Microsoft Research and the industrial research cycle

January 30, 2017 | By Microsoft blog editor

A personal view

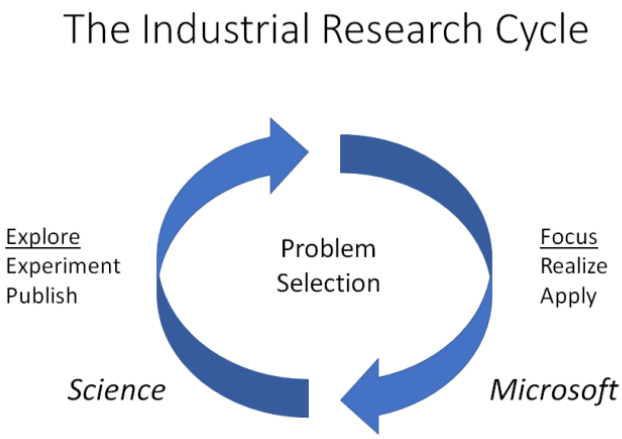
By Thomas Ball, Research Manager, Research in Software Engineering (RiSE) group, Microsoft Research

The industrial research cycle

Here is what I have told new hires of Microsoft Research (MSR) since I became a manager some 14 years ago:

> *MSR gives you the freedom to explore and expand the bounds of scientific knowledge, as in academia, but with the added challenge to align your scientific pursuits with company problems and to drive for impact on Microsoft, especially as you grow in seniority at the company.*

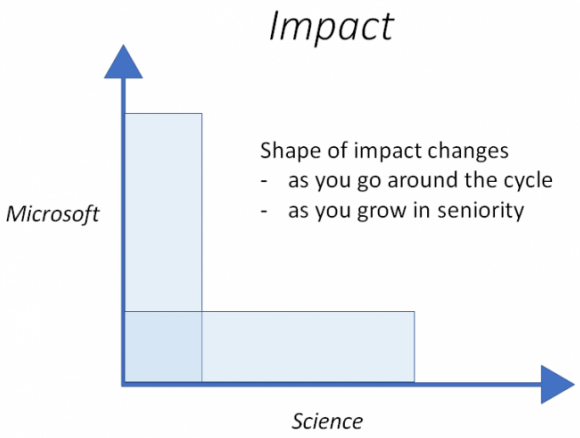
This statement is still as true today as it was when I joined MSR 17 years ago and reflects MSR's associated goals of advancing scientific frontiers *and* positively impacting the company.



I use the model of "The Industrial Research Cycle" to explain how MSR works. Researchers have the freedom to select problems and to *explore* in their discipline (the left side of the cycle) to advance science. They also have the responsibility and opportunity, once sufficient exploration has taken place, to *focus* their attention on an area that they believe can produce impact for the company (the right side). Ideally, the problems/solutions that one explores on the left side of the cycle eventually drive impact on the right side. And the experience one gains from the right side not only validates the science at scale, it also pushes exploration in new directions in the next phase. A researcher will go around the cycle many times during their career.

Impact over time

It is difficult to simultaneously explore and focus, and to do both well! Instead, one needs to engage in phases of exploration and focus over years.



I use the "Impact" diagram to explain the different forms/shapes of impact. The x-axis measures the level of scientific impact. The y-axis measures the level of Microsoft impact (see box). One's impact is measured by the area under the curve. The shape of one's impact curve changes over time, both as one goes around the industrial research cycle and as one grows in seniority at the company.

During an exploration phase, the shape of one's impact curve generally is horizontal because the primary audience is the scientific community. During a focus phase, the shape of one's curve is generally vertical, building on the foundation.

As one grows in seniority in the company, the expectations for focusing on Microsoft impact increase. On the other hand, junior researchers enjoy more freedom to explore. Fresh Ph.D. hires at MSR still have much work to do to establish themselves as recognized experts in their fields. While some may indeed engage with product teams early in their career, we do not expect junior researchers to jump right in to address problems of the company.

