Hacks are not Hacks: A New Analytical Model for the Role of Code work in Scientific Practice

|  |  |  |
| --- | --- | --- |
| 1st Author Name  Affiliation  City, Country  e-mail address | 2nd Author Name  Affiliation  City, Country  e-mail address | 3rd Author Name  Affiliation  City, Country  e-mail address |

# ABSTRACT

Paste the appropriate copyright/license statement here. ACM now supports three different publication options:

* ACM copyright: ACM holds the copyright on the work. This is the historical approach.
* License: The author(s) retain copyright, but ACM receives an exclusive publication license.
* Open Access: The author(s) wish to pay for the work to be open access. The additional fee must be paid to ACM.

This text field is large enough to hold the appropriate release statement assuming it is single-spaced in Times New Roman 8-point font. Please do not change or modify the size of this text box.

Each submission will be assigned a DOI string to be included here.

TBD

## Author Keywords

TBD

## ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous; See<http://acm.org/about/class/1998> for the full list of ACM classifiers. This section is required.

# INTRODUCTION

Code in science matters. Now is a remarkable moment. It is positioned in a history of increased abundance of more varied and high-resolution data and more robust computational capabilities.

In this work, we distinguish programming that MUST work [a la Kelly] from programming that can exist in an experimental, beta flux. Kelly makes this distinction articulated as .....

We conducted a qualitative 18-month study with 46 scientists in four oceanography teams, and up to 100 in various computational skill workshops attended or observed. After data collection, the first author presented some of the findings to the scientists. In this talk, I presented the claim that the ``best practices'' that they occasionally worried about not pursuing with enough diligence (``\_should\_ be using gitHub, but haven't gotten around to it....'') were more formal versions of practices already in place, but which extended beyond code to data and figures. For example, through formal software engineering testing was virtually nonexistent in the group studied, all groups engaged in a thorough, systematic validation and verification processes using consistent visual artifacts. We deveop this argument further in Section XXX, but the other sections (XX and XX) also take the stance that code work must not be critiqued in isolation of the data and visual products that it requires and enables.

The first author presented a preliminary articulation of these ideas to the interested study particupants, and one of the very last questions was: ``so you coverd in your talk some ways in which oceanographers can learned from software engineers and computer scientists, but what can computer scientists learn from oceanographers?'' We believe this is a question of utmost importance in the study of scientific programming practices by computer scientists, and this manuscript offers our answer so far.

# BACKGROUND

The imbrication of social and technological as seen in Leonardi, Orlikowski.

The way scientific work practices incorporate material and technical arrangements so that the practice of making a visualization is the science, not merely representing the science (Vertesi).

Borgman, Edwards from the dissertation

Steinhart and Jackson 2015 CSCW

Paul Edwards Vast Machine 2010 MITP quote data is made, not collected like apples or clams

# The MODEL

During dissertation research, the first author developed a model of deliberate change in code work based on ethnographic study of four oceanography teams in the Pacific Northwest. Oceanography constituted a particularly fruitful study context because of its interdisciplinarity.

> -Perfect world- the intersection of multiple perfect worlds and the stark distinction between what scientific software advisors think they should be doing and what they do in practice.

+ the properties of the perfect world reflect the values of understandability/extensibility/usability as they pertain to various imagined/anticipated audiences

> - Working environment – draws established ideas to show that the material, cognitive, and social resources of a project can shape the questions that can be posed.

+ "main claim" there IS a difference between code work for software that MUST work (infrastructural) and softwae that is experimental, in development, in "permanent beta" whatever. BUT there IS NOT a distinct "hack made by an indiv scientist, which is ad-hoc idiosyncratic etc" and other kinds of experimental software, without and beyond scientific programming domains.

FIGURE 1 goes here illustrating the model.

This model is agnostic relative to the specific tool. Indeed, it is very liberal with regard to what constitutes a tool.

# Three arguments that “hacks” are not hacks

When I say that the things that are called ``hacks'' are not hacks, I am claiming that they do not possess the qualities that make them seem like hacks: (1) contained to an individual who does not need to include the requirements of others; (2) a ``means to an ends'' meaning that there is no sense of elegance or intrinsic value of code driving it; and (3) ad-hoc, ephemeral.

## Code that would not convince peers does not convince the individual

## Beauty is in the Eye of the Beholder

``simply isn't interesting to most computer scientists.'' [B]

("students will now believe that it's worth learning how to test software systematically") [A] Well OF COURSE they believe that! testing and reproducibility are never not on the horizon of the perfect world. But the journey there takes a long time, and is filled with uncertainty. The Incentive is missing for investing in learning any particular thing out of a cornucopia of viable and enticing possibilities.

