How to Build a Research System in Your Spare Time

Ratul Mahajan
Microsoft Research

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The author takes full responsibility for this article's content. Comments can be posted through CCR Online.

Abstract– This paper is based on a talk that I gave at CoNEXT 2009. Inspired by Hal Varian's paper on building economic models,[1] it describes a research method for building computer systems. I find this method useful in my work and hope that some readers will find it helpful as well.

Categories and Subject Descriptors
C.2.m [Miscellaneous]

General Terms
Design, Experimentation

Keywords
Research method, computer systems

1. INTRODUCTION

Last year, I had the (mis)fortune of serving on several conference and workshop program committees, and I noticed that a common set of complaints keep cropping up for papers that describe research systems, such as:

- Do we need another paper on ....?
- Is this problem important?
- Does this solution work?
- What is new here?
- Why not solve the problem this other (simpler) way?

The authors of many of these papers had clearly put in a lot of work, and the papers usually contained some good ideas. As an aside, my papers too are not immune to these criticisms.

These complaints made me wonder if their root cause is poor communication or poor research process. That is, is it the case that we as authors are not communicating our results well or do the complaints represent a flaw in the research process itself?

This question is hard to answer in general. As authors, we often feel that concerns like the above can be addressed by better writing because "the reviewers did not get it." While good writing should always be a goal, ultimately writing only reflects thinking and experiences that are the results of the research process. Thus, the research process appears to be at least a major contributory factor.

So then, what should an ideal research process be and can it be articulated? A research process is not merely a set of dos and don'ts. It is a more systematic, step-by-step description of the research activity. The sequence is important because the initial steps can save you from making poor choices and pursuing less promising avenues, which you may be forced to paper over later with the help of writing.

Naively, I decided to present in this paper a method for building research systems. This method is based on what I have learned from my collaborators and colleagues as well as my past mistakes. I focus on building systems because much of my work falls in that category, but I expect that some of what follows applies to other types of networking research as well.

Note that adhering to a research method does not mean that there is no role for intuition, creativity, and hard work. On the contrary, these factors are absolutely necessary for success. The role of the process is to help focus, avoid common mistakes, and proceed with due haste. Think of it as best practices.

I do not intend to build consensus on the "right" way to do network systems research. My hope is simply that some readers will find the method below useful in their work, as I do in mine. There are undoubtedly other productive methods. I would love to hear from other researchers about important aspects that I have missed and aspects with which they disagree. Better yet, I invite them to articulate their method, in an editorial such as this one or elsewhere.

2. BUILDING A RESEARCH SYSTEM

In my work, I have found the following method to work well:

1. Pick the domain carefully The first step is to pick a domain or an area that you want to investigate. You may not have to go through this step if you already have one in mind. But I tend to switch domains on a regular basis because I run out of ideas. Thus, for me, this step is a conscious exercise.

2. Know the problem well before you start building By the time you pick a domain, you should have an inkling of what you want to do. But before rushing to design and build the system, identify a technical problem, solving which would represent a significant improvement in the state of the world. For instance, it would reduce cost, improve performance, or enable new functionality. If you already know the problem, the purpose of this step is to confirm that the problem is real and important. Without this step, you run the risk of solving a non-problem.

3. Debate several solution ideas and have a core idea behind what you build Suppose you know the problem that you want to solve, it is still not time to start building the system. First consider

[1] "How to Build an Economic Model in Your Spare Time," *Passion and Craft: Economists at Work,* Univ. of Michigan Press, 1997.

and debate several solution ideas to gain clarity on the design space and identify the core idea that will underlie your system. It is important to articulate this idea clearly and concisely because research systems are intended to validate a hypothesis such as this idea can solve this problem. If you build a system without articulating the central idea, it has little lasting research value.

4. When building, start small and then embellish It is now time to start building the system. It helps to start small by implementing your core idea and add complexity only as needed. This incremental approach ensures that no unnecessary complexity is added and it is easier to adapt if needed. When evaluating the system, in addition to showing that it works, show why it works and justify its complexities and simplifying assumptions.

