SymEngine and the Season of Docs

Rohit Goswami · 2021-03-25 · ramblings · thoughts · documentation

> *An overview of documentation complexity and an analysis of incentives.*

Background

As mentioned a few times before, last year a proposal of mine to improve the documentation for the SymEngine organization was accepted under the Google Season of Docs 2020 initiative. This is a more personal and expanded discussion on the report submitted on the SymEngine Wiki regarding the goals and completion metrics.

Figure 1: Promotional image Google seemed to strongly suggest

Documentation and Me

I have been a huge proponent of documentation throughout my decade long dabbling in FOSS projects. Quite bluntly, my memory is not great, and writing things down in a manner tailored to my own needs helps me reduce the amount of time it takes to do similar tasks. In many cases, the problem is not even the lack of existing documentation; it is just good to have a representation which encourages the minimum cognitive load for me.

SymEngine and Me

The SymEngine repository will turn 10 next year. There are few, if any, contenders in the field of C++ symbolic mathematics libraries; the Vienna math project seems to have stalled and does not support matrices.

Symbolic math is one of those fields I feel should be used a lot more in the applied sciences, but somehow gets overlooked. Part of this is undoubtedly because of Mathematica, Maple and other proprietary players, which stifle innovation and FOSS development, but also the documentation and technical debt incurred is pretty large.

Meeting the Mentors

Originally, the meeting with my mentoring group was rather anxiety inducing; this was mostly due to my own unfamiliarity with symbolic algebra. Over time though, the meetings quickly became something I started to look forward to; and we were able to collectively come up with a unique; yet sustainable documentation

workflow. I was also able to gain insights into the logic underlying the code which would have ordinarily taken longer to process on my own. I also ended up going over some references like Cohen (2003) and Cohen (2002).

Timelines and Deliverable Assets

One of the early issues I faced was restricting myself to documentation; with such an active and fast-growing project, there were several avenues I wanted to explore and improve, not all of which were technically under the ambit of documentation. This was an impulse which I eventually had to ignore; though the team were kind enough to strongly suggest submitting PRs in other directions after the documentation project concluded.

Doxygen Alternatives

Initially I had hoped to have an all-Sphinx documentation site, with all language bindings parsed into Sphinx. For this I explored exhale, as well as the more flexible doxyrest. None of these could match the flexibility or rich output of Doxygen for C++; so eventually I did the more rational thing and developed a nice theme for Doxygen instead, doxyYoda.