For all the handwringing about not testing, in a retrospective of learnings from years of SWC workshops, Wilcon writes that the workshops ``don't do as good a job as we would like teaching testing. mechanics of unit testing are straightforward ... and it's easy to come up with representative test cases for things like formatting data files, but what should we tell scinetists about testing the numerical parts of their applications? Once we've covered floating-point roundoff and the need to use `almost equal' instead of `exactly equal', our learners quite reasonable ask, `What would I use as a tolerance for my computaiton?' for which no one has a good answer.'' [B]

(INCLUDE FIGURE 6.2)

In this section I make the argument

``The reason most computational scientists don't care about quality is that there's no incentive for them to do so''[A Greg Wilson] because of publication stuff. We see Howison and Herbsleb making similar incentive arguments that problematize certain software production systems. Young et al and Mislan et al both appeal to ways in which the sharing of data and code respectively can be made more streamlined and accessible, to support the overall value of sharing.

## Moments of Flux / No going Back

The model we introduce in Section XX defines the working environment as subject to deliberate change. There are many options for the direction of this change in any of the myriad small opportunities for decision making. The direction is decided using an imagination of the perfect world.

Another way to orient change is by appealing to a software advisor. WHAT IS AN SA.

For example, one of the team's SAs says this: “You know, I just watch them, I can see when their eyes glaze over. The Principal Investigator (PI) is really good at pushing me on it: `do I - as a scientist - really need to know this?’ '' An uncharitable reading of this statement seems to reinforce the idea that scientists do not care about the quality of their code, and this dooms them to a life of ad-hoc hacks. However, this was spoken in the context of a group that at the beginning of the study had recently begun adopting version control with some associated growing pains, and by the end had integrated GitHub substantially into collaborative work practices. Furthermore, it was a post-doc and a graduate student who then took the initiative to ask the SA about automating some of the testing in their analytic pipeline, which he was more than delighted to begin explaining. A ``hack'' is typically described as ``not the best way of doing things,'' ``not very elegant,'' ``ugly but it works,'' or failing to use something it ``really should be.'' Instead of interpreting these terms at their face judgmental value, we can observe the progression of the ``ugliness'' over time and see that if there is truly a ``perfect world'' that the scientists prioritize, individual small moments of flux will slowly inch those ``hacks'' in that direction.

# implications for practice

Of the model:

Reframe dismay or surprise as comparison of working environment to the perfect world. Is the imagination of the perfect world is shared enough to justify deliberate change?

Necessary for follow-through on change:  
[ ] awareness - I have heard of it & know what it is  
[ ] intention - I should probably be using it…  
[ ] momentum - If I already knew it, I would use it  
[ ] opportunity - Clear course of action is available

Evaluate not only technical, but social and skills components of the working environment. Never “back at square one!”

# REFERENCES

1. @\_CHINOSAUR. 2014. VENUE IS TOO COLD. #BINGO #CHI2016. Tweet. (1 May, 2014). Retrieved February 2, 2014 from https://twitter.com/\_CHINOSAUR/status/461864317415989248
2. ACM. How to Classify Works Using ACM’s Computing Classification System. 2014. Retrieved August 22, 2014 from [http://www.acm.org/class/how\_to\_use.html](http://www.acm.org/class/how_to_use.html%2520)
3. Ronald E. Anderson. 1992. Social impacts of computing: Codes of professional ethics. *Soc Sci Comput Rev* 10, 2: 453-469.
4. Anna Cavender, Shari Trewin, Vicki Hanson. 2014. Accessible Writing Guide. Retrieved August 22, 2014 from <http://www.sigaccess.org/welcome-to-sigaccess/resources/accessible-writing-guide/>
5. Morton L. Heilig. 1962. Sensorama Simulator, U.S. Patent 3,050,870, Filed January 10, 1961, issued August 28, 1962.
6. Jofish Kaye and Paul Dourish. 2014. Special issue on science fiction and ubiquitous computing. *Personal Ubiquitous Comput*. 18, 4 (April 2014), 765-766. <http://dx.doi.org/10.1007/s00779-014-0773-4>
7. Scott R. Klemmer, Michael Thomsen, Ethan Phelps-Goodman, Robert Lee, and James A. Landay. 2002. Where do web sites come from?: capturing and interacting with design history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02), 1-8. <http://doi.acm.org/10.1145/503376.503378>
8. Psy. 2012. Gangnam Style. Video. (15 July 2012.). Retrieved August 22, 2014 from <https://www.youtube.com/watch?v=9bZkp7q19f0>
9. Marilyn Schwartz. 1995. *Guidelines for Bias-Free Writing.* Indiana University Press.
10. Ivan E. Sutherland. 1963. *Sketchpad, a Man-Machine Graphical Communication System*. Ph.D Dissertation. Massachusetts Institute of Technology, Cambridge, MA.
11. Langdon Winner. 1999. Do artifacts have politics? In *The Social Shaping of Technology* (2nd. ed.), Donald MacKenzie and Judy Wajcman (eds.). Open University Press, Buckingham, UK, 28-40.