Software Engineering Research Lab to Airplanes, Orion and Beyond

One professor's journey shaped by industry-university collaboration

Suresh C. Kothari
Electrical and Computer Engineering Department, Iowa State University
kothari@iastate.edu http://www.linkedin.com/in/surajkothari

ABSTRACT

This paper is a short story of my adventures of the past 20 years trying to integrate academic research with software engineering problems in industry. I share the challenges I encountered on the way, my failures and successes, evolution of my research, and its adoption in industry. Though I faced many hardships, I feel great satisfaction in knowing that my research is applied today in the design of avionics systems, automobiles, and even in NASA's Orion program. My latest adventure and honor is an opportunity to participate in two visionary Defense Advanced Research Project Agency (DARPA) programs aimed at developing an innovative technology to fight the war against sophisticated malware that poses grave security threats to individuals and nations. Without working with industry, it would not have been possible for me to formulate rigorous but practical research problems. These problems have shaped my research. I narrate my story to provide insights into bridging the gap between academic research and the problems industry practitioners face. My hope is the reader can benefit from the story and be able to achieve in 10 years what has taken me 20 years. I also hope that my story encourages industry practitioners to work with universities.

1. INTRODUCTION

There are at least two major hurdles in bringing together software engineering researchers and industry practitioners. First, companies operate under the tremendous pressure of deadlines and the need for short-term gains. It is difficult for companies to engage in a partnership involving fundamental research that requires a long-term vision. Second, Software engineering is a nascent discipline and its fundamentals have not yet fully emerged, therefore it is especially hard for industry to distinguish between fashionable versus fundamental research. These challenges bring with them big opportunities for pioneering fundamental research, revolutionary technological advances, big economic gains, and robust software products to benefit society.

SER&IP'16, May 17 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897022.2897032

As a professor at a US university with a strong emphasis on research, I have experienced that being at the intersection of research and application makes it difficult to meet the publication and other metrics of academic research excellence. Working with Industry requires an investment of time and resources in maturing research beyond what mere publication requires. Nonetheless, I have experienced that persevering to combine research and application eventually leads to relationships and projects that fuel scientific research. Our research on software analysis and transformation, starting with tool-assisted parallelization, led to an important application for safety-critical control systems software, which in turn furthered scientific research leading to DARPA funded projects on detecting sophisticated malware. This virtuous cycle has led to our research being applied in the design of avionics systems, automobiles, and even in NASA's Orion program.

It is hard for me to imagine that my research and teaching would have evolved the way it has without working with industry. My interactions with industry have immensely impacted my choice of research problems. Before working with industry, I gravitated to problems formulated by reading papers written by other professors like myself. I doubt if practitioners who created, maintained, or managed large software read our papers. And that did not matter to us, we could go on publishing papers and get rewarded for them. The key government agency that funded our research required us to include a section on the broad impact of our research but we knew that the proposals' reviewers judged us primarily by our publications. Fortunately, I came to the realization that the kind of research we conducted does not really have the broad impact we claim in our proposals. I wondered what I could do differently for my research to have impact beyond the academic circle of researchers.

It is easy to believe that research is a futuristic endeavor, and so its impact will come later from unexpected corners. It happens but rarely. As I kept wondering how to change my research for it to be used to solve real-world problems, I got a lucky break. A professor in atmospheric science approached me because of my expertise in parallel computing. He wanted my help to parallelize a climate model software. It was 150 thousand lines of FORTRAN code. As my student and I started working on that software, we realized the need for tools to work with that mountain of code. We built tools, they kept evolving, and eventually brought us to a point where we could parallelize the code very efficiently by using tools to perform the low-level, tedious, and time consuming tasks for us. Industry practitioners became inter-

ested in our tools to analyze and transform software. They wanted to work with us to create tools to address their problems with large software and thus started our university-industry partnership.

