseppe Masetti

Brian R. Calder

Matthew J. Wilson

Version 1.1 – March 1, 2017

UNH CCOM/JHC - Durham, NH, USA

EXECUTIVE SUMMARY

- HydrOffice is a research framework that collects libraries and tools for ocean mapping, from data acquisition to hydrographic survey review and nautical chart compilation.
- HydrOffice uses Python and C++ as programming languages with the intent to combine the flexibility and wide availability in the scientific community of the former with the faster runtime execution of the latter.
- HydrOffice has a modular structure so that only the required elements are installed for a given tool. To avoid conflicts with existing and future Python installations, HydrOffice makes full use of Python virtual environments.
- The development environment uses:
  - Git and Mercurial for version control systems.
  - GitHub and Bitbucket for code hosting services.
  - AppVeyor (Windows) and Travis-CI (Linux and OS X) for continuous integration services.

CONTENTS

RATIONALE

HydrOffice represents a collaborative effort led by the Center for Coastal and Ocean Mapping to develop a research software environment with applications to facilitate all phases of the ping-to-chart process: facilitate data acquisition, automate and enhance data processing, and improve hydrographic products.

The environment, by minimizing the efforts to kick-start and by easing the configuration management, facilitates the creation of new tools for researcher, students and in the field; and, potentially, it eases the industrialization of some of these tools. The overall goal is thus to speed up both algorithms testing and Research-to-Operation (*R2O*).

HydrOffice's wide scope is structured in three research themes:

- Facilitate Data Acquisition.
- Automate and Enhance Data Processing.
- Improve Hydrographic Product.

These themes drive the creation of a collection of *hydro-packages*, each of them dealing with a specific issue of the field.

HydrOffice has open licenses and encourages free contribution, and can facilitate development with an existing infrastructure and interface. Individual tools within HydrOffice are built in contained, modularized structures, such that they can be easily updated and maintained.

One of the main HydrOffice requirement is easiness in its extension. This goal is achieved by natively supporting a plugin architecture:

- Base packages provide with common boiler-plate code.
- Several hydro-packages where each one ship a few task-specific algorithms and can access common code from the base packages.

Furthermore, a skeleton package is provided with a base GUI to speed up and to ease the focus on the targeted weakness.

Finally, the individual tools in HydrOffice are usually also provided as "frozen", standalone, click-and-play solutions that require no installation on behalf of the user.

All of the HydrOffice applications are made available within Pydro (an in-house hydrographic environment developed and maintained by NOAA Office of Coast Survey's Hydrographic Systems and Technology Branch), to facilitate delivery to (and feedback from) NOAA users.

WHY DO WE USE PYTHON AS THE PROGRAMMING LANGUAGE OF CHOICE?

Python is a programming language of increasing popularity due to its flexibility, ease of use and simplicity to learn, portability, and cost (free)[1]. It has also a wide acceptance in the scientific community.

Python can easily interact with FORTRAN and C/C++ code, by using the 'extension' mechanism. Such a mechanism is mainstream in the Python ecosystem, where important libraries (such as Numpy) are largely based on extensions for reasons of computational speed.

Python easily interacts with R, another popular language in the scientific community that more specialized for statistical data analysis[2].

[1] For a comparison with Matlab: http://www.pyzo.org/python_vs_matlab.html.
[2] For a basic comparison of Python and R: http://blog.udacity.com/2015/01/python-vs-r-learn-first.html. The python package "rpy2" provides an interface to R from Python: https://rpy2.bitbucket.io/

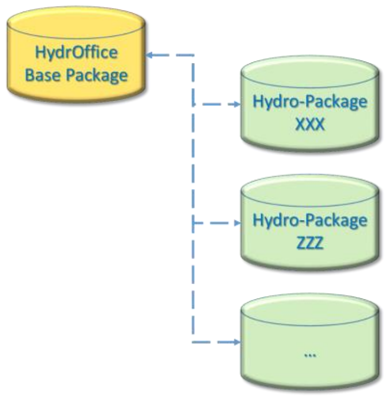
SELECTED SOLUTION

This section briefly outlines some aspect of the required solution.

CODE ARCHITECTURE

The driving criterion for the code architecture is the requirement to support packages (Python's name for what are called libraries in C/C++) as plugins.

Base packages are provided with common boiler-plate code. This base package are used by task-specific packages, called *hydro-packages*. Furthermore, a skeleton package is provided with a base GUI so that it can be quickly developed and easily extended.



Thus, the idea is to have the same base package used by multiple hydro-packages. A hydro-package is thus only dependent on the adopted base packages.

The requirement here is to make it possible for the user to install only the needed packages. Furthermore, having a base package permits both code re-use and the sharing of a similar GUI look-and-feel.

NAMING CONVENTION

HydrOffice was formed as the contraction of two terms: "hydro" and "office":

- The first term is used to underline the main scope of the suite, hydrography.
- The second term is used to limit such a broad scope to applications that are commonly used in a hydrographic office to support data acquisition, to review a survey or to facilitate the chart building and updating workflow.

LICENSE

During the incubation phase, a project may use the following conservative license:

```
Copyright (c) 2017 PydrOffice's Authors


All Rights Reserved


THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF PydrOffice's Authors


The copyright notice above does not evidence any actual or intended publication of
such source code.
```

BASE PACKAGE

Once stable, the base package is usually made available under two different licensing options designed to accommodate the needs of the various users:

- GNU LGPL version 3, a very permissive open source license.
- CCOM/JHC Industrial Associate license, appropriate for proprietary/commercial use, e.g., development of proprietary/commercial software where you do not want to share any source code with third parties or otherwise cannot comply with the terms of the GNU LGPL version 3.