5. Make it real Finally, make your research go beyond the paper. This step will often require you to step outside systems research or even research, but it is critical for your work to have an impact.

By necessity, the steps above are a simplification of reality, but they do capture the essence. For maximum effectiveness, the steps must be executed in the order specified, though there can be some overlap between them and some back-and-forth. Skipping a step altogether reduces research quality. The steps that authors tend to forget are 2, 3, and 5.

The later steps usually take more time, but the earlier steps are equally – if not more – important because the success of the later ones hinges on them. Steps 2 and 3 are not necessarily pure thought experiments, and they often involve an experimental component.

The rest of this paper describes these steps in detail. I will draw on my experiences and use anecdotes from my work, not because it is exemplary, but because I was there when it was done.

3. PICKING A DOMAIN

When picking a domain, the most important criterion is that you find it fascinating. If inter-domain routing is not your cup of tea, do not pick that research area (even if you are a graduate student whose adviser's tenure depends on it).

But the question is: what beyond that criterion? In trying to find a domain, I am wary of hot trends. Currently, these trends appear to be social networks and data centers. It is not that hot areas are unimportant. But if enough smart people are already working in an area, unless you have a novel insight or perspective before diving in, your time and energy is probably best spent elsewhere. If you are a graduate student, another downside to picking a hot area is that you will graduate with several others who have worked in the same area, which will make it harder for you to stand out. This happened to me, and in retrospect, my choice of thesis topic was not that inspired.

A more promising strategy is to observe the world for big changes. The fault lines created by such changes represent promising avenues for research on accommodating those changes. These changes could be in workloads, technology trends, or even new concerns such as energy consumption. Some of the very successful research projects have been driven by such observations. For example, two of the three award papers at the SOSP 2009 conference fall in this category. As another example, my recent work on vehicular networks is driven by the observation of increasing demand for connectivity from moving vehicles. I wanted to understand how to enable that connectivity in a cheap and reliable manner.

Another kind of change to look for is adoption or availability of new technologies. For instance, in the wireless domain, the availability of cheap software-defined radios led to significant, exciting research. A similar phenomenon is occurring today with MIMO (e.g., 802.11n) and programmable directional antennae.

Yet another kind of change, which researchers tend to overlook, is change in government regulation. Current examples of expected regulatory changes include net neutrality, privacy, and white spaces.

A second strategy for picking domains is to prefer those that are underexplored. A domain can be underexplored because it is not on other researchers' radars, because folks have not figured out how to do systems research in it, or because people presume that the problems in that domain have been solved by solutions in related domains. It is worthwhile to question such premises.

One of my recent projects focused on diagnosing faults in small enterprise networks. My collaborators and I observed that these networks had not been studied before, likely because of the presumption that solutions designed for large enterprises also work for small enterprises. On a closer look, we found that presumption to be false. Studying small enterprise networks gave us a very different perspective on the design of diagnostic systems, a perspective that we found later was useful for large enterprises as well.

A third strategy for picking domains is to look for unique opportunities. This opportunity could be data that yields novel insights into the workings a real system. Alternatively, if you encounter a new technique or tool, ask for what else it might be useful. This question also applies to tools that you develop because they could be useful in a different context. The starting point for Rocketfuel, an ISP topology discovery system that I built with Neil Spring, was a tool that I had built to understand routing misconfiguration.

4. IDENTIFYING THE PROBLEM

Picking a domain does not mean that you have also identified a technical problem to solve. But you should have at least some notion of the problem. If all you know is that you want to work on data centers, you did not do a good job of picking a domain. Go back to the previous step.

Before solving what you think is a potential problem, frame it more concretely. Problem framing involves identifying the exact weakness in the status quo that you want to address and estimating the benefits of addressing that weakness. If you want to work on scaling data centers, you must first understand the scalability bottlenecks, the benefits of scaling, and any characteristics that you can leverage to scale.