Working with industry has given me valuable exposure to a large variety of complex problems of large software. This exposure has fueled the evolution of our software analysis and transformation tools research. It has revealed exciting opportunities to develop powerful mathematical abstractions to manage complexity of software. Without such exposure, I doubt that we would have been able to develop good abstractions applicable to real-world software. The tools and abstractions evolve in a feedback loop - the tools enable understanding that leads to good abstractions, and in turn those abstractions become the foundation for building even more powerful tools. The university-industry partnership has helped me develop a vision for automated tools research. I will quote Frederick Brooks [10] to articulate it:

> If indeed our objective is to build computer systems that solve very challenging problems, my thesis is that IA > AI, that is, that intelligence amplifying systems can, at any given level of available systems technology, beat AI systems. That is, a machine and a mind can beat a mind-imitating machine working by itself.

Working with industry paved a direct path for my research to have the broad impact that reaches far beyond the circle of academic researchers. The research of my team has become valuable to industry. I founded EnSoft [6], a company to produce software analysis and transformation tools to manage the complexity of large software. Today, 310 companies in 27 countries, including all major avionics and auto-mobile companies use EnSoft's products. Also, 19 academic institutions including 8 universities competing in the EcoCar Challenge Competition [4] use EnSoft's SimDiff product for model-based development of state-of-the-art control system software. And, students in 174 institutions in 26 countries have academic licenses of Atlas [8,12,13], a platform to build tools to analyze, understand, transform, and verify software using mathematical visual models. Leveraging Atlas, we were able to propose to Defense Advanced Research Projects Agency (DARPA) a new approach to detect highly sophisticated, one-of-a-kind, unforeseen malware that can have catastrophic consequences. This has led to multi-million dollar DARPA funded projects in collaboration with our research lab and EnSoft.

After narrating my story, I isolate lessons learnt and describe a model of exchanges between research and industry that I have found works well. I include my thoughts on the mega trends in software and its applications that I believe will shape software engineering research and the need for industry-university partnerships.

2. UNIVERSITY-INDUSTRY RESEARCH PARTNERSHIP

One of the key factors which enabled me to bring our research to bear on industrial applications was the desire and concerted efforts to shape our software engineering research to address practical needs. It started with an interdisciplinary research project. In 1992-94, our group at Iowa State University, a group at the Argonne National Lab (ANL), and a group at the National Center for Atmospheric Research (NCAR) worked on parallelization of a regional climate model called MM5. It took more than three years to parallelize the 150K lines of FORTRAN code for MM5. It was very tedious and time consuming to go through the legacy code developed by a large group of atmospheric scientists over a period of twenty odd years. The experience gave us the insights and motivated us to develop a semi-automatic, domain-specific tool to facilitate parallelization of a class of climate model codes. It took about two years to develop the first version of ParAgent [16].

The ParAgent worked well. A post-doc could parallelize the MM5 code in two weeks compared to more than three years it took a team of four programmers to do the same manually. We then received funding from Environmental Protection Agency (EPA) and Pacific Northwest National Laboratory (PNNL) to evolve and apply the ParAgent tool. Encouraged by the tremendous savings of cost and human labor realized by using ParAgent, we decided to explore its use in commercial projects. We found that the companies would be interested if we could expand the scope of our research to build new tools to improve efficiency of specific software engineering tasks of interest to the companies.

Expanding the scope of ParAgent was a difficult task in an academic setting. It required substantial software development. It was difficult to get research funding to support it. Reviewers denied our proposals because they did not consider the work to be basic research. Eventually, we got a break because of a new Iowa State University initiative aimed at fostering collaborative research with companies. It provided matching funds to encourage companies to engage in collaboration with us.

Our first collaboration with industry started with Rockwell Collins because of this initiative. They were interested in applying our research to automate the analysis of safety-critical software to produce the evidence necessary for certification of their avionics software. The collaboration was a great experience to gain domain-specific knowledge about safety-critical software and its development process. An excellent collaborator at Rockwell Collins helped us understand the applicability of our research to software-enabled control systems.

Despite this successful partnership, it was hard to break out of the tensions arising from big differences between academic and industry criteria for evaluating research. It was not easy to translate our work into an adequate number of academic papers to be considered reasonably productive for my annual faculty review. The funding was quite limited and not enough to continue the work at a pace to generate substantial interest for big industry or government funding. We submitted joint proposals with the company to defense and aerospace agencies but we did not succeed, probably because our research was still in early stages. Since our projects were funded annually, we could not offer multi-year research assistantships to attract good PhD students. With new students every year, a significant time was spent on training them. By the time they gained some proficiency, it was time for them to complete their Masters degree and leave. Overall, it looked like a loosing battle.

