

The Sphinx Project

Based on a True Story

Robert Lehmann

robert.lehmann@student.hpi.uni-potsdam.de



Project documentation

September 1, 2011

supervised by Martin von Löwis

CONTENTS

1 Sphinx in a Nutshell	1	3.4 Jacob Mason	5
2 History	2	4 Interlude on Internationalization	6
3 Process	2	5 Case study	6
3.1 Georg Brandl	4	6 Closing remarks	7
3.2 Armin Ronacher	5		
3.3 Daniel Neuhäuser	5		

ABSTRACT

Sphinx is an open source documentation tool commonly used for software libraries in the Python community. We will discuss its community-driven but patriarchic development process and its open nature under formal and historical aspects. A sample set of bugs reported against it and fixed by volunteer contributors will shed light on how it is maintained.

*~

1 SPHINX IN A NUTSHELL

Sphinx transforms documents composed in a lightweight markup language to a number of different output formats, among them HTML, Adobe PDF, L^AT_EX, EPUB, and Troff. It extends the Docutils framework¹, which readily parses reStructuredText to an intermediate document tree structure, applies its own transformations² and provides its own writers.³

It is a vast improvement over bare Docutils as it supports structured document hierarchies, whereas Docutils is merely useful for single files. Additional features include automatic index generation, full-text search, syntax highlighting, and themes.

Sphinx embodies about 21,000 lines of Python code and another 4,000 lines of tests. Over the five years of its development it massed up 250,000 churns⁴.

¹<http://docutils.sf.net>

²Docutils Transforms operate on a document tree and can modify the contents and structure as they please. Substitutions and localization, eg., are implemented in that manner. For a list of Docutils Transforms, check <http://docutils.sourceforge.net/docs/ref/transforms.html>.

³Details on the build process are documented at <http://sphinx.pocoo.org/ext/tutorial.html#build-phases>.

⁴Churns are changed lines of code in Mercurial-speak.

Basic COCOMO yields a cost estimate of about \$400,000, considering Sphinx an *Organic Project*. A rough Expert COCOMO II assessment⁵ classifies Sphinx as a low-risk project (4% risk) with most of the risk in process and schedule failure, which neatly models our perception of the project — the 1.0 stable release, for example, was postponed several times.

2 HISTORY

Sphinx has organically grown to much of what it is today and its workflow is explained in its early roots. It was originally conceived as a replacement for Python's existing documentation toolchain, which was based entirely on L^AT_EX at the time. Among other criticisms about the maintainability and approachability of the legacy documentation the build process also became significantly easier with Sphinx. [1]

Sphinx was not written for publication initially and was announced only as a by-product of the new Python 2.6/3.0 documentation in August 2007. [2] It was previously known as `py-rest-doc`[1] and shipped with a full-fledged tool to convert TeX sources to reST files. Near its first release in March 2008 [3] that part of the code base was purged completely. ⁶ As of writing, the latest stable release is version 1.0.7. [4]

It has been adopted by the Python community at large. Prominent users include but are not limited to the Pocomo projects (Flask, Jinja), other Python projects (Zope, Django, Bazaar, and CPython itself), and even several books (The `repoze.bfg` Web Application Framework, Pomodoro Technique Illustrated).⁷ Services like *Read The Docs* offer to host Sphinx projects for free.⁸

3 PROCESS

Sphinx, in addition to being licensed under the BSD license, has an open process: its source code is freely available on Bitbucket⁹ and its development is discussed on `sphinx-dev` at Google Groups¹⁰.

⁵24,000 lines of code, 20,000 reused lines (30% integration, 1% assimilation), XH/X-H/N/VH/L, VL/N/N/N/H, N/N/L, H/VH/N/H/N/VH, VL/H/VL.

⁶It lives on in a Python repository at <http://svn.python.org/view/doctools/converter/>.

⁷An extensive list is available at <http://sphinx.pocoo.org/examples.html>.

⁸<http://readthedocs.org>

⁹Bitbucket is a code hosting platform for Mercurial repositories operated by Atlassian, Inc. Sphinx is located at <http://bitbucket.org/birkenfeld/sphinx>.

¹⁰<http://groups.google.com/groups/sphinx-dev>

It has participated in the Google Summer of Code™ under the umbrella of the Python Software Organization.¹¹ The paper author worked on *Sphinx Native Language Support: Toolchain for Creating, Tracking and Viewing Internationalized Versions of Sphinx Documents*¹² during the summer of 2010 and was granted commit access in October that year.