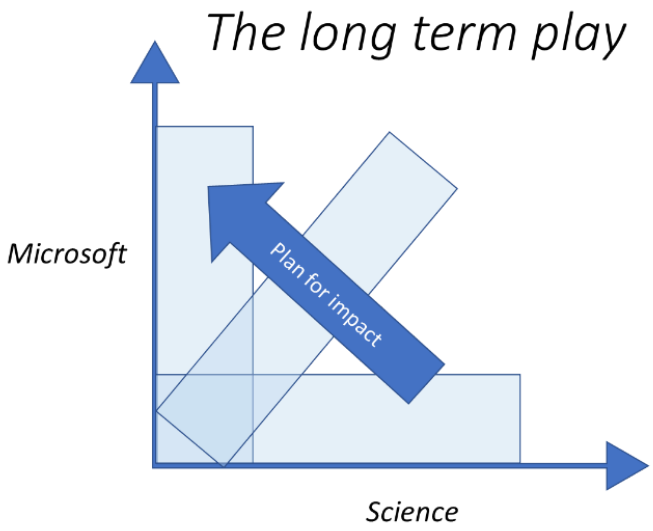
Some measures of Microsoft impact

- Improving the reliability/performance of Microsoft systems
- Changing how Microsoft designs/implements systems
- Opening new opportunities for Microsoft product groups

While we encourage our researchers to actively publish, MSR does not emphasize quantity of publications. Quality is our top priority.

Pipelines and partners

MSR invests in scientific efforts that may not have immediate impact on Microsoft but that will build a new muscle/capability for the company in the long run. I use the "The long term play" diagram to show that a coordinated and long-term effort often is needed to turn scientific results into company impact.



Below are three examples showing the path to impact, which requires working closely with partners over the long term, building relationships and trust, and changing company culture through new ways of approaching a problem.

Automated defect detection and driver quality

In late 1999, Sriram Rajamani and I started the SLAM project at MSR to investigate new approaches for automatically finding code defects in device drivers. When the Windows Driver Quality was formed in 2002, Byron Cook, Jakob Lichtenberg and Vladimir Levin came into the team to deliver a tool called Static Driver Verifier (SDV), based on the SLAM engine. The first version of SDV was delivered with Windows in 2004. During the last decade, SDV's underlying analysis engine has been improved/replaced by MSR three times (see papers on SLAM2, YOGI and Corral) by different sets of researchers working closely with the Driver Quality team, including Ella Bounimova, Aditya Nori, Rahul Kumar, Shaz Qadeer, Akash Lal and Shuvendu Lahiri.

From empirical software engineering to tools for software engineers

In 2004, I hired Nachi Nagappan into MSR to spearhead Empirical Software Engineering research at Redmond. For five years, Nachi and colleagues Brendan Murphy, Jacek Czerwonka, Christian Bird and Thomas Zimmermann studied key issues affecting software quality and developer productivity, through analysis of product version histories, bug databases and other data sources.

To scale such analyses across the company, Wolfram Schulte joined with Nachi, Brendan and Jacek to create CODEMINE, a data analytics platform for collecting and analyzing Microsoft software engineering process data. This project started around 2009 (codenamed SWEPT) and culminated around 2013, giving insight into software engineering problems across Microsoft product groups. CODEMINE was essential to making a case for the formation of a new team called Tools for Software Engineers, which is moving the company to a cloud-based software engineering infrastructure.

Computer science education

More recently, the Touch Develop project (www.touchdevelop.com) started in MSR in 2011 to make it possible to program scripts *for* smartphones *on* smartphones. An unexpected use of Touch Develop was in K-12 computer science education— teachers found that children were engaged by scripting their smartphones to react to environmental stimuli.

