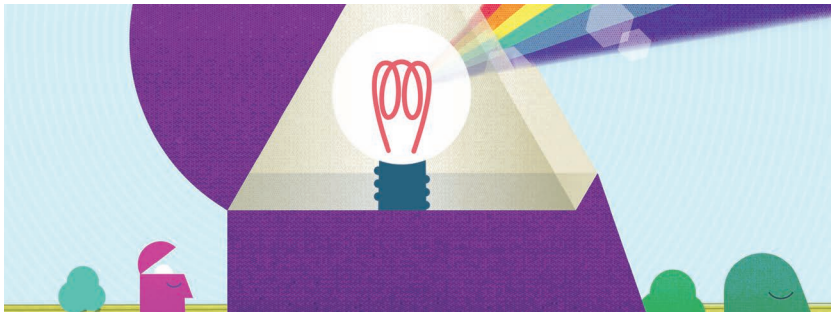
The Future Is Already Here

Jan Bosch, Chalmers University of Technology

// *In this article, we argue that the traditional view on the role of academic research in software is outdated and present an alternative approach to academic research. Finally, we exemplify our approach through a concrete case, Software Center.* //



THE TRADITIONAL VIEW on technology-driven innovation is that it follows a process starting with basic research, applied research, advanced engineering, and, finally, general engineering.[1] In this view, the assumption is that the first two stages occur in academia, whereas industry becomes involved as the technology reaches a higher level of maturity and can be employed in commercial products.

Although the traditional view may be correct for some areas of technology, unfortunately, it is fundamentally flawed in the area of software as, for the last decades, all major innovations have occurred in industry.[2] The last major innovation originating from academia, to the best of our knowledge, is Python, which was created in 1989.

Introduction

To quote William Gibson, the famous science fiction author: "The future is already here; It is just not evenly distributed." The same can be said for software, data, and artificial intelligence (AI). Some companies are creating amazing solutions using bleeding edge technologies developed in house, whereas many companies are years, if not decades, behind in the adoption of these same technologies.

Also, the reality of software development in industry has changed quite fundamentally. In the age of DevOps, systems are no longer built but rather grown, pruned and evolved in short, agile sprints based on the feedback coming back from deployments in the field.[3,4,5] Rather than building according to requirement specifications, most systems evolve using experimental methods, such as A/B testing, and engineers continuously adjust their software in response to the data coming back from the field.[6,7] Many academics unfortunately have not yet adjusted their view on software development to reflect this new reality.

Because of the two aforementioned challenges, the position we present in this article is that the role of academic research in software needs to shift from being the originator of novel ideas to the identifier, generalizer, and accelerator of new innovations occurring in industry. The role of academic, independent research is to identify novel innovations in industry that are created in highly specific contexts with the intent of solving a unique problem experienced by a company. Once identified, the role of research is then to generalize the approach, typically by studying incarnations of the same approach in multiple companies. Finally, the role of research is to accelerate the adoption of this novel but now generally applicable technology across industry and society.

Rather than operating under the assumption that all relevant new knowledge is created in academia, academic research needs to accept the reality of innovation in software and adjust its role and ways of working accordingly. In the end, academic research is to the largest extent funded by the public, and consequently, our responsibility is to spend our time and energy on the activities that will have the greatest positive benefit to society.[1] In software, this means identifying, generalizing, and accelerating the adoption of new technology.

Digital Object Identifier 10.1109/MS.2024.3358417
Date of publication 25 January 2024; date of current version 11 April 2024.

The contributions of this article are threefold. First, we discuss the traditional view of the role of academic research in software, its consequences, and why we question it. Second, we present an alternative approach to academic research in software and digital technologies that focuses on identifying, generalizing, and accelerating adoption of innovations. Third, we exemplify our approach through a concrete case, Software Center (SC), a collaboration among more than 15 large, international companies and five Swedish universities.