Framing the problem is an exercise that you must do for yourself. Papers by other researchers are not a good source. If a paper describes a problem in detail but does not solve it, chances are that the problem is not important or very hard. If it is very hard, you need insights and perspectives that cannot be obtained only by reading that paper.

Instead, you must "scrutinize" carefully using measurements, data analysis, surveys, etc. Your goal is to establish a concrete understanding of the real issues. Use your imagination to guide what to scrutinize and how. Never let imagination alone or hearsay frame the problem for you. I take this process seriously because I have been surprised many times. Issues that I think are problems turn out to non-problems after careful scrutiny.

For instance, for my work on vehicular Internet access using WiFi, I presumed that current WiFi handoff methods, in which clients talk to only one AP at a time, lose a lot more packets compared to handoff methods that use multiple APs simultaneously. But measurements showed that the difference was less than 20%, and thus, that particular problem was not compelling. Further analysis, however, showed that the real problem with using only one AP was something different – poorer, by roughly a factor of seven, support for interactive traffic. I was glad that I did my own measurements before going off to solve the wrong problem.

4.1 Some scrutiny how-tos

I express below a few thoughts on how to scrutinize. First, for many hard problems, scrutiny is not straightforward, and thus you should be open to unconventional approaches. These include, but are not limited to, building your own testbed, reading market research reports, social engineering, and back-of-the-envelope calculations. In my work on network diagnosis, I read hundreds of trouble tickets to understand real problems that operators face. In an earlier work, against the good advice of my advisers, I spammed network operators to be able to separate intentional configuration changes from misconfigurations. These approaches succeeded where conventional methods failed.

Second, conduct your own scrutiny as far as possible, even if it is significant effort and even if it requires repeating what others have done before. There is no substitute. By doing it yourself, you may find something new because of luck or because your methods are different. The misconception that I had about using one versus multiple APs was based on reading other papers. Even if you do not discover anything new, you will get to know the problem at a visceral, intuitive level that will help you design a better solution.

Third, use as realistic a setting as possible when you scrutinize. Drawing again on my vehicular WiFi work, because my testbed was more realistic, I found wireless connectivity to be quite different than earlier measurements. Earlier work found that the connectivity of a client to an AP could be cleanly divided into three phases, entry and exit phases with poor connectivity and a production phase with good connectivity. But I found that the connectivity was complex and such separable phases did not exist.

Finally, project the improvement opportunities that you find into the future and weigh against expected technology trends. If a problem will be less pressing in a few years – because of Moore's law, for instance – it is probably not worth solving. After all, it will be at least a few years before your solution can be adopted.

Similarly, if the projected gain is small, the problem is not worth solving. There is always an overhead to deploying new solutions, and real (non-research) instantiations of a solution typically provide less than anticipated gain. How much gain is worthwhile depends on the what aspect you are improving. If you are improving performance, 10-20% gain is rarely worth the effort. But 10-20% reduction in cost can be significant.

4.2 Benefits of well-done scrutiny

The primary benefit of well-done scrutiny is identification of real issues and opportunities. By doing the scrutiny yourself, you would have obtained detailed insights into the domain and the problem. You would have likely also uncovered important aspects of the domain that can help design the solution. For instance, if you find locality in Web requests, caching becomes a possibility; or, if you have elephants and mice in traffic, managing elephants alone might suffice for traffic engineering. Such insights are almost impossible to obtain without good scrutiny, and they contribute greatly towards the success of the design exercise.

There are secondary benefits as well. One is that this exercise tells you how to evaluate your design. Your design should have solved the issues that you just identified. Another secondary benefit is that careful scrutiny can reduce risk with respect to publishing. Including the results of your scrutiny makes the paper stronger and the problem more convincing for the readers. Further, if you decide to abandon that line of work because you did not find a compelling problem, you may publish your scrutiny results independently.