*
**

Sphinx' development style models that of GNU Wingnut[5]: individual patches are merged back into the upstream repository fairly quickly and seldom distributed in a standalone fashion. There is no huge distinction between developers and users — everyone can file a ticket and attach a patch (or fork). Technically, all developers can commit to all parts of the code; Pull Requests further blur the lines of commit rights.

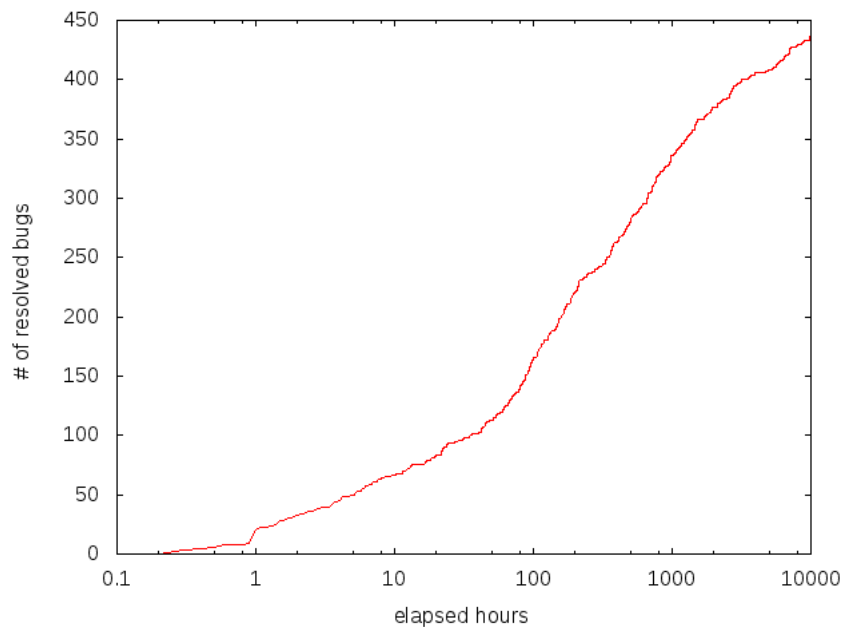


Figure 1: Of the 650 bugs (plus 100 invalid tickets), most were closed fairly quickly.

It is interesting to note that about half a dozen of contributors have also contributed to Docutils in the past; almost all POCO team members have contributed to Sphinx in one way or another. The community runs a repository

¹¹<http://code.google.com/soc>

¹²<http://gsoc.robertlehmman.de>

of about 40 Sphinx Extensions¹³. There are another 100 forks of Sphinx (not all of which really contain incoming commits) hosted on Bitbucket and about 400 users are watching mainline activity.¹⁴ Except for one unfortunate case, all pull requests were merged into the mainline repository.

Commit activity is distributed in a Zipfian way. Development impulses and bug fixes are largely (and especially lately) driven by the community; for the 1.0 release, the Japanese community had a dinner party. [6] However, there are some key figures we will discuss and classify according to Nakakoji and Yamamoto's *Roles of Community Members*. [5, p. 4]