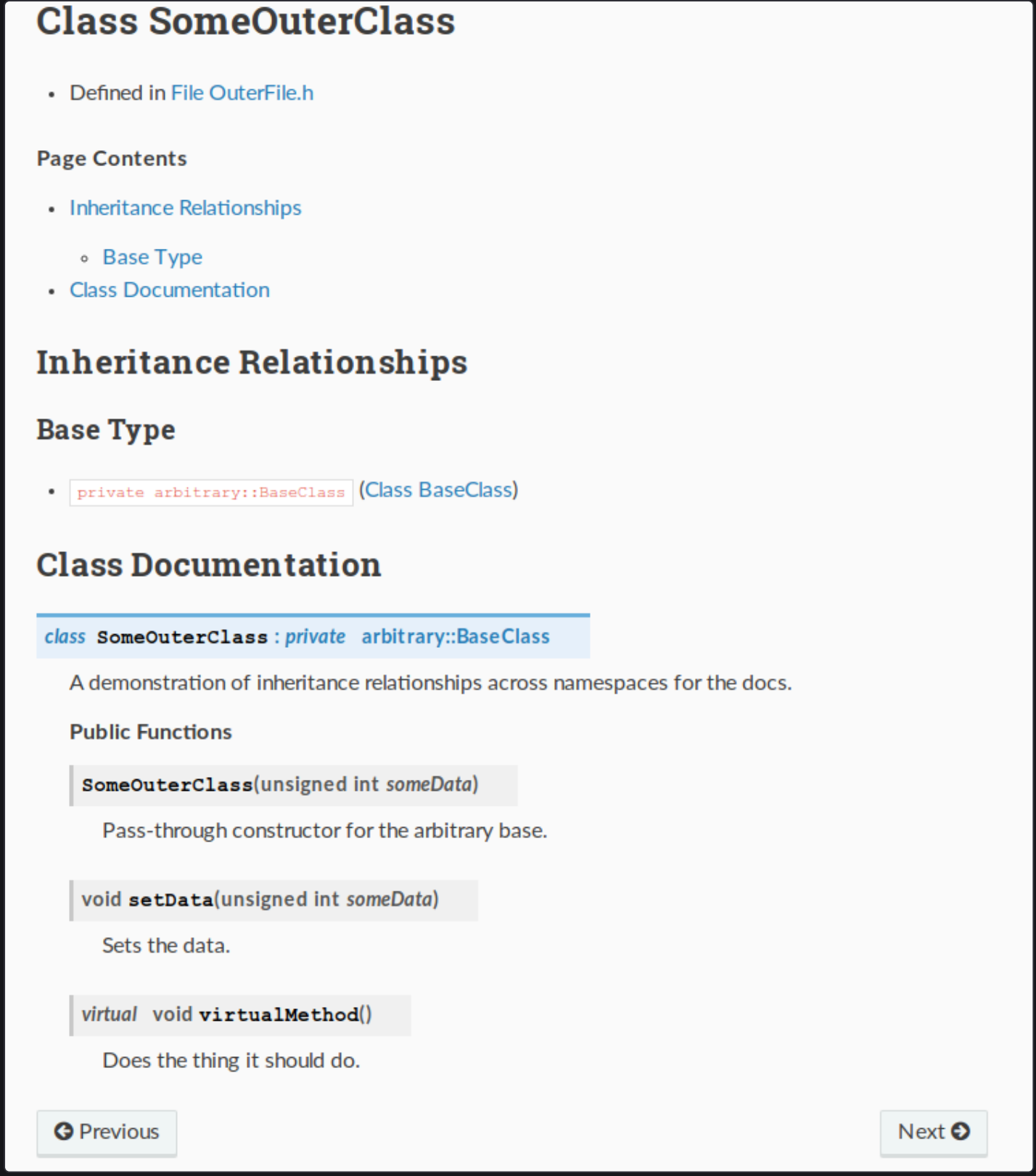
Base Doxygen

Figure 2: Base Doxygen

- Is ugly

Other than that, there were no standards for consistent documentation originally.

Exhale Documentation



Class SomeOuterClass

- Defined in File OuterFile.h

Page Contents

- Inheritance Relationships
  - Base Type
- Class Documentation

Inheritance Relationships

Base Type

- `private arbitrary::BaseClass` (Class BaseClass)

Class Documentation

class **SomeOuterClass** : *private* arbitrary::BaseClass

A demonstration of inheritance relationships across namespaces for the docs.

Public Functions

SomeOuterClass(unsigned int *someData*)

Pass-through constructor for the arbitrary base.

void **setData**(unsigned int *someData*)

Sets the data.

virtual void **virtualMethod**()

Does the thing it should do.

Previous Next

Figure 3: Exhale example

- Cannot include source code

That is a deal breaker, since the algorithms are often described step by step.