HYDRO-PACKAGES

Each hydro-package may have the license judged appropriate by the authors. The default solution in the hydro-package skeleton is the same as the Base Package (i.e., dual license)[3].

[3] There is a large open debate about licensing; an interesting point of view is described here: http://giovanni.bajo.it/post/56510184181/is-gpl-still-relevant.

LOGO

The logo was designed with the intent to connect the real marine world to the nautical chart, with a symbolic intermediate layer to represent the office work.

HydrOffice

The icon simply shows an abbreviation of the project name.

HO

DISTRIBUTION

Each package, both base and hydro-, provides a "setup.XXXXXX.py" file that is used to build a *wheel*, a modern format for Python package distribution[4].

[4] More info on this type of Python packaging at http://pythonwheels.com/.

SUPPORTED PYTHON

There is not (currently) a good reason not to follow the current Python development:

- 2.7+ (both 32- and 64-bit)
- 3.5+ (both 32- and 64-bit)

PACKAGE REQUIREMENTS

The installation of the base package also takes care of setting up the Python environment, both dependencies and versions.

To avoid possible conflicts between different package versions and related requirements, a stand-alone Python installation is performed. Specifically, the popular *virtualenv* tool is used to encapsulate HydrOffice packages into a virtual environment[5].

[5] More info in *virtualenv* at https://virtualenv.pypa.io/en/latest/.

PACKAGES

This paragraph briefly describe base and hydro packages.

A package can be in five statuses, mimicking the nature itself of the research:

- Proposal: this represents the usual initial state, when the idea of something that could be useful is born but still the details are not well defined *[Phase: "Eureka!"]*
- Incubation: this phase permits a better definition of the research idea, and not all the projects move forward from incubation *[Phase: "I am working on such a good idea to make it a useful tool"]*.
- Stable: a package reaches this status after a good period of incubation in which the original research idea was developed and a useful tool is created and people use it (thus code maintenance is required) *[Phase: "I finally got a useful tool, and I care about the code maintenance"]*.
- Frozen: a tool with limited helpfulness. This status can be reached for several reasons: e.g., the developer has not time to maintain the hydro-package; a commercial implementation was implemented; etc. *[Phase: "That tool is getting outdated, since nobody has interest in maintaining the code"]*.
- Dead: for whatever reason, the package is no more useful and is not maintained *[Phase: "That tool was useful but now that other tool is the way to go"]*.

To reaffirm the obvious, not all the packages will cross the five statuses in order, but given the research focus of HydrOffice, they will reach the "Dead status" since the idea was not actually useful, properly implemented, or migrated in long-term maintained projects (e.g., commercial software).

DEVELOPMENT TOOLS

HydrOffice adopts a distributed approach, where each developer works directly with his or her own local repository, and changes are shared between repositories as a separate step.

VERSION CONTROL SYSTEM

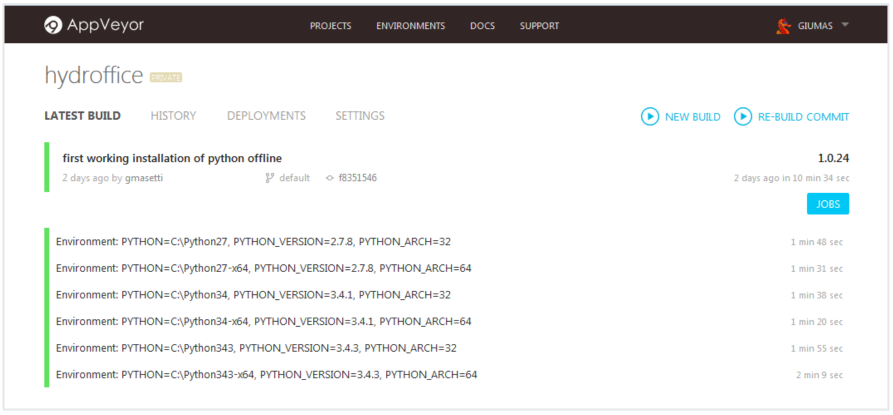
HydrOffice uses Git and Mercurial (Hg)[6].

CODE HOSTING SERVICE

HydrOffice packages are stored as private and public repositories on GitHub and Bitbucket.

CONTINUOUS INTEGRATION SERVICE

Each commit to a HydrOffice repository should trigger AppVeyor and Travis-CI to download and build the new code for testing in different 'clean' environments.



```
first working installation of python offline                                  1.0.24
2 days ago by gmasetti              default   f8351546      2 days ago in 10 min 34 sec

Environment: PYTHON=C:\Python27, PYTHON_VERSION=2.7.8, PYTHON_ARCH=32          1 min 48 sec
Environment: PYTHON=C:\Python27-x64, PYTHON_VERSION=2.7.8, PYTHON_ARCH=64      1 min 31 sec
Environment: PYTHON=C:\Python34, PYTHON_VERSION=3.4.1, PYTHON_ARCH=32          1 min 38 sec
Environment: PYTHON=C:\Python34-x64, PYTHON_VERSION=3.4.1, PYTHON_ARCH=64      1 min 20 sec
Environment: PYTHON=C:\Python343, PYTHON_VERSION=3.4.3, PYTHON_ARCH=32         1 min 55 sec
Environment: PYTHON=C:\Python343-x64, PYTHON_VERSION=3.4.3, PYTHON_ARCH=64     2 min 9 sec
```

[6] Some reasons for choice of Hg: http://blogs.atlassian.com/2012/02/mercurial-vs-git-why-mercurial/.