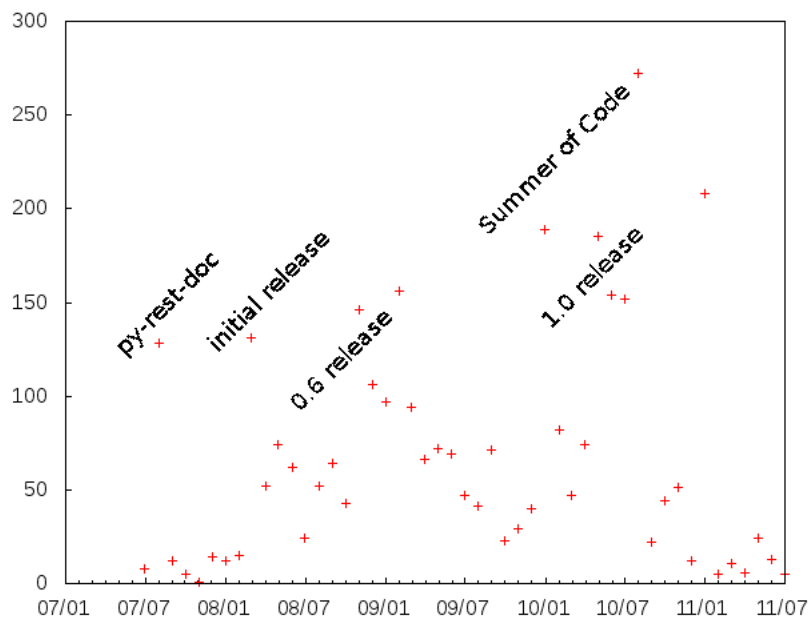


Figure 2: Commit activity by month

¹³Extensions are non-core components which can hook into the build process to provide new functionality. See <http://sphinx.pocoo.org/extensions.html> for details. sphinx-contrib is located over at <http://bitbucket.org/birkenfeld/sphinx-contrib>.

¹⁴Forks can be found at <http://bitbucket.org/birkenfeld/sphinx/descendants>, watchers are listed at <http://bitbucket.org/birkenfeld/sphinx/zealots>.

3.1 Georg Brandl

Georg Brandl¹⁵ is inarguably the **Project Leader** and Creator of Sphinx. He earned a degree in Physics at Technical University Munich¹⁶ from 2005 to 2010 and became a Python Core Developer the same year, where he managed the 3.2 release. He is a member of the Pycoc team¹⁷ since 2004.

In 2008, he received the Python Software Foundation Community Award for *“building the Sphinx documentation system as an alternative to the LaTeX-based system [they] had been using previously, and converting the Python documentation to use it.”*¹⁸

Over the years he accumulated 205,000 churns¹⁹ in over 2500 commits, which renders him the by far and large most active Sphinx developer.

3.2 Armin Ronacher

Armin Ronacher is another Pycoc team member and was a very early contributor to Sphinx. He originally wrote its Web serving component and could thus be classified as a Core Member. That code was removed quickly²⁰ after the first Sphinx release in June 2008. Armin subsequently championed individual parts he was specially interested in as a Peripheral Developer, for example the JavaScript and C++ domains²¹. It earned him about 6,000 churns in barely over 60 commits. He has been inactive for the most part of 2011, operating as a Bug Fixer at best.

3.3 Daniel Neuhäuser

Daniel Neuhäuser was a participant in the Google Summer of Code 2010. He contributed Python 3 and versioning support to Sphinx and is probably closest to an Active Developer today. He achieved 6,000 churns in a little over 200 commits and subsequently joined the Pycoc team.

3.4 Jacob Mason

Jacob Mason was another Google Summer of Code 2010 participant. After his contribution of the latest Web package he has disappeared and can be consid-

¹⁵<http://pythonic.pocoo.org/about>

¹⁶<http://users.physik.tu-muenchen.de/gbrandl/>

¹⁷The Pycoc team is a loose collective of Python developers, responsible for popular software packages such as Werkzeug, Pygments and Jinja. <http://pocoo.org>

¹⁸<http://www.python.org/community/awards/psf-awards/#august-2008>

¹⁹Counted without revisions bfb5b73af019, e88201ce226b, and 2d7e85eoc7b4, which imported the Python docs.

²⁰<http://bitbucket.org/birkenfeld/sphinx/changeset/217966c6c467>

²¹Sphinx Domains are a concept of domain-specific documentation directives.

ered a Peripheral Developer. In a solid 100 commits he nearly accumulated 7,500 churns.

✱✱

4 INTERLUDE ON INTERNATIONALIZATION

Internationalization for programs is a solved problem for the most part [7] and has penetrated the F/OSS community at large: [8] `gettext` provides a slim but powerful interface to translate strings in program source code. [9] Translatable strings can be marked up as such during compile time, and replaced with a localized message at runtime.

Python has grown `gettext` support since Mailman made the leap in its 2.1 release. [10] Even though Sphinx uses another extraction mechanism²² it still sticks to the standards and file formats of the `gettext` family.

`sphinx-i18n` attempts to provide a `libintl`-esque mechanism for documents processed by Sphinx. If Sphinx succeeds to generate the standard `gettext` file formats, translators can stick to the toolchains familiar to them. Thus only message extraction and patching need to be reimplemented.

While message extraction is a seemingly simple problem there are some tidbits to consider: how long should a single message be? [11] How many messages should go into a catalog? In which ways are block- and paragraph-level markup handled? What about message context, references, hyperlinks?