In 2002, I decided to try a new experiment, an attempt to foster our research outside of university. I founded EnSoft [6]. Because of my collaboration with industry, I had come to know of many practical software engineering prob-

lems that we could solve for industry. My idea was to make money by solving these problems and use the profit to continue our research in software engineering. Having a company was also a practical solution to the intellectual property (IP) issue we had faced. I could influence the IP negotiation directly without having the university lawyers in the middle. My undergraduate assistant and a former PhD student joined the company and that made it easier to continue my university job along with the company.

Starting a company opened new doors. As a company, we could get projects from the product groups besides the R&D wing of Rockwell Collins. An R&D collaborator introduced us to the flight control systems group manager. It eventually lead to an opportunity to work with a Distinguished Engineer at Rockwell on a tough software problem they had faced while developing the display for the Boeing 787 Dreamliner. We helped them to solve the problem, and the software we developed actually became a part of the Boeing 787 Dreamliner cockpit display system.

The manager at Rockwell Collins gave a project to apply our research to build a prototype tool for the flight control systems. After the delivery of the prototype, the manger became interested in having us develop a tool to audit safety-critical software as a part of the certification process. Three years after starting the company, it was our first break to be able to apply our research to build a software tool that would be actually used to perform a practical task of significant value to a major avionics company. The tool turned to be important to the avionics company and it became a qualified tool that they could use in their certification process.

After developing the tool, we advertised it as a product on our website. We started getting requests for trials from a variety of companies. Seven years later, it has turned out to be a product that is used worldwide by more than 300 companies, including all major avionics and auto-mobile companies. It turned out that the tool serves an important need for developing complex control systems software.

There is an important lesson here. We could not have thought of the need for the tool. It was the flight control systems group manager who could think of the need because of his practical experience. On the other hand, it would have been a real challenge for the manager to develop the tool in-house; it required a somewhat unique blend of knowledge of mathematics and software engineering that we had. A partnership between research and industry practices was crucial to come up with a successful product. In addition to the core algorithm that came from our research, requirements of usability, software quality, and the ability to gather and satisfy the corner-cases would have been almost impossible to meet in a university research group. The fact that we had a company gave the manager the confidence that these requirements would be met.

Initially, it was so hard to get companies to take a look at our research. Interestingly, after trying our product, two executives from Toyota flew to Ames Iowa to discuss their need for software tools. It was the time when Toyota was in news because of the problems with their cars. It was not clear if the problems originated from software. It was a lucky break for us. We now have a multi-year engagement with Toyota to create a tool-chain to ensure high reliability of safety-critical software in their cars.

In 2005, an opportunity came along to work with 1000+ IBM software engineers in the hypervisor group. Through a faculty fellowship program offered by IBM, I could spend a semester working closely with three senior software engineers. These engineers were considered a think tank that used to be called in to solve complex bugs in the hypervisor software. I had an opportunity to observe how these experts reasoned about software with millions of lines of code to find the defects that lead to strange execution behaviors resulting in system crashes on large servers at customer sites. I observed the clever strategies the experts had developed. I also noticed that their strategies were intrinsically limited because of two important factors. First, they did not have good abstractions as a foundation for their strategies. Second, they only had rudimentary program analysis tools to mine large software. The two factors were interrelated. Without powerful tools to extract the cross-cutting semantics from large software, it was impossible to build appropriate abstractions.

The one-semester faculty fellowship program at the IBM got me thinking about research on Intelligence Amplifying Mechanization (IAM), a technology to facilitate powerful strategies to reason about software. The idea was a generalization of the ParAgent research we had done earlier. To deal with complex software problems, we need a good man-machine combination to reason about the problem. Human reasoning is critical to solve the problem and the machine is needed to amplify the reasoning in order to scale it to large software.