5. SELECTING SOLUTION IDEAS

At this point, you know well the problem that you want to solve. You may even have an idea or two about how to solve it, and you may be eager to start building the system. But it is too early to start building. Instead, you should first take a problem-centric view and consider as many solution ideas as possible, including those that are not yours. This view will protect you from being blind sided by your own brilliant idea and ignore other, perhaps even better, ideas. Discovering later that other ideas are better is much costlier. Your idea may still be valuable, but perhaps not as good for the problem at hand.

To select a robust solution idea, first fish for possible ideas and then conduct a triage to identify the most promising one.

5.1 Fishing for solution ideas

To compile a list of possible solution ideas, filter the problem to its crux by removing any non-essential detail, which will enable you to focus on the core of the problem. These extraneous details may be related to, for instance, how exactly a solution can be implemented. There will be time to figure that out later. One, but not the only, way to abstract the problem is to build simple models. A good goal is to abstract to a level of a puzzle that you can describe to people outside of your research area. Once you have this puzzle in hand, look at it from different perspectives. What if you did this? What if that happened? What if you modified it thus? What if this constraint was missing or was introduced?

Another useful activity, once you have the abstract problem, is to make connections to similar problems. Where else such a problem might arise? Do not restrict yourself to systems literature, or even CS literature, or even academic literature. For instance, the solution idea for a win-win routing protocol that I designed for competitive yet cooperative ISPs, stemmed from a popular magazine. The magazine described a method to divide jointly-owned property among a couple after a divorce. The division method had the nice property that it incented each person to honestly reveal his or her relative valuation for individual objects. This property was what I wanted in my solution. In another recent work of mine, techniques from removing sampling biases in surveys of illegal drug users and MRI machines proved useful.

When trying to make connections, having a vivid imagination helps but that is a gift over which we have little control, at least in the short term. I am not very imaginative, and so I leverage those

around me. I describe my problem to others, including people outside my area, which is possible if you abstract the problem enough. These people often give pointers to related problems. More importantly, they also poke at the problem from a different perspective. Because they are not familiar with the constraints of the environment, they ask questions such as "why don't you do this?" I used to dismiss such suggestions either out of defensiveness – how can they solve the problem in one conversation when I could not for weeks?! – or because I deemed them impractical. I now consider them carefully. What would happen if I really used their idea? Is there a fundamental mismatch or something that can be finessed?

As an aside, do not worry about other people stealing your problem. Chances are that they are not interested. Even if they are, remember that you have a head start because of the insights from your scrutiny. The benefit of talking to other, smart people outweighs the risk of theft.

Another activity that helps while fishing for solution ideas is reading broadly and exposing yourself to various solution concepts. You should read a relevant paper even if you think it is bad. Your goal is not to judge the paper but to let its ideas mingle with yours and learn about other ways of looking at the problem. Keep reading even if your head hurts with confusion. It is hurting probably because it cannot categorize everything you are learning. With time and thinking, you will be able to sort everything.

There are different schools of thought on whether you should read other people's ideas before developing your own. Varian discourages it. I find it hard to think in vacuum and find that more ideas are generated as I get exposed to other ideas.

Once you are done with idea fishing, you will have a list of ideas that could solve the problem at hand. Be sure to include in this list even non-technical solutions. For instance, over provisioning is a possible solution to relieve congestion.

5.2 Idea triage

It is now time to identify, from the list of ideas, the most promising idea around which you will design your system. I think of this exercise as setting up an idea racetrack on which different ideas compete. This race track could be entirely in your head. It does help to write things down, however.

Compare the relative merits of each idea and discount those that have show-stopping weaknesses. This comparison is necessarily a subjective exercise, and the result depends on your taste and experience. If you strongly prefer packet-switching, you may find it hard to be drawn towards a solution based on circuit-switching. The beauty of systems research is that while there may be many "wrong" answers, there can also be multiple "right" answers, with different assumptions and constraints. The act of comparing ideas forces you to be explicit about why you prefer one idea over the other.

5.2.1 Some idea triage how-tos

While comparing ideas, focus on the essence. Try honestly to make each idea work, even those that are not yours. Do not discount existing ideas on issues that are surmountable. A common shortfall is to dismiss ideas that were "not invented here," that is, they were proposed in another context. Know why an idea does not work, independent of its origin.

