

On the mixing of data and code

AUTHOR: Steven Robertson
CONTACT: steven@strobe.cc
TAGS: Article

ABSTRACT

No abstract yet.

This article is less published than most. Don't judge it / me based on it just yet.

- There's something beautiful about good code.
 - Good APIs or paradigms, but generally true also
 - The structure of good code is easier to model, both in our heads and on paper
 - Clean code is also easier to maintain; fewer links means fewer points of failure, makes it easier to deal with unit tests.
 - This is obvious to most experienced programmers.
- This notion came to mind a few months ago, after the [Climate Research Unit data leak](#). Much of the controversy
 - The code references external data sets liberally, and includes undocumented constants that relate closely to the missing data.
 - First thought was, "this is ugly."
- With a bit of horror, I realized that I was also thinking "this is *familiar*."
 - I've coded like this. What is wrong with me?

PRODUCT AWARENESS

- Writing good code takes an intentional effort. Even if you have knowledge and experience, the planning (or, for agile folks, refactoring) involved in keeping code clean is non-zero. But the benefits - of

increased generality, ease of maintenance, etc. - far outweigh the cost.

- I have a lot left to learn and do, but I know better. Yet I still produce absolute garbage, such as the code on the CUDA atomics benchmarks page. It would be simple to chalk it up to laziness or ineptitude, but it's not just me and the CRU here: [wavelet toolbox], [contourlet paper], [recent example of rendering].
- Clearly there's something systemic going on: some difference in the fundamental task at hand that results in this particular kind of crap coming out instead of quality programming. It seems too broad to simply be inexperience, so what's up?
- In my own work, the switch turned out to be the *product*. If the end result of my effort is a piece of software, like Quod Libet, I focus on clean, well-structured code¹. If, on the other hand, the goal is a *report* — fora scientific paper, a school assignment, or a blog post — I almost invariably produced this kind of procedural, copy-pasted POS.
- Here's the thing, though: those code snippets worked.

THE REAL PROBLEM

- After realizing how systemic this kind of coding was in my own works, I tried writing the simulation script for my next lab report as if I was going to release it: keeping the data and configuration in separate files, factoring program components into nice little objects, writing docstrings.
- The result was just as long as my other approaches, had just as many bugs to work out, but took significantly longer to create. And I have experience with this kind of thing.
- More frustrating was that the now-isolated code was now *less* portable than the copy-pasted mess I was using before. After a few rounds of hasty enhancements made in order to accommodate the requirements of subsequent labs, the script wouldn't accept the data from the old labs anymore.
- In this case, copying and pasting procedural code with data baked in created a system that was more reproduceable. This single anecdote is hardly scientific evidence, but it does raise the possibility that, rather than being unilaterally evil, the practice of mixing code and data might actually present some advantages in particular situations (particularly those in which the product to be distributed is not the code itself, but the output of running the code on the data).
- But if the problem that caused CRU's crisis isn't related to poor coding practices,

shouting "Write better code!" at researchers without the time or experience to do so may not be the best approach.
- This is hardly a scientific result,

- - I certainly plan to

¹I make no claims as to whether or not I achieve this goal, but the effort is clearly there.

```
SYSTEM-MESSAGE
```

```
WARNING/2 in <string>, line 108
```

```
Bullet list ends without a blank line; unexpected unindent. backrefs:
```

use good programming practice for my upcoming Codec Game project (more details in later post) - but by the time many scientists are working in a field, trained in MatLab's ugly procedural sin, the effort of unlearning this is too great.

- In any case, there are enough examples in the field to conclude that ugly coding practices are at least popular, and may in fact be better for one-shot scripting. While it is admittedly evil in a software product, perhaps the tight binding of code to a particular dataset is the way to go for scientific research. Perhaps ugly coding is not the problem.
- But clearly there is a problem. The CRU data leak was a media circus, and resulted in charges. Either mistakes were made, or deceit was practiced; in either case, the results should never have made it past peer review.
- The problem, IMO [and others' O], is reproducibility. A published scientific paper is just as vulnerable to spin as a press release containing benchmarks; in both cases, what we really need to make informed judgments is the raw data, and the process used to collect and process it.
- Scientists won't make pretty web apps to go along with every paper they write; the traditional approach used in the software industry — get your programmers to write better code — is unrealistic for most researchers. What's a *practical* way to improve reproducibility?

HISTORY OF SCIENCE

•

(unfinished article)