For the last seven years, we have been working on the IAM technology. At the university, we have been focused on developing mathematical formalism to serve as the foundation for the IAM technology. At the company, we developed a prototype tool so that our research group could perform experiments to assess and evolve the formalism by applying it solve complex problems of large software. After six years of groundwork, we landed a multi-million dollar contract from DARPA to advance Atlas [12] as an IAM platform. We leverage Atlas during the DARPA APAC Project to build a Security Toolbox to detect highly sophisticated malware in Android Apps [13]. We are now funded on another multi-million dollar DARPA project that aims to address *algorithmic complexity* and *side channel* vulnerabilities in software. Both are joint projects between EnSoft and Iowa State University. The mathematical formalism we have developed through research at university and a carefully engineered platform at a company have given us a distinct advantage to compete. Our research has led to unprecedented capabilities to detect sophisticated malware. In a DARPA competition involving 77 Android application challenges, using the Android Security Toolbox we created, our team identified the malware in all but 6 apps with no false positives. DARPA announced our team as the top performer in this challenge, among contenders that included premier institutions and top software engineering centers.

3. LATCHING ON MEGA-TRENDS

Based on experiences, I believe that to undertake fundamental research and to foster successful partnerships with companies, it is definitely helpful and maybe even necessary to latch on to mega-trends. These will be trends that involve issues where critical societal needs and significant amounts of money are stake and the issues are so complex that the solutions will require many decades of research and engineering work. I will mention trends we have latched on to and

which I believe have played as a guidepost for our software engineering research:

- A dramatic reversal in human to computer cost ratio - a rough estimate would be a reversal of 500,000 to 1 from 1960 to today.
- Gigantic and Pervasive Growth of Software - The IBM 360 operating system was envisioned to have a footprint of 16K; it eventually turned out to be a 64K footprint and that was considered "very fat" operating system - contrast that with a modern operating system with more than 50M lines of code.
- Extreme Needs for Highly Reliable Software - If a desktop crashes, it can be rebooted, not an option if it is the flight deck computer.

The mega-trends suggest important problems for research. For example, automated software engineering is an important problem as its solution can lower the labor cost. The focus of automation can be one or more commonly performed software engineering tasks. In case of the ParAgent tool, the task to be automated is that of transforming sequential code to parallel code. Another important topic would be software analysis to detect malware.

Revolutionary fundamental research is needed to make quantum jumps in solving the mega-trend problems. Also, the mega-trend problems are inherently of interest to the industry and the government. Latching on to mega-trend problems have helped us in forging partnerships with industry. It has taken us a lot of effort and long time to evolve fundamental research that can lead to concrete advances in solving mega-trend problems. However, once on track, the research has the potential to continue for a long time with many ground-breaking advances. For us, it has also led to major DARPA projects, among the largest projects at the university. The faculty fellowship opportunity at a company was the critical beginning that six years later resulted in our success with these DARPA projects.

4. REFLECTING ON UNIVERSITY INDUSTRY PARTNERSHIPS

It has taken me twenty years and much hardship to achieve success by forging university-industry partnership. My story may not help others in minimizing their hardship, however, it can help them achieve in ten years what has taken me twenty years.

I have used a narrative style to catch attention. The insights and lessons learnt may not have been clear enough through my narrative style. My narrative style also does not directly address an important issue, a danger many professors perceive. There is a belief that working with industry dilutes the quality of underlying basic science. It is not clear from my narration if our research involved meaningful scientific advances and insights. This section is aimed at addressing these points so that the paper can be of better benefit to the readers.