✱✱

5 CASE STUDY

`sphinx-i18n` originated as a Summer of Code deliverable. Still, after all reasonable scrutiny, it was not ready for users at publication. Ian Lewis presented Sphinx' internationalization component at mini PyCon Japan²³ and noticed a subtle flaw in message catalogs: messages were unnecessarily shuffled around, not resembling their original order and context. Subsequently, Shirou Wakayama played around a bit more²⁴, reasoned that the behaviour was indeed a defect, and filed a ticket.²⁵

²²Python historically provided `pygettext` but the original `xgettext` utility has caught up to feature parity. Babel is used as a viable and extensible alternative to the `gettext` suite.
<http://babel.edgewall.org>

²³<http://slideshare.net/Ianmlewis/Sphinx-11-I18N>

²⁴<http://d.hatena.ne.jp/rudi/20110202/1296632643>

²⁵<http://bitbucket.org/birkenfeld/sphinx/issue/630>

A related regression²⁶ was reported one month later. Suddenly, a message could end up twice in a catalog which is not only invalid but also nonsensical: there is no way to pick one of the duplicate translations in case of a conflict. It can be fixed fairly easily by utilizing a data structure which (1) retains order and (2) stores unique items — a sorted set. While I worked on a patch, the magic of open source came into play: I received not less than four pull requests containing a total of 15 patches.

One set of patches was merged by Georg Brandl in the meantime, which moved committing an authoritative patch by the component maintainer out of question.

Coming up with a patch of its own would arguably have been much easier in that case. Every received solution exhibited its particular set of merits though and all authors put a lot of thought and consideration into their contribution. Additionally, there is a huge motivational and political aspect to pull requests: some authors have continued to submit improvements only after their initial patch set was merged into the mainline. Understanding all patches and cherry-picking individual features seemed like the most viable approach and worked out pretty well.

6 CLOSING REMARKS

While writing this paper we have received another four pull requests for the internationalization component alone. I would like to thank all contributors to Sphinx for making the community what it is and this experience possible.

REFERENCES

- [1] G. Brandl. (2007, June) Introducing py-rest-doc. [Online]. Available: <http://pythonic.pocoo.org/2007/6/23/introducing-py-rest-doc>
- [2] ——. (2007, August) reST python documentation live. [Online]. Available: <http://pythonic.pocoo.org/2007/8/17/rest-python-documentation-live>
- [3] ——. (2008, March) Sphinx is released! [Online]. Available: <http://pythonic.pocoo.org/2008/3/21/sphinx-is-released>
- [4] G. Brandl *et al.*, “Sphinx 1.0.7 released,” *sphinx-dev*, January 2011. [Online]. Available: <https://groups.google.com/group/sphinx-dev/msg/bb980739f33e357>

²⁶<http://bitbucket.org/birkenfeld/sphinx/issue/653>

- [5] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *Proceedings of the International Workshop on Principles of Software Evolution*, ser. IWPSE '02. New York, NY, USA: ACM, 2002, pp. 76–85. [Online]. Available: <http://doi.acm.org/10.1145/512035.512055>
- [6] G. Brandl, S. Yoshiki *et al.*, "Sphinx 1.0 final released," *sphinx-dev*, July 2010. [Online]. Available: <https://groups.google.com/group/sphinx-dev/msg/of3dda1c88ca14d4>
- [7] P. de Mauro, "Internationalizing messages in linux programs," *Linux J.*, vol. 1999, March 1999. [Online]. Available: <http://dl.acm.org/citation.cfm?id=327697.327701>
- [8] S. Turnbull, "Alphabet soup," *Linux J.*, vol. 1999, March 1999. [Online]. Available: <http://dl.acm.org/citation.cfm?id=327697.327699>
- [9] O. Y. Tykhomyrov, "Introduction to internationalization programming," *Linux J.*, vol. 2002, pp. 3–, November 2002. [Online]. Available: <http://dl.acm.org/citation.cfm?id=583910.583913>
- [10] B. A. Warsaw, "GNU Mailman, internationalized," in *Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2003, pp. 11–11. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1247340.1247351>
- [11] R. Lehmann. (2010, June) Fine or coarse? [Online]. Available: <http://gsoc.robertlehmann.de/fine-or-coarse/>