OurCode: Experiences Transitioning University Research into a Developer Tools Startup

Consuelo López
consuelo@ourcode.io
OurCode Inc.
Buenos Aires, Argentina

Sahar Mehrpour*
smehrpou@gmu.edu
George Mason University
Fairfax, Virginia, USA

Austin Z. Henley
azhenley@cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Thomas D. LaToza*
tlatoza@gmu.edu
George Mason University
Fairfax, Virginia, USA

Abstract

Commercializing a university research project by creating a company offers a direct path for software engineering researchers based in a university to put their research directly into the hands of practicing software engineers. In this paper, we offer insight and reflections on tech transfer through the context of our journey with OurCode, a developer tools startup aimed at rethinking how developers interact with software documentation. We detail our experience shifting from a project focused on basic research into a business mindset focused on meeting customer needs. Our lessons underscore the importance of balancing technical innovation with business needs, leveraging mentorship for strategic guidance, and fostering community-driven approaches to branding and user engagement. We report on the process of customer discovery and lessons learned about onboarding and software documentation practices and how engineering leaders make choices around the adoption of new developers tools. We conclude with reflections on the challenges and value of tech transfer in software engineering, including what researchers can learn from failed transitions to align their innovations with real-world needs and startup dynamics.

CCS Concepts

• **Software and its engineering → Software creation and management**.

Keywords

Tech transfer, developer tools, startup journey, customer discovery, academic commercialization, developer experience

ACM Reference Format:
Consuelo López, Sahar Mehrpour, Austin Z. Henley, and Thomas D. LaToza. 2025. OurCode: Experiences Transitioning University Research into a Developer Tools Startup. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3696630.3728554

*Also affiliated with OurCode Inc.

FSE Companion '25, June 23–28, 2025, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3728554

1 Introduction

Software engineering aspires to improve the everyday practice of building software through better tools and practices. The process of moving basic research ideas into practice through tech transfer is central to the identity of software engineering as a discipline. There are many tech transfer pathways, most commonly through industrial research teams embedded in companies.

Working inside a company, while also participating in the software engineering research community, industrial researchers bridge the gap between research and practice by applying research in a real world setting. However, for university researchers, this pathway to tech transfer can be challenging. In particular, industrial research projects are initiated and managed to support the larger company context. For a university researcher looking to quickly transition their research innovations into practice, finding an industrial research team whose own goals closely align with the university research and who have the interest and bandwidth to pursue the work can become an insurmountable barrier to even begin.

An alternative pathway for university researchers to pursue tech transfer and put new tools and practices into the hands of engineers is to found a developer tools startup. In this pathway, researchers at a university form a company, build a team, and create a product or service that can be sold to customers. This pathway offers a direct and immediate way for university researchers to move basic research in developer tools into practice.

In this paper, we report on our experiences commercializing basic research in software engineering by founding and building a developer tools startup. Our aim is to offer university researchers a starting point for understanding the challenges they will face in commercializing research, some of the approaches available, and our hard-earned lessons from applying these in practice. While our aim is to shed light on the overall pathway of commercialization through a startup, we focus most closely on the challenges our company faced. The approaches we describe may be specific to the United States, where our company was founded.

Figure 1 describes the overall timeline of our project, encompassing both the university research and the activities to create the startup. Our project began with the creation of a new technique

to keep software documentation up to date and more tightly integrated with code, ActiveDocumentation [12]. This project yielded several subsequent research publications, describing a new technique to easily create active documentation [13] and an empirical study demonstrating that many of the knowledge transfer issues developers face can be addressed through active documentation [11]. Following the completion of user studies which demonstrated that the approach could dramatically speed developers' work with unfamiliar code [12, 13], we began to explore commercialization through a startup. We participated in a program to help university researchers create a startup, the NSF Regional I-Corps, and then officially founded a new company, OurCode Inc., structured as a Delaware C Corp. We applied for funding, partnered with startup advisors, and began developing a website, branding, and social media strategy to drive awareness of our startup. To understand if this project could lead to a successful and viable business, we participated in the NSF National I-Corps program, where we conducted interviews with over 100 software engineers, engineering leads, engineering managers, and CTOs and examined if there was a need for our product. In December 2024, we officially shutdown our business.

In the remainder of this paper, we share our experiences and lessons learned commercializing software engineering research, beginning by briefly reviewing our initial research project. We then describe the pathway to commercialization we pursued, focusing on some of the key challenges any developer tool startup will face including funding, intellectual property, branding and marketing, and utilizing mentoring resources. We then examine the central focus of a startup in detail, customer discovery and achieving product market fit, and our experiences with customer discovery. We reflect on the most pressing challenges our company faced and conclude with lessons that go beyond our experience, offering guidance for researchers pursuing tech transfer and contributing to broader conversations about how academia can better align with the practical needs of the software industry.

Figure 1: Timeline showing our milestones as we transitioned from research to startup.

2 Related Work: Research Tech Transfer

Since the 1980s, academic institutions have increasingly prioritized the commercialization of research to diversify their revenue sources and to grow beyond academic impact [10, 18, 20]. This has led to the establishment of dedicated technology transfer offices, which manage the licensing of intellectual property (IP) generated through research. These offices often facilitate patent applications, negotiate licensing agreements, and build connections with potential industry partners. Beyond IP licensing, universities have developed on-campus incubators and accelerators that provide mentorship, funding, and access to resources needed to develop and scale startups (e.g., George Mason University's Mason Enterprise Center[1]). Many institutions also support faculty and researchers in forming their own companies with leave policies for company formation, entrepreneurial training, connections to venture capital networks, and early-stage funding.