This turned into a project with the BBC to create a small physical computing device with an easy-to-use coding platform (built on Touch Develop). One million of these devices, called micro:bits, were delivered in 2016, enough for every fifth grade student in the UK to receive one. Because of the BBC micro:bit, Microsoft is now investing in a new programming platform for CS education.

Organizing for big impact on big problems

Today, we find a handful of companies developing planetary-scale distributed systems. Amazon, Facebook, Google and Microsoft all have built such systems, and are engaged in optimizing them for performance, reliability, availability, security and privacy. Microsoft Azure is one such system, which provides compute, storage and networking services, and interacts with an ever-growing number of mobile devices and IoT endpoints.

Optimizing every level of the stack, from the hardware assets, to the low-level operating system code, to the user-facing services, is key to its success, and affords opportunities for researchers across a wide range of disciplines, including those in systems, formal methods, software engineering and programming languages.

Here are four new, larger-scale projects related to the cloud that the RiSE group is deeply involved in:

- The P programming language is transforming the way Microsoft programmers undertake the task of building large asynchronous systems. P has been used to develop USB 3.0 drivers in Windows, as well as services in Microsoft Azure.

- Project Everest is constructing a high-performance, standards-compliant, verified implementation of the full HTTPS ecosystem, from the HTTPS API down to and including cryptographic algorithms such as RSA and AES.

- Project Parade is parallelizing a large class of seemingly sequential applications by treating runtime dependencies as symbolic values. The results of this project are leading to substantial performance gains in popular algorithms for machine learning and big data.

- Project Premonition aims to detect pathogens before they cause outbreaks, by creating new technologies to autonomously locate, collect and computationally analyze the blood-borne pathogens carried by mosquitoes.

Want to be part of the industrial research cycle?

No matter if you're exploring or focusing, the ride at Microsoft Research is an exciting one. If you are interested in joining us on this journey, please visit our careers page.

Related Items

Empirical Software Engineering Group (ESE)

The Empirical Software Engineering working group empowers software development teams to make sound data-driven decisions by deploying novel analytic tools and methods based on ESE's empirical research on products, process, people, and customers.

Research in Software Engineering (RiSE)

coordinates Microsoft's Research in Software Engineering in Redmond, USA. Our mission is to advance the state of the art in Software Engineering and to bring those advances to Microsoft's businesses. Visit our career page. Foundations of Correctness Formal Methods - Ivy, Lean, Z3, Symbolic Automata,

FORMULA Programming Languages - Dafny, F*, Koka, P Software Productivity Program Synthesis - BlinkFill, FlashFill Debugging and Testing - Time travel debugging, automated test generation Program Analysis - Corral, Duality, Angelic Verification, SymDiff…

Research Areas

Programming languages and software engineering

Project Premonition

Established: March 2, 2015

Interested in evaluating Project Premonition technologies and data? Sign up here Project Premonition aims to detect pathogens before they cause outbreaks Emerging infectious diseases such as Zika, Ebola, Chikungunya and MERS are dangerous and unpredictable. Public health organizations need data as early as possible to predict disease spread and plan responses. Yet early data is very difficult to obtain, because it must be proactively collected from potential disease sources in the environment. Researchers estimate between...

SLAM

Established: November 5, 2001

SLAM is a project for checking that software satisfies critical behavioral properties of the interfaces it uses and to aid software engineers in designing interfaces and software that ensure reliable and correct functioning. Static Driver Verifier is a tool in the Windows Driver Development Kit that uses the SLAM verification engine. "Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas,...

Tools for Software Engineers

Established: June 29, 2012

The Tools for Software Engineers (TSE) team mission is "Enabling Microsoft to accelerate software development". TSE contributes to and innovates on major parts of Microsoft's engineering system. TSE's current focus is to shorten the continuous integration cycle time which is the minimum time required for a typical source code change to move from changed sources via compilation and unit testing to deployed binaries. See projects CloudBuild, MSBuild vNext, and CloudStore. TSE provides additional engineering services...

Share this page