It helps immensely to work closely with someone during this exercise, so you can take opposing ends of various arguments. If you are like me, you will find it hard to argue strongly against yourself. Working with someone else is both fun and productive. In a recent project, Srikanth Kandula and I argued for over a month. These arguments gave us deeper appreciation of the problem and eventually led to a better system design than what would have been possible had one of us given up easily.

Sometimes, thought experiments and verbal arguments are not sufficient to effectively compare ideas because the comparison depends on complex system properties. In such situations, consider implementing critical parts of each idea to help answer the relevant questions.

5.2.2 Benefits of idea triage

The primary benefit of the triage exercise is the identification of the winning idea. I cannot stress enough the importance of making explicit the core idea behind your system. Systems are complex, and someone else trying to solve a similar problem should be able to identify the key idea behind your system, independent of the context in which you built and evaluated the idea. Help them by separating the meat from the gravy.

An important property that is desirable in the winning idea is conceptual simplicity. You should be able to explain it to non-systems-researchers in a way that make sense to them. That an idea is conceptually simple does not imply that it is simple to implement as well, though that is a definite plus.

Another benefit of idea triage is clarity into the design space. This clarity by itself makes the triage exercise well worth it, even if the winning idea is the one that you had originally conceived. If you are new to the domain and were overwhelmed with all you read, you will now have a clearer understanding.

You may also realize that you are solving a narrower problem than the one you set out to solve. That is, your idea works in a narrower set of circumstances than what you originally imagined. For instance, it may be most effective when network delays are low or when there is stability in traffic. Without considering competing solution ideas, it is hard to get that understanding. It is important to list these these assumptions explicitly. When readers complain that your solution does not work, it is because they are thinking of cases where it does not work. You can avoid this complaint by outlining the cases where it is expected to work. Have you ever been awed by a paper because its solution met the design goals perfectly? Chances are that the authors proceeded backwards, by modifying the design goals to match the solution.

Finally, idea triage enables you to classify past work on your terms. That is, identify an axis of classification along which your solution idea is unique. Such an axis makes it easy to explain how your work is different. For instance, in the work on small enterprise network diagnosis, we realized that a big difference between our approach and past work was that we were maintaining the health of network entities as multi-dimensional variables. After explaining why this distinction mattered, it was easy to explain how our solution was different and for what scenarios our approach was best.

6. BUILDING AND EVALUATING

You are now ready to start building the system. The key is to start small. Do not roll out the whole thing at once. You should check as

quickly as possible if your solution idea works in limited settings. If it does not work in those settings, it is probably not going to work in more realistic situations. The intent is to hit potential hurdles before investing too much time. You may need to revise your solution to overcome these hurdles. Keshav describes this process nicely: start simple, learn as you go, and be prepared to change.[2]

If the main experimental platform is hard to control or too variable, consider using more controlled experimental platforms first. These could be simulation, emulation, or even a separate controlled testbed. In the vehicular WiFi work, my collaborators and I used simulation before rolling out the system to moving vehicles. That helped debug many issues. The extra time invested in making the simulations work was much less than the time it would have taken us to resolve those issues when running on the vehicles.

As you build confidence that the system is working as expected, make the implementation more real. Additional complications will arise that you must discover and resolve.

How you evaluate your system depends heavily on the specifics of the system and the problem you claim to solve. But always consider the following questions:

1. How well does it work?
2. How does it compare to the state of art?
3. Why does it work? Do its assumptions hold in practice?
4. What are the benefits of its complexity?
5. What are the costs of its simplifications?

While the first two questions appear straightforward, pay attention to the quantitative metrics that you use. Use metrics that are appropriate for your system, rather than simply focusing on standard measures.