The remainder of the article is organized as follows: In the section "Paradigm Shift," we discuss the paradigm shift that software has seen in the last decades. Then, we explore the traditional view on software research. Subsequently, we present an alternative approach to conducting academic research in software. We present SC as a case in the section "SC," provide some examples to illustrate the center, and conclude the article.

Paradigm Shift

When Google launched Gmail in 2004, the company used the notion of perpetual beta to indicate that Gmail would not be finished as a product at any point in time but rather that it would continue to evolve and that it, in fact, would never be finished in the traditional sense of the word.

Since then, virtually all software as a service (SaaS) products use continuous deployment (or DevOps) as a way to continuously deliver value to customers. More and more industries are following suit. During the last decade, embedded systems companies have also started to offer continuous deployment of new software in their offerings. Ranging from telecommunications to cars, DevOps is increasingly becoming the norm.[8,9]

In parallel, the lean startup movement popularized the notion of the minimal viable product (MVP) as a mechanism to validate customer interest early and against low cost.[10,11] The underlying notion is that customer and user responses to a new innovation are fundamentally unknowable, and the only way to establish customer interest and, consequently, market opportunity is through experimentation.

These developments have resulted in a paradigm shift in the engineering of software-intensive systems in industry. Due to the high risk of building systems without continuous customer feedback, the initial development phase before the first deployment of the system in the field is limited as much as possible and will, over time, represent a tiny percentage of the overall R&D effort invested in the system.[12]

Once the first release of the system (MVP) is deployed and customer feedback is promising, development takes on an experimental, hypothesis-driven nature where multidisciplinary teams consisting of user experience experts, product managers, and engineers hypothesize new functionality to influence the key performance indicators the team wants to improve and run experiments, such as A/B testing, to validate these hypotheses. This process continues as long as the system is deployed and monetized. There is no end to development, although the amount of resources allocated may vary over time.

To summarize, engineering software-intensive systems is no longer about building systems according to the requirement specification as a project with a clear start and end. Instead, it is about growing and evolving these systems in the same way in which a gardener will guide the growth of a tree. Except for the initial sprouting (developing the MVP), the process is continuous, feedback driven, and evolutionary in nature.

Traditional Software Research

Research in software has stayed with the old, waterfall-style practice of development.[1,13,14] Even today, students are taught to start with a requirement specification, design a software architecture, write the code, integrate, test, and release. Also, most research publications still assume the traditional approach to software development as well, and we are stuck in a paradigm that no longer matches the reality that the software industry is experiencing.

The aforementioned development is exacerbated by the view of many academics that new technology innovations originate in academia and, over time, move from basic and applied research in academia to advanced engineering in industry. If one believes one is at the wellspring, there is little need to search downstream for water sources. Of course, there are many academic researchers collaborating with industry, illustrated, for example, by the many publications discussing DevOps. However, many still take an "industry as a laboratory" approach.

Our point is that this viewpoint is entirely inaccurate. It is the engineers in industry that have real, complex challenges to solve. These people are as intelligent and well educated as the average academic, and because they have skin in the game, it is in industry that the main breakthrough innovations in research are created.

We can see the consequences of software research being out of tune with industrial practice as for the last decades, not a single major innovation in the field has originated in academia.[2] Instead, ranging from programming languages to databases and from development methods to

processes, it is industry that is driving the progress.

In addition to industry, the open source software community has also had an enormous impact on the field in terms of tangible impact with Linux running most of the servers in the world, Eclipse still being used by close to 10% of developers, JavaScript being the most popular language among developers, etc.

The commonality between companies and the open source community is that the solutions that put a dent in the universe are created by people that have a stake in successful, high-performing solutions. In many ways, the engineers building these solutions have skin in the game. It is up to them to solve hard problems for their organization.

> Instead, ranging from programming languages to databases and from development methods to processes, it is industry that is driving the progress.

We are not the first to identify this; for instance, Wowk et al.[1] raise a similar concern related to the impact of academic research on society. Also, the Stanford report on AI research[14] highlights the rapid increase in publications originating from Fortune 500 companies.