4.1 How to develop the partnership?

- The first thing is something I did not do, but I realized its importance later. It is important to find good mentors and seek their help. I think one needs both types: the academic and the industry mentors. My criteria for an academic mentor are: (a) the person has significant experience of serving and prospering in an academic institution, (b) the person has done research that has had a clearly visible practical impact, and (c) the person has visibility in industry. My criteria for an industry mentor are: (a) the person has had a distinguished technical career in industry, and (b) the person can effectively communicate the technical challenges from her or his work. Good mentors can be very helpful to expedite progress. For the last few years I have been compiling a list of good mentors to help my graduate students.
- I learned an important lesson early on. I participated in meetings the university arranged to foster collaboration. Professors presented their research during these meetings. It was a seemingly good idea to inform industry about different research projects professors are working on so that they can choose the research they like. The idea did not work as well. My takeaway was that, it should be the other way around. Professor should first get informed about the problems industry faces as opposed to trying to sell their research to industry. The industry practitioners get disenchanted listening to talks from professors who have little knowledge of practical problems.
- Interdisciplinary research helped me to start relationships with industry. It gave me an idea of what it takes to work on large software written by others. Using this experience, I developed tools that I could show to industry as concrete artifacts instead of describing hypothetically how our research can help to manage complexity of large software. It is important to convey what the research can do as opposed to describing it as you would do for publishing a paper.
- Academia and industry evaluate research quite differently. The novelty of algorithms, the papers, the citation counts, the publication venues, the best paper awards are among the top indicators of quality of research for academia. For industry, it is mostly about how effectively the research addresses their problems and how much training and other effort it would take to adopt the research. The two criteria are not mutually exclusive but it does require a lot more effort to satisfy both. When working with industry, it is important to keep in mind the contrast between its needs and priorities and those of an academic setting.
- Developing a long-term partnership with industry is like growing a big tree, it takes time. A seed germinates and you have to be extra cautious when it is a seedling, it can get easily trampled on. It takes a long time for it to grow and become a beautiful tree. It requires patience and constant nurturing.

The initial years, in my case almost ten years, were very difficult. While I was working hard on driving research in a direction that would be amenable to a highly successful partnership with industry, my research productivity, as measured by metrics such as the number of papers, suffered and I faced tough problems and had to forego recognitions that I could have attained had I continued academic mode of research. Since I was already a full professor when I migrated to practical software engineering research, I did not face the danger of losing my job. I had a colleague who engaged himself into a similar activity before he was tenured

and eventually failed to get tenure. Fortunately, he started a company and did become very successful. It is an arduous journey to undertake research that can lead to highly successful partnerships with industry.

The numbers game is hurting credibility of academic research. As David Parnas, a highly respected researcher in software engineering, points out [14] "the computer science literature is being polluted by more and more papers of less and less scientific value." The industry cares about the benefits and uses of research for real-world software problems; you cannot convince industry to work with you by producing a long list of papers you have published.

4.2 What are the important benefits of partnership?

- Working with industry changed my vision of research. The exposure to their constraints and challenges helped me to formulate different types of research problems. I also got a clear picture of how research can impact the real-world software. I would consider this the most important benefit. It is this vision of research that has helped me to write successful multi-million dollar proposals involving research that impacts both the nation and the industry.

- Another important benefit for me was the founding of my own company. The industry partnership has been crucial for me to understand what commercial products could be valuable and why.

- The industry partnership benefited my educational efforts. It has enriched my teaching. In essence, I teach students how to reason about large software. By using real-world problems I have learnt from industry, I can engross my students while I teach them how to think. Without those problems, it would be a superficial exercise. My industry contacts helped me to get about $\$350,000$ dollars in cash as the seed funding plus equipment and software from companies to support the effort I led to start an undergraduate software engineering degree program.

- Through industry partnership, I got to see the tools industry uses and I could avoid reinventing the wheel by replicating what they already had. Understanding the shortcomings of their tools, helped me to get new vision of far more sophisticated tools. It was like getting the vision to build a MRI machine to view internals of software.

- I received IBM Fellowship for a semester. It also gave me a chance to closely observe how the distinguished engineers in IBM solved complex software problems. This experience led research that culminated in the Atlas platform [8,12] for building tools, and eventually led to multi-million dollar research projects.

- I started a company to break the academic barriers to forge relationships with companies. In retrospect, that turned out to be a very important decision. It has enabled me to extend my research and its outreach to industry on a scale that would be hard to achieve without the company.

4.3 Does industry partnership mean dilution of scientific research?

This is a controversial topic. For me, instead of dilution, the industry partnership has been pivotal to undertake deep scientific research. Initially it was a lot of engineering to build tools to analyze and transform large software. It changed after we built sophisticated tools, and later even a platform to build tools. As mentioned before, the vision for building such tools came because of partnership with industry.

