Aim for the eagle: Making the best use of our software research skills to fight climate change

Jon Pipitone, Jorge Aranda
Department of Computer Science
University of Toronto
40 St. George St.
Toronto, Canada
(jp, jaranda)@cs.toronto.edu

Valeria Cortés
Office of Student Life
University of Toronto
21 Sussex Ave.
Toronto, Canada
val.cortes@utoronto.ca

***Abstract*—Climate change poses great challenges to the viability of our civilization, and if we are to overcome these challenges we need to bring the best of our abilities to the table. In this paper we propose a set of guidelines for software researchers concerned about doing the right thing for the planet.**

"Aim for the eagle, you may bag the pheasant, but you won't eat crow." —*Anonymous*

I. Introduction

Climate change is the greatest threat currently facing humanity. As the call for participation to this workshop reminds us [3], unless urgent and drastic cuts are made to our greenhouse gas emissions, global warming is likely to trigger a number of tipping points with catastrophic consequences to our society and the planet's ecosystems. Therefore, the problem of climate change requires immediate action from everyone, including ourselves. All of us should bring the best of our abilities to the table.

Software researchers, in particular, have a set of skills that are much needed to address the problem. We are experts at abstraction, at computational thinking, and at modeling complex systems, among other things, and the software that we build and study runs the modern world. The sooner we take action to apply our knowledge and ability to the service of humankind, the better our chances to mitigate climate change.

In our case, we must acknowledge that our research is our best and only tool to make a difference. With such an urgent threat at hand, we may feel compelled to act outside of our familiar capacities as researchers: to engage in political action, or even to consider leaving academia itself in order to attack the problem directly. This would be a mistake. We believe that we must put all our efforts into our research, where they will be of most use, and disregard other avenues as distractions. In this position paper we discuss our ideas on how to be most effective in selecting and conducting our research.

II. Guidelines

Applying software research to a different domain may appear to be a daunting prospect, but it actually does not require us to change our research approaches, as the same general problems occur in every field of study. The only important consideration is to identify our intellectual assets, and intelligently choose research projects to maximize our potential for impact. In the following sections we propose guidelines and recommendations that clarify how best to undertake the switch to software research on climate change issues.

A. Scope

Think really big. Avoid projects targeted at specific stakeholders and that may have only an immediate impact. Instead, select projects which have the greatest potential applicability and highest potential future returns, even if the immediate benefits are unclear. The greatest strides in science are taken by such pursuits. For example, Babbage's Analytical Engine was never built, but decades later its design ideas had a groundbreaking impact on the architecture of modern computers. Similarly, Leonardo's inventions (his helicopters, submarines, etc.) are, centuries later, recognized by their prescience and their influence in modern life. Amazingly, some of his inventions have recently been built and shown to work – a marvel considering he only ever produced sketches of his machines. Our point is that only by choosing to do such ambitious, blue-skies research can we hope to have as large an impact as our eminent predecessors. Most importantly, it is only this kind of research which will lead to solutions grand enough to topple the climate change goliath.

In practice, this means we need to apply our community's key intellectual assets, abstraction and generalisation, when crafting our research agendas. As software researchers, we have expertise in abstract thinking as well as the design, management, and refactoring of complex information systems and generic frameworks. With these assets we are well positioned in the fight against climate change because at some level of description *everything* is a system – be it an information system, networks of computers, the social networks we are all part of (our families and circles of friends), our bodies, or even the fragile and complex web of interactions that make up our biosphere. Software is simply an *instance* of a domain in the much broader domain of *systems* to which our expertise applies. We are systems researchers!

As such, to make a contribution we need only apply the techniques we have for software systems to any other system. A few examples illustrate this point.

1) The use of design patterns for policy-making: Modern law- and policy-making produces such complex and massive tomes of interconnected pieces that it is impossible for anybody to fully understand them and their implications. Fortunately, software researchers have been there before. We can use our knowledge of software design patterns and apply it to the design of laws and policies for the betterment of society. Imagine a tool, not unlike existing software IDEs, for the development of legislation. It would feature refactoring tools for extracting common law elements, resolving conflicts between existing and proposed legislation, or modifying law constructs for future extensibility.

Applying design patterns to law will also enable policymakers to use a more advanced vocabulary to discuss their work: we can easily picture them talking about using, for instance, the Decorator or the Facade patterns in a particularly complicated piece of legislation, or of applying Abstract Factories for the creation of local laws that also adhere to a global framework. Using design patterns in this context would make the creation of climate laws a snap, leading to the speedy signing of urgently needed policies for carbon regulation.

2) The use of model-driven development for climate simulations: The climate modeling community still seems trapped in 20th century technology with its use of legacy programming languages and rudimentary design tools. We know from recent studies [4] that much of the scientists' time is spent running arcane verification and validation tests simply so to prove to themselves that their sloppy code does what they intended it to! We can apply our cutting edge model-driven technology to speed up their development processes and ensure correctness. Our modeling languages excel at capturing the essence of complex models such as those used in climate research. Climatologists could create their climate models in a customized version of UML, and with the push of a button the underlying computational code would be automatically generated and optimized for their hardware settings. Scientists could explore their research questions much more quickly and, as an added benefit, their automatized computer models would become free of suspicion from climate change deniers.

3) The use of fault prediction mechanisms for "rewiring" our understanding of climate change: As humans, we carry several well-documented cognitive biases. In particular, we are notoriously bad at coming to grips with the dynamics of climate change: for instance, we find it difficult to understand that cutting our emissions now will not stop global warming for a few more decades, or that climate and weather are two very different concepts and a snowstorm is no indication that climate change has stopped.