Toward a Paradigm Shift

If we accept the premise that academic research is no longer concerned with being the wellspring of innovation, the immediate question is what its role is. In our view, academic research is still incredibly important, but it should focus on different activities, to be specific, identification, generalization, and acceleration. Below we describe each in more detail.

Identification is concerned with recognizing novel techniques and ways of working at companies. Every company is addressing a set of challenges, and engineers periodically come up with novel ways of addressing these challenges. As an academic, it is easier to interact with engineers from multiple companies and to identify patterns that capture novel ways of working.

A second pattern for identification is to recognize the differences among industries. In practice, we see that many novel technologies and ways of working originate in SaaS companies, then transition to the embedded system industry, such as automotive and telecommunications, and finally are adopted by IT companies, such as banks and insurance companies. When working with embedded systems or IT industries, it is consequently relatively easy to identify the next big developments.

The second activity is concerned with generalization. Generalization is concerned with three activities, i.e., conceptualization, increasing applicability, and validation. When having identified an innovation in some company or companies, the first step is concerned with describing the innovation independent from the specific and concrete context or contexts in which it was initially conceived. This allows us to assess where the concept could be applied without adjustments and where it cannot. For the cases where it cannot be applied without adjustment, we extend the concept with variation points and more generalized aspects in order to maximize the applicability of the new concept. Finally, the third activity is concerned with validation. As researchers, we may believe we have identified a promising innovation, but unless it is validated in one or more companies and it delivers the expected benefits, we are operating on beliefs rather than evidence.

The third activity is concerned with accelerating the adoption of the, now validated, innovation across industry. From a societal perspective it is best for innovations to be adopted as rapidly and as widely as possible. Hence, as academic researchers, working with companies to adopt the concept delivers on that societal benefit, as well as allows for the identification of limitations that need to be addressed as well as new innovations.

Of course, especially when working with acceleration of adoption of an innovation, one may easily be accused of acting as a consultant. However, in practice, the goal for an academic researcher is always gaining additional knowledge. In this case, new knowledge can be concerned with the process of adoption, the limitations of the innovation, or the identification of new innovations triggered by the adoption of the validated one.

In practice, many software researchers are already working in the way outlined in this section. However, the prevalent paradigm in research does not necessarily recognize

these activities as real research as it does not match the underlying assumptions of the traditional paradigm. Hence, we need a paradigm shift in software research in order to align the paradigm with the reality that many of us are experiencing in our day-to-day work.

SC

To illustrate the emerging paradigm of software research, we introduce SC. SC was formed in 2011 in response to the realization by several large companies in the Gothenburg region in Sweden, specifically, Ericsson, Volvo Cars Corporation, AB Volvo, and Saab Defense, that software was becoming increasingly important for their products and offerings to their customers. SC was formed at Chalmers University of Technology and Gothenburg University with the aforementioned companies as founding members.

SC is funded by the member organizations and does not rely on public funding, which has proven an important aspect for its longevity as many centers struggle to survive the end of a publicly funded project. Since 2011, SC has grown to more than 15 companies and five universities. It started with three research projects, mostly concerned with testing, architecture, and metrics, and five researchers. In its 12th year of existence, it now has over 20 research projects, ranging from software engineering to AI engineering and from product management to business model innovation, and involves over 30 researchers.[15]

Several of the member companies are competitors in certain business areas. We have addressed this by ensuring that all intellectual property created in SC is available to all partners, encouraging companies to, where possible, bring in not the business units that are in the strongest head-to-head competition but rather less sensitive units and finally by focusing more on the "hows" of software development, data-driven practices, and AI engineering solutions rather than the "what" (e.g., specific features, A/B experiments, or ML models). This allows competitors to peacefully coexist in the center.

Although initially implicit, SC employs the paradigm described in the section "Toward a Paradigm Shift" with identification, generalization, and acceleration at the core of its operations. However, successfully employing this model is challenging, and to address this we put a number of mechanisms in place, including sprints, the golden rule, prioritizing collaboration, evolution, and constant feedback.

