On the impact of academic distributed systems research on industrial practice

Michael D. Schroeder

Assistant Director
Microsoft Research Silicon Valley

Introduction

This article is based on the remarks I was invited to make at the 2002 Workshop on Future Directions in Distributed Computing on the impact of academic distributed systems research on the real world. In this short piece I document the gist of the presentation.

1 Good ideas eventually have impact.

My overall assessment is positive. Good ideas do have an impact, independent of where those ideas were developed and tested, although the path from academic research to industrial impact is often indirect and can take a long time. One thing that academic research is good at and that industry finds especially useful is a complete model of a topic or area. Such a model defines a design space and makes clear the full range of possible mechanisms along with the functional properties, feature interactions, tradeoffs, and limits. Research provided such a model for parsing, for example, in the 1960s. A more recent example is the fairly complete understanding we now have of consensus. Work to develop a comprehensive understanding of security technology is nearing this level of completeness. For other sorts of research results, convincing demonstrations of the ideas working in a system along with careful measurements and assessments of the effectiveness can speed the impact.

2 Technology transfer to product groups is hard.

There is only a short window of opportunity during a product cycle when a product group is open to outside ideas. It is difficult for someone outside a company to connect with such a window. The best way to do technology transfer from an academic venue is to get involved in a startup (but see below the negative effect of this) or get involved in a custom system effort. As an example of the latter, the Isis technology from Cornell is used in European air traffic control systems.

Another issue for product groups is matching technology to business opportunities. Most good computer science research tends to produce solutions looking

for problems. This tendency is a result of the higher probability that such bottom up work will generate creative results and advances in the state of the art. Top down work to develop a methodology, a tool, or a system that solves an explicitly articulated problem or market need tends to generate good engineering, not creative research or good theses. Unfortunately (for academic researchers) product groups think of their problems top down. They often will have limited patience for bottom up solutions because they will find it hard to figure out if the technology solves the problem at hand.

3 Impedance match academic research with product groups is low.

Many groups like the industrial systems research lab where I work are built out of the same sort of people as academic systems research groups. We also have the same top level goals: to contribute to the state of the art in computing. This is particularly so in the area of distributed systems. The result is that industrial research groups often have a good impedance match with academic research groups.

In the case of Microsoft Research, we work hard to track and connect with the windows of opportunity in product groups. When these opportunities are engaged we bring to the table both internally developed technology and an awareness of the work going on in outside groups. Thus we can provide a conduit between academic research and the company's product groups.

4 Product groups want help in unpopular and hard problem areas.

Right now there are three areas in distributed computing where I think academic research could help product efforts:

First, improved technology for the software engineering process and for testing. This area is hard because researchers do not have an accurate appreciation of the size of the software artifacts involved, of the size of the test sets, nor of the impact of backward compatibility requirements.

Second, good programming models for web-based services. Web-based services will be where distributed systems programming goes mainstream. Until now the software engineering profession has been unsuccessful in getting most applications programmers to properly use the RPC-based programming model for distributed systems. It is widely agreed that this model will be inadequate for web-based services. What model will replace it? How will this new model be embodied and taught? How will it encompass failure, debugging, and performance considerations?

Finally, better solutions to the system management problem. System management is a large component in the cost of ownership for most customers. System management is also a primary culprit in unreliability and insecurity of

systems, especially distributed systems. Most academic researchers think this is a boring problem.

5 Academic barriers to interaction with industry.

In addition to the generally difficulty of doing technology transfer, universities have in the recent past erected two new barriers to collaboration.

The first was the wave of startups formed by research faculty. These startups remove people and ideas from the pool that could interact with industry.

The second was the attempt by many universities to make money from the intellectual property (IP) generated by faculty and students. The attempts to exploit IP raised the hassle factor for all sorts of interactions between industry and academia. Many collaborations have foundered simply because it was too much trouble to sort out the IP issues. This barrier has also had the effect of reducing the amount of financial support for research that industry channels to academia. I suspect a bottom line analysis would show that the universities lose more in research support than they gain from selling IP.

Recently the wave of startups has diminished and several universities have started to relax their IP views.

6 Fresh Ph.D.s often lack some of the skills that industry values.

The most important technology transfer that the university research community does to industry is via the students that take industrial jobs. While I am not a consumer of software developers and architects, I am a consumer of Ph.D. researchers. Generally speaking I think universities could do a better job of training these young researchers than they do. Of course the very best students will train themselves. It's pretty hard to do anything that will help or hinder them. Perhaps five percent of the output from the top university computer science departments fall into this category. This is the category we mine most diligently. The place where better training might matter is in the five percent to twenty five percent band. Can better training make more of these students into top-notch researchers? I think so.

When screening candidates for research positions we look for students who have identified an interesting problem, produced a creative solution, and concisely characterized the insights gained. We evaluate a thesis both as an example of the quality of the research skills a candidate has developed and as an example of the quality and impact of the technical advance of which the candidate is capable.

A good candidate will have intellectual spark, drive to succeed, independence, and a good awareness of technical context. The good candidate will have a broad range of technical interests and have flexibility to work on a variety of problems. They will also realistically assess the impact of their work. Inflated claims of the impact do not help a candidate's case.

Ocassionally we see symptoms of what we call "advisor failure." This most often manifests itself in a student lacking technical perspective, a skill we value highly. Such a student usually has done a good job of drilling down into a particular technical issue but does not understand what else has been done before, how the "new" research relates to problems and solutions already published, and whether the line of work is worth pursuing further. With respect to the last point, I sometimes believe that a candidate feels a need to claim interest in doing future work on the thesis topic just to validate that the topic is a deep and interesting one, even if it isn't. Often I'd rather that a student not want to follow up, but instead be ready for new problems and have thought about some such problems that would be worth pursuing.