Authors tend to answer the earlier questions on the list and ignore the later ones. But the later ones are equally important. They provide real insight into the value of your idea and make your work "scientific." Answering the last question is especially important if you are proposing a simple solution. Simple solutions are appealing but are sometimes perceived as simplistic. The last question helps to convince the readers that the simplicity does not come at the expense of effectiveness. Similarly, readers tend to be rightly skeptical of overly complicated solutions. Answering the fourth question shows that any complexity in your solution is worthwhile.

7. MAKING IT REAL

Finally, you completed your research and wrote a nice paper. Are you done? No! You need to make your research go further. Ask what the big impediments are to making your work have impact beyond the paper. There are a range of activities that you may undertake, some easy and some hard.

Release data that you gathered and the code you wrote. These are not purely altruistic activities because they make your work more visible and allow others to extend it more easily. Because Rocketfuel data was released, four papers in the next SIGCOMM conference (2003), i.e., roughly 12% of the program, used that data. All of these papers helped advertise the work.

[2]http://blizzard.cs.uwaterloo.ca/keshav/mediawiki-1.4.7/index.php/Hints_on_doing_research

Another activity is to talk to practitioners, in order to convince them to adopt the technology you built. This exercise is incredibly frustrating. You will learn quickly, perhaps to your surprise, that good technology is not the sole criteria for successful adoption. There are a range of other factors, many of which are outside of your control. But by being aware of them you will have better odds of succeeding the next time.

Thinking seriously about making your research real presents new research opportunities as well. These follow-on research problems tend to be more fundamental because you now have a deeper understanding. You are unlikely to stumble upon these problems if you are not worrying about making your original research real.

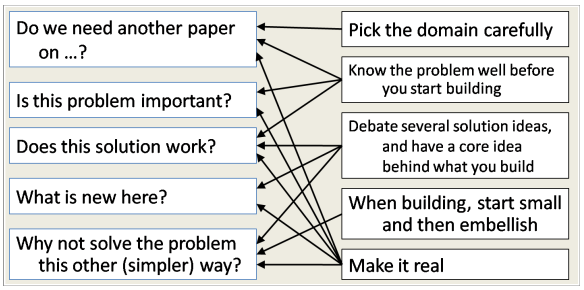
Sometimes, you will hit research problems in other areas as well. By talking to practitioners in the context of our network diagnostic system, we learned that it will not be adopted unless we can explain its analysis results to operators. Showing the results of complex statistical analysis to humans is something that is poorly understood even by UI experts. This problem needs to be addressed before many of the inference-based diagnostic systems, which systems community develops, can be adopted. With help from UI researchers, we recently designed an interface that enables operators to understand and verify statistical analysis. This experience also pointed to a basic weakness in systems whose output must be consumed by humans. Designers of those systems need to incorporate the difficulty for a human to verify its analysis. Without that, designs can be made increasingly complex for incremental gains in statistical accuracy.

Activities to make your work real will often take you outside what you might think of as a traditional systems-researcher role. There is an opportunity cost to these activities – in the time it takes to carry them out, you could be doing more research – but the time invested is well worth it. You already spent a fair bit of time on the original research, you might as well spend some more to boost its chances of having an impact. These activities will amplify your current research and greatly benefit your future research.

8. SUMMARY

Good research processes help avoid common mistakes, improve efficiency, and make us more effective. As a community, we should openly share what does and does not work well. To get the conversation started, I described a five-step process that I find useful.

In the spirit of good research processes, let me argue that this process helps address the common concerns for research systems that I outlined earlier. The mapping from the steps in the process to the concerns it helps address is:



You will be better able to explain why another paper (i.e., your paper) on the topic is needed if you carefully pick the domain and

identify the problem. The investment into problem identification also helps argue that the problem is important because you went through the process of convincing yourself of that fact. You will have an easier time arguing that your solution works if you define its goals and scope precisely, which requires knowing intimately the problem and the the design space. If you have a core idea to what you build and make that idea explicit, it becomes easier to highlight your innovation (which could also be a first-time implementation of an existing idea). Having debated several solution ideas and slowly built your solution, you will be able to more convincingly state why your approach is appropriate for the problem. Finally, if you invest in making your research real, it helps with all of the concerns; it is hard to argue against an impactful system.