Research in SC takes place in six-month sprints. Each project has a long-term vision and set of desired outcomes, but every six months a decision is made concerning the continuation of the project. As a consequence, each project is intended to deliver something of value every sprint. The value can include problem analysis, solution development, or validation and does not have to cover an entire research cycle. However, projects that fail to do so can see their funding cut or removed entirely. In addition, projects are evaluated both for academic excellence, typically through publications, and for industrial relevance.

One consequence is that Ph.D. students, some of the primary recipients of funding in many academic contexts, are funded to a very limited extent as it is so difficult to manage situations where a project is stopped before the Ph.D. student has finished their thesis and has graduated. SC funds one Ph.D. student per university but explicitly maintains the right to stop the project where the Ph.D. student is employed. The majority of funding is used to fund more senior researchers such as assistant, associate, and full professors.

> In our view, academic research is still incredibly important, but it should focus on different activities, to be specific, identification, generalization, and acceleration.

The second mechanism is concerned with "the golden rule," which is often translated as "he or she with the gold rules." Every six months, a group of company representatives, typically technology experts, facilitated by the center director, prioritize the research projects and recommend the projects to fund, cut, and end. The recommendation is then taken to the steering committee, which consists of representatives from all member groups, including academic partners, and typically approved without adjustments.

An important consequence is that researchers of funded projects know that they need to work with the

member companies as part of their research in order to build relationships and a deep understanding of the realities in the member companies. The advantage for researchers is that they conduct empirically founded, longitudinal research that captures valuable results not just for the involved companies but for industry and society at large.

The third mechanism is that projects are, in part, prioritized based on the number of involved companies. The more companies support and are involved in the project, the more likely the project is to be prioritized. This results in practitioners from different companies meeting each other in project activities with significant knowledge exchange among member companies as a consequence, with the research project simply providing the meeting ground.

There is significant value for researchers as well in that research is conducted in multiple companies in parallel, allowing for significant triangulation of findings as well as vastly improved validation of project results.

The fourth mechanism is concerned with evolution. The center leadership constantly scouts for new needs from the companies. Since the founding of the center, we have added research on data-driven development, including A/B testing and other experimental approaches, software ecosystems, AI and AI engineering, product management, systems engineering, safety certification, and cybersecurity. The research agenda is driven not by the interests of the individual researchers but rather by the needs of the companies.

Although most research activities are conducted with member companies, when a new topic is identified and there are companies that have already adopted and are working with it, we may seek to engage with these companies in order to learn from them how to operationalize. For example, data- and experiment-driven practices such as A/B testing are the norm in SaaS companies. By engaging with several of these companies, we learned about their practices and were then able to start on translating this to the reality of the member companies.

By engaging with several of these companies, we learned about their practices and were then able to start on translating this to the reality of the member companies.

Although one might claim that the aforementioned is not research, many fail to appreciate the challenge of bringing a topic from one industry to another. For example, A/B testing in SaaS companies is conducted in the servers of one's own data center, and the cost of running experiments is very low. In, for example, an automotive company, the vehicles are owned by customers and might be anywhere on the planet. In addition, many functions in a vehicle are safety critical, and consequently, any A/B experiment concerning that functionality needs to be safety certified, significantly increasing the cost of experiments. Because of this, the B variant in an A/B test often involves a very small number of vehicles (often employee or test vehicles), requiring advanced statistical approaches to be able to draw conclusions in a highly unbalanced A and B group setting.

Finally, continuous feedback is a key mechanism to ensure that our research activities and other initiatives in SC are aligned with the priorities of the companies. We employ several techniques to accomplish this. We conduct a survey for every sprint. We periodically meet with the SC representatives of every company. The leadership of the center, the theme leaders, act as account managers for the member companies. In addition, every meeting with one of the member companies includes forms of informal feedback that allows us to continuously confirm that we are addressing the key priorities of the companies.

One valuable approach is the engagement in research by individuals at the companies. For example, some of the senior researchers work with individuals at the member companies that decided to pursue their Ph.D. degrees in parallel to their positions at their respective employers. These individuals provide invaluable bridges between the academic research and the industrial practice and, among other benefits, also provide continuous feedback to the center.