With these tools, it has become possible to perform unprecedented experiments on large software that have revealed fundamentally new ways of analyzing and transforming software. I will elaborate this point with a concrete example from our research.

4.3.1 A research example

Formal verification of large software has been an illusive target, riddled with problems of low accuracy and high computational complexity [1, 7, 9, 11, 17]. With growing dependence on software in embedded and cyber-physical systems where vulnerabilities and malware can lead to disasters, an efficient and accurate verification has become a crucial need. The verification should be rigorous, computationally efficient, and automated enough to keep the human effort within reasonable limits, but it does not have to be *completely* automated. The automation should actually enable and simplify human cross-checking which is especially important when the stakes are high. Unfortunately, formal verification methods work mostly as automated black boxes with very little support for cross-checking.

Avionics companies have to verify that their safety-critical software works correctly. We worked with an avionics company to produce tools to support their verification process. That exposure has evolved into a new vision for verifying software. It is to integrate automation with human intelligence to solve software verification problems where complete automation has remained intractable. The key innovation is a mathematically rigorous notion of verification-critical evidence that the machine abstracts from software to empower human to reason with. We use *visual software models* as the key enablers for verification-critical evidence. The goal is to create a powerful fusion of automated evidence abstraction and evidence-based reasoning and verification. The evidence-based verification is automated wherever possible and complemented by human reasoning wherever needed. A side benefit is that the evidence can be shared so that a team of analysts can perform the verification collectively. We call the overall approach Evidence-Enabled Collaborative Verification (EECV).

We have developed the EECV machinery to tackle the verification problem of Matching Pair Verification (MPV), specifically to verify the correct pairing of mutex lock and spin lock with their corresponding unlocks on all execution paths. We applied our EECV machinery on three recent versions of the Linux operating system with altogether 37 MLOC and 66,609 verification instances. Our evaluation results on these instances in comparison to the state-of-the-art Linux verification tool shows the effectiveness and scalability of our approach.

4.3.2 An Overarching Research Vision

Working with industry helped me to develop an overar-

ching vision for research. I realized that analyzing large software is the task central to enhance productivity, safety and cybersecurity. It is a task too big for humans alone and too complex for machines to do accurately. The key is to build human-machine collaboration systems.

With the focus on human-machine collaboration, the key research questions are:

- What should be the query language for humans to formulate their questions to a machine to obtain information that is crucial to understand large software?
- What representation should the machine use to communicate knowledge about software to humans?

Understanding large software requires iterative refinement. The questions evolve as human understanding increases. A human-machine collaboration system must support interactive collaboration. For example, a human should be able to follow up with question to ask the machine to refine the answer to a previous question. In effect, the answer to a question may need to serve as the input for the next question. This can be facilitated if it is a composable query language and it integrates seamlessly with the knowledge representation the machine uses.

There are two crucial program comprehension needs: (a) enable the human to develop the knowledge that can relate low-level program artifacts to high-level knowledge in order to apply domain-specific ideas to solve complex problems, (b) enable the human to recognize the coding patterns used by developers to manage complexity of their software. Without leveraging the domain knowledge and the coding patterns, one is left to mere generic analysis which becomes intractable. While it has been intractable to parallelize large climate model software using generic compiler techniques, we could create a tool to do so by developing an automated parallelization technique enabled by the knowledge about a class of numerical methods. While it has been intractable to achieve high accuracy for verifying the Linux operating system software with generic formal verification techniques, we could create a highly accurate verification technique by recognizing certain coding patterns used by the Linux developers and using them to improve the accuracy.

Many real-world problems are intractable to solve with generic automated analysis or formal verification. The practitioners in industry have valuable knowledge which they routinely use to work on their complex problems with large software. The issue is that the application of knowledge is mostly manual and that incurs tedious efforts, time delays, and human errors. There is an opportunity for significant automation and the need for research to make it a reality. Automated software engineering researchers in academia have a huge opportunity to work with industry to develop powerful knowledge-based automated software engineering techniques and tools. Since the domain knowledge resides with industry practitioners, a partnership with industry is crucial.