Doxyrest Documentation

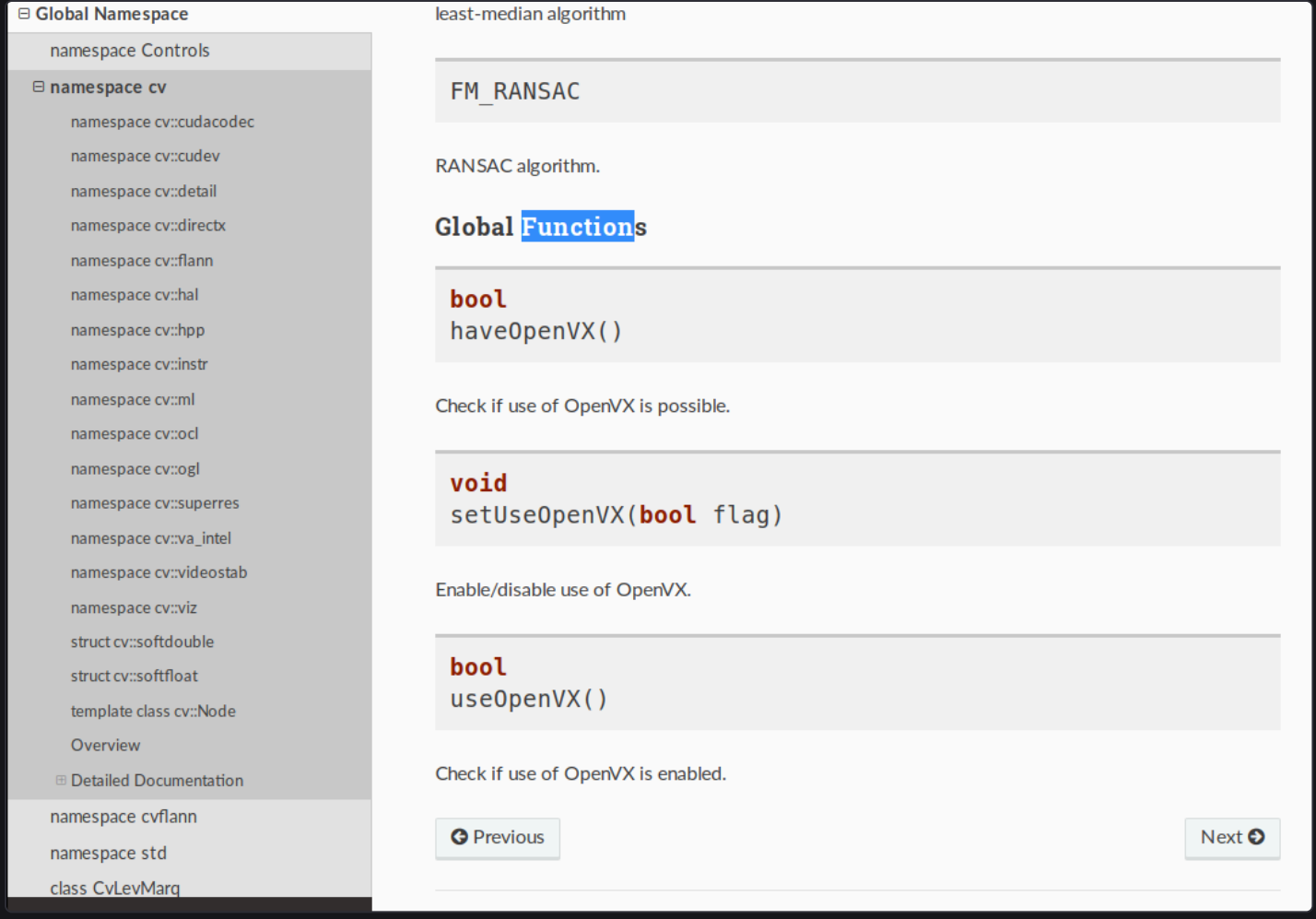


Figure 4: Doxyrest

- Includes more structure than exhale
- Can be extended to other source languages
- Has a rather complicated setup

Doxygen with DoxyYoda

Figure 5: Doxygen with doxyYoda

- All the goodness of Doxygen
  - Includes hierarchies

- Also is now pretty

Tooling Decisions

This is more of a technical specification; but after rooting around, it was decided to throw consistency of design to the wind and use native tools for each language binding as shown in Table 1.

Table 1: Language bindings and tools

Language **Package** ---

[R](https://symengine.org/symengine.R) [pkgdown](https://pkgdown.r-lib.org/)

[Python](https://symengine.org/symengine.py) [Sphinx](https://www.sphinx-doc.org/) [C++]

(https://symengine.org/symengine) [Doxygen](https://www.doxygen.nl/index.html) + [doxyYoda]

(https://github.com/HaoZeke/doxyYoda) [Julia](https://symengine.org/SymEngine.jl/) [Documenter.jl]

(https://juliadocs.github.io/Documenter.jl/stable/man/guide/) [Notebooks or MyST](https://symengine.org)

[Sphinx](https://www.sphinx-doc.org/) + [myst](https://myst-nb.readthedocs.io/en/latest/) + [jupytext]

(https://jupytext.readthedocs.io/en/latest/install.html) ---

Notebooks and Documentation

I do not like Jupyter Notebooks. This is because, to me, a rabid `org-mode` fanatic [^fn:1], it seems pretty odd that they are essentially monstrous `json` nightmares instead of plain text. Thankfully jupytext and MyST-NB solve this problem admirably. There are still some use-cases for having pure notebooks, especially since GitHub now renders them; so a CI (Github Actions) generates a “notebooks” branch for the entrenched as well.

Cell-Tags and Disappointment

One of the major disappointments faced during the project had to do with the way cell tags work; or rather, do not work. It was planned to have notebooks executed with `papermill` and switch between development and stable packages based on cell-tags; however, cell-tags apparently cannot be used for meta-injection; which means they are absolutely useless. A workaround of course might be using macro expansions; however, `papermill` did not really seem up to the challenge of injecting non-python variables either so this was abandoned.

Future Plans

Personally, I will not be participating in further rounds in the foreseeable future, mostly because I intend to continue working on SymEngine until it is compliant with the standards I championed, and the addition of more projects to the portfolio I consider to be part of my moral responsibilities is not a good idea right now.

Conclusions

It is difficult to gauge the effect of financial incentives on the quality of code. That analysis and possible rant I’ll save for another post. Projects like SymEngine attract very good proposals during high incentive development bursts like the Google Summer of Code and Season of Documentation; but these contributions tend to rot over time; which makes them of dubious use. That said, my own experience involving SymEngine was personally enriching, and I look forward to working on the project further. I believe

the most useful aspect of the program is the ability to meet with the rest of the development team regularly, none of the other projects I work on have regular meetings, and it is a model I intend to carry forward in some of my projects.

References

Cohen, Joel S. 2002. *Computer Algebra and Symbolic Computation: Elementary Algorithms*. A K Peters/CRC Press. <https://doi.org/10.1201/9781439863695>.
---------. 2003. *Computer Algebra and Symbolic Computation: Mathematical Methods*. A K Peters/CRC Press. <https://doi.org/10.1201/9781439863701>.

```
org-mode](https://github.com/HaoZeke/haozeke.github.io/), my
`doom-emacs` configuration is also [extensive and
online](https://dotdoom.rgoswami.me)
```