In the section "Toward a Paradigm Shift," we claim that a new paradigm for software research is required and that many researchers are, in fact, already working in this new paradigm. In this new paradigm, software research is concerned with three main activities, i.e., identification, generalization, and acceleration. SC embodies

this new paradigm, and we can identify that many of the mechanisms described previously as well as others are concerned with one of these activities:

- *Identification*: Through the interaction with member companies, observing the industry at large as well as through our involvement in the research community, we are continuously scouting for new topics that are or are likely to become critically important for the SC company members. The mechanisms of evolution and continuous feedback allow us to facilitate the identification of new topics. In addition, the sprint mechanism allows us to rapidly respond to new topics by initiating new research projects.
- *Generalization*: Frequently, there are already some small, scattered initiatives around new topics at several of the member companies. Our research activities and close connections with the companies allow us to study these initiatives in order to understand the commonalities and variations among these companies. In addition, our interactions with nonmember companies that have already adopted and deployed a new topic allow us to learn about the specifics in deploying a new technology or way of working in the organization. The collaboration mechanism as well as the golden rule and sprint mechanism allow us to generalize the key aspects of novel technologies and ways of working.
- *Acceleration*: In line with William Gibson's "the future is already here" principle, much of the work in SC is to help accelerate the adoption of new technologies and ways of working in the member companies and to study the inhibitors and other challenges. The earlier example concerning A/B testing is a good illustration of this as the basic principle of A/B testing is trivial, and yet it is incredibly difficult to realize in most companies for a variety of reasons, ranging from existing work practices to lacking infrastructure, as well as, in fact, companies often being unclear on what to optimize for. The collaboration mechanism allows companies to accelerate by learning not only from the researchers but also from each other. The golden rule mechanism ensures that we only focus energy on topics for which there is support within the companies. The sprint mechanism allows us to take frequent, small steps toward the goal while facilitating frequent course corrections where necessary.

Illustrating the Model

To illustrate the model of identification, generalization, and acceleration, we share some examples from SC. First, one of the companies in the center worked for years with one of the researchers to develop a very advanced metrics system to instrument software development in the organization. Other companies in the center then requested the researcher to work with them on building a similar metrics system. In this case, instead of years, one of the companies got the metrics system in place in three months.

As a second example, in our work with SaaS companies, we identified the importance of A/B testing as a mechanism for determining the impact of new functionality in complex system contexts. However, as several of the companies in SC are subject to safety regulation, we needed to develop techniques to allow for the use of A/B testing in these contexts, specifically, for managing unbalanced A and B groups. Because of this work, several companies in the center are now using A/B testing that likely would not have used this for many years to come otherwise.

We run a yearly survey to ask as many company participants as we can to assess SC from the perspective of accelerating the adoption of new functionality. Although we still have ample room for improvement, the perception of the companies is that we are indeed helping significantly in the acceleration of adoption of digitalization practices.

The traditional research paradigm in software states that research starts as basic and applied research in academia and then trickles down into advanced engineering and basic engineering in industry. Consequently, many researchers view themselves as working on the wellspring of innovation in software and feel little need to interact with industry as these are viewed as downstream from the "real" research and consequently irrelevant.

This view on research in software is outdated and out of touch with reality. For the last decades, not a single major innovation in software has originated in academia. Instead, industry has taken over the role of being the key innovator, and numerous wonderful innovations in technology and ways of working originate from brilliant companies around the globe.

We are in the midst of a paradigm shift in software research where the role of academic research is not to

ABOUT THE AUTHOR

JAN BOSCH is a professor at the Chalmers University of Technology, SE42196 Gothenburg, Sweden and the director of the Software Center. His research interests include AI engineering, data-driven practices, software architecture, and digitalization. Bosch received his Ph.D. from Lund University. He is a Member of IEEE. Contact him at jan.bosch@chalmers.se.

come up with new ideas. Instead, the task of academic research is to identify new innovations occurring in industry, generalize and conceptualize these new innovations, and then accelerate the adoption of these innovations across industry and society.