As my research evolved, I have realized that for automated software engineering to be highly beneficial, it needs to go far beyond automation techniques and tools. It actually requires new thinking to solve problems differently. The way to solve a problem using an automated tool can be very different from solving it manually.

The real power of automation is the ability to build models to solve complex problems. An automated tool can enable knowledge-based modeling that is impossible to do by hand because of the humongous size of software. The practitioners in industry are not used to build such models and without them the benefits of automation can be quite limited. We need research on developing powerful models to abstract and solve complex software problem. The thinking that goes into developing such models is non-trivial and must also be dispersed through software engineering curriculum and training courses for industry. While teaching a course on how to reason about large software using automated tools, I observe that the students initially use tools to merely automate some manual steps without changing the manual problem solving method. After learning modeling techniques, they change the problem solving method itself to unleash the tremendous power of automation.

Research on knowledge-based automated software engineering is also critically important for cybersecurity. Especially concerning to the Department of Defense (DoD) are the malware attacks that a nation or a terrorist organization can launch to cause catastrophic events or to steal secrets by inserting just a few lines of malicious code in mission-critical software. DARPA has launched big research programs to advance automated program analysis to address cybersecurity. I have participated in two such programs: DARPA Automated Program Analysis for Cybersecurity (APAC) [2] and DARPA Space/Time Complexity Analysis for Cybersecurity (STAC) Program [3]. These programs focus on sophisticated malware, which is one-of-a-kind and specially crafted for the target. Detecting it can be like searching for a needle in haystack without knowing what the needle looks like. Detecting sophisticated malware in large software is a problem too big to be solved by humans alone and too complex for machines to do accurately; it requires a human-machine collaboration system.

Sophisticated malware is not just a concern for the department of defense. For example, software in the car can let hackers send commands through the car's entertainment system to its dashboard functions, steering, brakes, and transmission, all from a laptop that may be across the country. The automobile companies, in fact all companies that have internet-enabled mission-critical software in their products, will need to address sophisticated cybersecurity problems rooted in software vulnerabilities. It is another opportunity for industry and academic researchers to work together to develop new software analysis and verification technologies for safety and security.

Others may need a different vision to suit their expertise and research. Nonetheless, it is important to develop a worthy research vision so that the research does not get diluted and the opportunities for collaboration become clear.

4.4 Science or Engineering?

The fundamental research and applied research are two different things is a flawed assumption for software engineering. Application is the opportunity to verify that the fundamentals are correct, and they need to feed on each other. Otherwise one becomes completely untethered from reality. Scientists build things to study something and engineers study something to build things. Software engineering research should be centered around building something. And, it should not be building toy systems. The challenges of software engineering lie in building, evolving, analyzing, and verifying large software. The fundamental research should

focus on these challenges. I believe and I have experienced that such fundamental research can be and will be of interest to industry.

One must also ask how much should we emphasize science in software engineering research? Many software analysis problems have exponential complexity. It is important to look for pragmatic engineering solutions to address these problems. On a related theme, Professor Kota writes [5]: "Technological innovation has long been key to American prosperity, especially when it is applied to cutting-edge manufacturing, and the centerpiece of such innovation is engineering. Investments in scientific research produce indispensable knowledge, but it is by applying that knowledge through rigorous engineering and practical development that people and nations produce wealth, thereby achieving economic strength, and national security. The United States has fallen behind in this "translational research." A core problem lies in America's failure to maintain one of its historical core strengths: engineering. Distinct from science, engineering means not just analysis and discovery but synthesis and innovation aimed at turning abstract ideas into tangible products."

Software engineering must emphasize innovation aimed at turning ideas into tangible software.

4.5 About Academic Environment

The Promotion and Tenure (P&T) criteria promote the numbers game and not the tangible evidence a professor can produce to convince industry the utility of his or her research. Without some changes in P&T process, it is risky and often not rewarding to spend time on research that is likely to lead to any major collaboration with industry. A support system at university is critical to undertake research that will be valued by industry. In my case, the industry partnership initiative at Iowa State University was critical in the beginning stage.