9. Q&A

After the talk, several thoughtful questions by the audience provoked interesting discussions. I present those questions below. This presentation is limited by my memory. I am forgetting some of the questions, paraphrasing the ones that I remember, and losing the finer points of the discussions.

9.1 Simple solutions

Q: Suppose one follows all the steps described, but eventually ends up with a simple solution. Then what? Simple solutions are hard to publish.

First, let me say that simplicity is in fact a strength. Simple solutions are what is desirable when designing systems because complex ones have a harder time being adopted by practitioners. That said, I agree that simple solutions can be hard to publish, because reviewers often confuse simplicity with lack of depth or sophistication. The best strategy that I have found against this bias is to highlight the benefits of simplicity as well as show that additional complexity is unwarranted. The latter point can be made by comparing your simple solution to a more complex one or to a (hypothetical) optimal. Show that what your solution leaves on the table in terms effectiveness is acceptably small. The step of debating solution ideas provides good fodder to make this case.

9.2 Negative results

Q: Suppose one follows all the steps described, goes down a particular path that looked promising in the beginning, but eventually ends up with a negative results. Then what? Negative results are almost impossible to publish.

Yes, negative results are very difficult to publish in our community, which is a shame because they can be very powerful. They inform us about what is impossible and help avoid blind alleys. Brewer's CAP conjecture and the impossibility of distributed checkpointing come to mind as excellent examples. The issue, however, is that a negative result is useful only if its exact circumstances can be specified, so we can understand its generality and how to get around it; otherwise, we run the risk of sending a wrong signal to future readers. Circumscribing the exact conditions of a negative result is hard because of the experimental nature of our work. I do not have any general guidelines here and would love to hear from those who do. But if we can figure that out, I believe that the community will be more receptive towards negative results.

9.3 The role of creativity

Q: If you can capture research as a process, what is the role of creativity? Where is creativity in your process?

It is everywhere. Following a process does not reduce the role for creativity. As I mentioned earlier, it helps you avoid common mistakes and become more efficient. It also helps you avoid blind spots. It is not going to make you less creative.

Further, if creativity is defined as the ability to make new associations, then I would argue that certain elements of the process I described makes your work more creative. These elements include reading broadly and talking to people, especially those outside the area. As Einstein said, the secret to creativity is hiding your sources.

9.4 When to stop?

Q: The process you describe is a good one for starting and conducting research in an area. The challenge I often face is when to stop working in an area. Any thoughts on that?

This is a very interesting question! It is also one to which I have not given much thought. I suppose we should stop well before we find ourselves bereft of good ideas and excitement. To avoid hitting that point, my own style has predominantly been to jump from one domain to another, after one major effort. When I have tried to do more in an area, I have often found myself having less fun. I will not recommend this model for everyone, however. There is value in people pursuing different models, and some of us going deeper into their areas, as long as its fun and productive.

9.5 Risky research

Q: Is this process likely to make you prefer less risky research?

No. Lets bear in mind risk is by itself never a goal. What you do not want to do is to take risks that are not worthwhile. Risks are worth taking only when the reward is proportional. The process I described around identifying the problem and careful scrutiny helps you better understand the risks and benefits.

Acknowledgments

This paper is based on what I have learned from my many mentors, collaborators, and colleagues. (Any shortcomings represent my learning failures.) These include Sharad Agarwal, Tom Anderson, Victor Bahl, Aruna Balasubramanian, Miguel Castro, Sally Floyd, Srikanth Kandula, Jitu Padhye, Lili Qiu, Charles Reis, Maya Rodrig, Antony Rowstron, Neil Spring, Arun Venkataramani, David Wetherall, John Zahorjan, Ming Zhang, Yin Zhang, and Brian Zill. Listing one's influences is a dangerous endeavor because one invariably forgets some of them; my apologies to those I missed.

I thank S. Keshav for encouraging me to write this paper. Srikanth Kandula and S. Keshav provided useful feedback for which I am grateful.