We introduced SC as a case study and illustrative example operationalizing and embodying this new paradigm. The success of the center is shown by the constantly growing set of member companies. To accomplish this, SC employs several mechanisms that allow us to deliver on our mission including sprints, the golden rule, prioritizing collaboration, evolution, and continuous feedback. Through these mechanisms, the researchers in SC employ the new software research paradigm of identification, generalization, and acceleration.

In the future, we aim to involve more companies and universities in the center to grow the scope of our impact. In addition, we aim to build ways to quantify the impact of our research in terms of successfully identifying, generalizing, and accelerating innovations in software.

Acknowledgments

This research was supported by SC. I thank Helena Holmström Olsson for her feedback and help in the preparation of this article.

References

1. D. A. Wheeler. "The most important software innovations." Wheeler. [Online]. Available: https://dwheeler.com/innovation/innovation.html
2. C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, May/Jun. 2016, doi: 10.1109/MS.2016.68.
3. F. M. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," *J. Softw., Evol. Process*, vol. 29, no. 6, 2017, Art. no. e1885, doi: 10.1002/smr.1885.
4. T. Blüher, D. Maelzer, J. Harrendorf, and R. Stark, "DevOps for manufacturing systems: Speeding up software development," *Proc. Des. Soc.*, vol. 3, pp. 1475–1484, Jun. 2023, doi:10.1017/pds.2023.148.
5. A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. Holmström Olsson, and J. Bosch, "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies," *J. Softw., Evol. Process*, vol. 30, no. 12, 2018, Art. no. e2113, doi: 10.1002/smr.2113.
6. R. Kohavi and S. Thomke, "The surprising power of online experiments," *Harvard Bus. Rev.*, vol. 95, no. 5, pp. 74–82, 2017.
7. A. Dakkak, J. Bosch, H. H. Olsson, and D. I. Mattos, "Continuous deployment in software-intensive system-of-systems," *Inf. Softw. Technol.*, vol. 159, Jul. 2023, Art. no. 107200, doi: 10.1016/j.infsof.2023.107200.
8. L. E. Lwakatare et al., "DevOps in practice: A multiple case study of five companies," *Inf. Softw. Technol.*, vol. 114, pp. 217–230, Oct. 2019, doi: 10.1016/j.infsof.2019.06.010.
9. E. Reis, *The Lean Startup*, vol. 27. New York, NY, USA: Crown Business, 2011, pp. 2016–2020.
10. T. R. Eisenmann, E. Ries, and S. Dillard. "Hypothesis-driven entrepreneurship: The lean startup." SSRN. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2037237
11. H. H. Olsson and J. Bosch, "From opinions to data-driven software R&D: A multi-case study on how to close the 'open loop' problem," in *Proc. 40th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, Aug. 2014. pp. 9–16, doi: 10.1109/SEAA.2014.75.
12. T. Gilb, "Evolutionary Delivery versus the 'waterfall model'," *ACM SIGSOFT Softw. Eng. Notes*, vol. 10, no. 3, pp. 49–61, 1985, doi: 10.1145/1012483.1012490.
13. J. Bosch, J. Carlson, H. H. Olsson, K. Sandahl, and M. Staron, Eds., *Accelerating Digital Transformation: 10 Years of Software Center*, vol. 13. Cham, Switzerland: Springer Nature. 2022.
14. K. Wowk et al., "Evolving academic culture to meet societal needs," *Palgrave Commun.*, vol. 3, no. 1, pp. 1–7, 2017, doi: 10.1057/s41599-017-0040-1.
15. "SQ8. What should the roles of academia and industry be, respectively, in the development and deployment of AI technologies and the study of the impacts of AI?" Stanford Univ., Stanford, CA, USA, 2021. [Online]. Available: https://ai100.stanford.edu/gathering-strength-gathering-storms-one-hundred-year-study-artificial-intelligence-ai100-2021-1/sq8