Some of my experiences are peculiar to the US setting. Research and teaching expectations for promotion and tenure, the support from university to promote partnership with industry, the ties university has with industry, whether the university is situated in a hub of companies with need for software professionals are factors that significantly impact the dynamics of creating university-industry partnerships.

5. CONCLUSION

While I write about my journey and my research, I have not done it alone. The research has involved an enormous amount of software development. I have been very fortunate to have a great team of graduate and undergraduate students, post-docs, technical staff, and my colleagues and highly talented engineers at EnSoft to enable our research.

I quote Parnas on software engineering, academia, and industry. Parnas comments [15]:

> Industry is aware of the need for improvement and sporadically forms new groups and initiatives that attempt to bring about change in what practitioners do. Most mainstream academics do not get involved. On the academic side, we see new notations, formalisms, proof methods, and design approaches. These gain little traction with industry because they do not appear to address the practitioner's problems. Rather than show a better, more efficient way to do things, they call for additional work that has no obvious benefit.

The industry and the open source community produce the software. The litmus test of good software engineering research should be the value it provides to producers of software. If the research papers are not important to them, if they do not apply the research, how do we justify the value of that research?

6. ACKNOWLEDGMENTS

I am greatly indebted to my colleagues at EnSoft and my research team at Iowa State University for their help. This material is based on research sponsored by DARPA under agreement numbers FA8750-12-2-0126 and FA8750-15-2-0080. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

7. REFERENCES

[1] Bill Gates Keynote: Microsoft Tech-Ed 2008. http://news.microsoft.com/speeches/bill-gates-keynote-microsoft-tech\%E2\%80\%A2ed-2008-developers/.

[2] Darpa automated program analysis for cybersecurity. https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-11-63/listing.html. Accessed: Jan. 2016.

[3] Darpa space/time analysis for cybersecurity. https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-14-60/listing.html. Accessed: Jan. 2016.

[4] Ecocar 3: An advanced vehicle technology competition. http://ecocar3.org/vthevt/team-sponsors/.

[5] Engineering 2.0: Rekindling american ingenuity. http://www.huffingtonpost.com/sridhar-kota/engineering-20-rekindling_b_4844449.html.

[6] Ensoft corp. http://www.ensoftcorp.com.

[7] Formal verification in large-scaled software: Worth to ponder. https://blog.inf.ed.ac.uk/sapm/2014/02/20/formal-verification-in-large-scaled-software-worth-to-ponder/.

[8] [video] atlas: a new way to explore software, build analysis tools. https://www.youtube.com/watch?v=cZOWlJ-IO0k.

[9] Dirk Beyer and Alexander K Petrenko. Linux driver verification. In *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, pages 1–6. Springer, 2012.

[10] Frederick P. Brooks, Jr. The computer scientist as toolsmith ii. *Commun. ACM*, 39(3):61–68, March 1996.

[11] Carlos Canal and Akram Idani. *Software Engineering and Formal Methods: SEFM 2014 Collocated Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, WS-FMDS, Grenoble, France, September 1-2, 2014, Revised Selected Papers*, volume 8938. Springer, 2015.

[12] Tom Deering, Suresh Kothari, Jeremias Sauceda, and Jon Mathews. Atlas: a new way to explore software, build analysis tools. In *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014.

[13] Benjamin Holland, Tom Deering, Suresh Kothari, Jon Mathews, and Nikhil Ranade. Security toolbox for detecting novel and sophisticated android malware. In *Proceedings of the 37th International Conference on Software Engineering*. IEEE Press, 2015.

[14] David Lorge Parnas. Stop the numbers game. *Commun. ACM*, 50(11):19–21, November 2007.

[15] David Lorge Parnas. Software engineering - missing in action: A personal perspective. *IEEE Computer*, 44(10):54–58, 2011.

[16] S.C. Kothari. Automatic parallelization, aspect-oriented programming, and beyond. In *Keynote Address, High-Performance Computing Asia Conference, Bangalore, India*, 2002.

[17] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal methods: Practice and experience. *ACM Computing Surveys (CSUR)*, 41(4):19, 2009.