This points to inherent "faults" in the design of our minds — and as software researchers we know much about predicting and detecting the localization of such faults. We can look at the brain as a very intricate hardware/software system, with connections between neurons acting as the system's call graph and their synapses as its execution paths. Therefore, with more and more detailed CT scans, we will be able to construct a map of the brain, and then use our already mature fault prediction mechanisms (based on cohesion, coupling, method size, and other such metrics) to point to the areas of our brains in most need of intervention. Based on our findings, neurologists could eventually "rewire" our brains, leading to an improvement in our understanding of climate change dynamics and, therefore, to immediate action to tackle this challenge.

We think that the beauty of these proposals lies in the fact that we, as software researchers, can make use of our intellectual expertise by applying our ideas to domains we know little about, and still have a, possibly, groundbreaking impact in the near future. Since all systems are essentially equivalent, our insights from the software realm are immediately transferable to other domains.[1]

B. Stakeholders

In order to use our abstraction expertise successfully, we must be careful to maintain an appropriate distance from the specific needs of individual stakeholders or situations we are researching. If you have to work for a particular user (perhaps because of requirements of your research funding), we advise that you do not waste your time trying to figure out their specific challenges: this will only bog you down

[1]Although, of course, some implementation details would still need to be worked out.

with minutia and will hurt the world-wide applicability of your ideas.

Consider that most of the critical issues blocking action on climate change have obvious software solutions, such as the need for generic tools to manipulate and process data in order to aid decision making and communication, to name an example out of many. Additionally, software is everywhere; it influences and mediates every decision we make, from our choice of breakfast cereal in the morning to which gas station we choose to fill up at. Therefore, since we are experts at tool making, we should focus our energy on making tools we know are appropriate for the general task, and reach out to stakeholders only to fine-tune or validate our work.

Similarly, avoid falling in the trap of working for groups with political motivations, such as local environmentalist organisations or political parties. This may sound harsh. No one can deny that they are full of perfectly well-intentioned people, but their day-to-day involvement with these issues means that they often can't see the forest for the trees, and you may find yourself developing a similar shortsightedness if you collaborate with them extensively. Besides, these groups tend to include politically biased individuals, and we, as scientists, need to maintain our objectivity and our independence from political agendas as much as possible: being at the grassroots level means getting your hands dirty. As well, to put it frankly, there are very few funding opportunities in such involvements.

Furthermore, it is not even clear what productive involvement with such groups would look like. Research strategies like Action Research [1], [2] and Participatory Design [5] could perhaps give them some immediate benefit, and they could give you a few interesting insights to report to our community, but these insights are a long way from having the potentially groundbreaking impact of the ambitious, universal projects we endorse.

Finally, collaboration with undergraduate student projects and with university staff initiatives should ideally be kept at a minimum. Undergraduate students may be passionate about this cause, but this passion may be overwhelming and lead to involvement with activist groups, which we've shown to be in detriment of our work. As for university staff, beyond good will all they can offer is contacts in the community and programs to develop soft skills such as teamwork and conflict resolution [6] —perhaps valuable to your inner development, but not relevant to the cause of fighting climate change.

C. Applicability

As researchers, we work purely in the realm of ideas. We achieve our greatest impact when we focus solely on the creation and dissemination of novel ideas, and let people with other skills (and with incentive structures that reward nitty-gritty work!) worry about implementation details and detailed field trials. We suggest that the ideal work dynamic is one where researchers come up with an idea, explore it, prepare a prototype or proof-of-concept if they must, publish, and move on to the next success.

This suggestion goes hand-in-hand with our recommendations on scope: here we are suggesting that once you have planted the seeds for a revolution you should move on. We need to come to terms with the fact that technology transfer usually takes about 25 years. If we stick around that long with a single project, we won't be successful researchers –and we need that success to get more grants, tackle greater and greater research questions, and live up to the huge demands that climate change poses to every one of us.

III. Conclusions

Abstraction, detachment from policy making and activism, and academic aplomb in the face of poor prospects for the applicability of our ideas are the best assets we can use as software researchers to fight climate change, this most urgent threat to the viability of our civilization.

Hopefully it is clear by now what we are up to in this position paper. We believe climate change is a serious issue, and we believe that software researchers have the opportunity to be of benefit, but only as long as we focus on tractable projects grounded with plenty of stakeholder involvement. To continue this discussion, please visit: http://groups.google.com/group/se-for-the-planet

We would like to acknowledge the helpful discussion and feedback from Jono Lung, Joseph Pipitone, Neil Ernst, and Steve Easterbrook.

References

[1] D. Avison, F. Lau, M. Myers, and P. A. Nielsen. Action research. *Communications of the ACM*, 42(1):94–97, 1999.

[2] D. V. Craig. *Action Research Essentials*. John Wiley and Sons, 2009.

[3] S. Easterbrook, K. Mens, and S. Zschaler. 2nd workshop on software research and climate change; http://www.cs.toronto.edu/wsrcc/index.html, 2010.

[4] S. M. Easterbrook and T. C. Johns. Engineering the software for understanding climate change. *IEEE Computing in Science and Engineering*, 11(6):65–74, 2009.

[5] F. Kensing and J. Blomberg. Participatory design: Issues and concerns. *Computer Supported Cooperative Work*, 7(3-4):167–185, 1998.

[6] S. R. Komives and W. Wagner. *Leadership for a Better World*. Jossey-Bass, 2009.