In their seminal 1985 paper, Redwine and Riddle examined various software technologies to understand their development and dissemination [16]. They found that it typically takes 15 to 20 years for a technology to mature from initial concept to widespread adoption, involving six distinct phases: (1) Basic research, where ideas and concepts are explored and key research questions are established. (2) Concept formulation, which involves the informal sharing of ideas, building a research community, and publishing solutions to specific subproblems. (3) Development and extension, during which the technology begins to be used, underlying ideas are clarified, and the approach is broadened. (4) Internal enhancement and exploration, where the technology is applied to new domains, real-world problems, stabilized, and supported with training materials to demonstrate its value. (5) External enhancement and exploration, which engages a broader community beyond the original developers to provide substantial evidence of the technology's effectiveness and applicability. (6) Popularization, the final stage where high-quality, supported versions of the technology are produced, commercialized, marketed, and the user base is expanded for mainstream adoption. Similarly, the book, *Diffusion of Innovations*, describes the broader process of diffusion as having an innovation communicated through channels, over time, among people in a social system [17]. Both of these frameworks together offer a comprehensive roadmap for understanding the lifecycle of software technologies, which may be particularly apt for transitioning software engineering research into products.

In the software engineering research community, there have been numerous examples of academics who have started companies to commercialize their research. Semmle was a company that

[1] https://enterprise.gmu.edu/

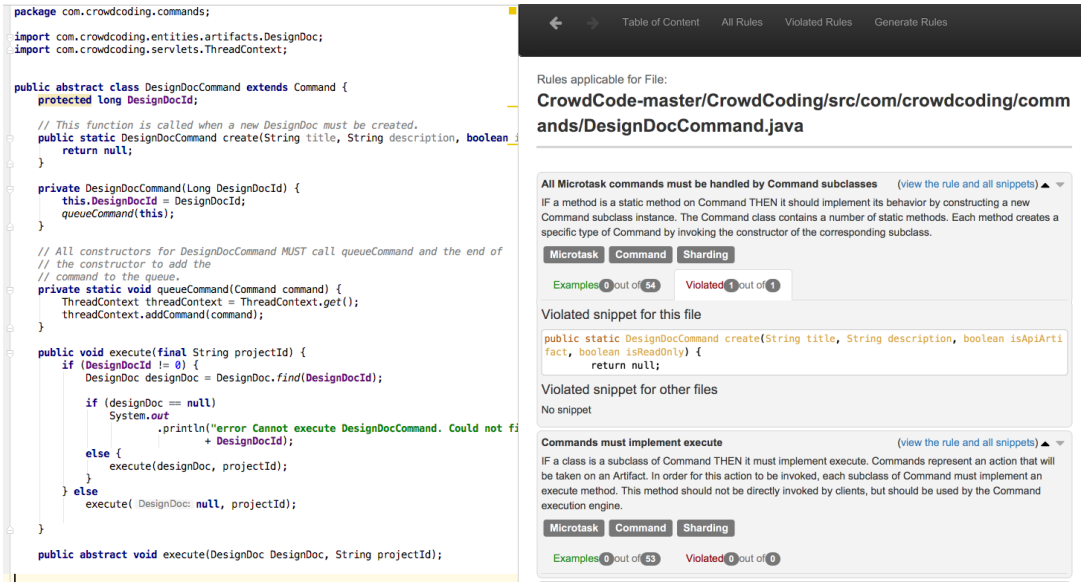

Figure 2: ActiveDocumentation is a plugin (right) for an IDE (left) that checks documented design decisions (panels) against code. As code changes, it continuously checks the documentation against the code at compile time, highlighting relevant sections of the documentation and violations of design decisions in the documentation immediately as they occur.

built a code analysis platform that originated at University of Oxford and acquired by GitHub in 2019 [7]. Monoidics, a spin-off from research at Imperial College London, developed a formal verification tool, Infer [2], and was acquired by Facebook in 2013 [9]. GrammaTech, from Cornell University, has been developing program analysis tools since 1988. Coverity, a spin-off from research at Stanford University, developed a static analysis tool for C and C++ programs and was acquired by Synopsys in 2014 [21]. Majicke built the bug finding tool, Sapienz, at University College London and was acquired by Facebook shortly after [19]. Akita developed a tool for monitoring API traffic and was acquired by Postman in 2023 [15]. Tasktop, started by University of British Columbia researchers, developed an integration platform for software development tools and was acquired by Planview in 2022 [14].

One of the most detailed experience reports of commercializing academic research is by Chilana, Ko, and Wobbrock [3]. Beginning as a research project at the University of Washington, LemonAid [4] was embedded in websites to help users find answers to their questions. After several evaluations of the system, including field deployments of LemonAid running publicly [5], they believed there was little incentive to continue the evaluations, and instead decided to commercialize LemonAid. They raised $50,000 from their university's commercialization gap fund, which they used to get customers, develop the product, and find investors. Then they raised a round of venture capital to build a team and get the product to market. Their experience report elaborates on the challenges of what needs to be added to their research prototype to be a scalable product, who their target customer is, and the value proposition to their customers [3]. We share our experience, process, and lessons learned in the remainder of this paper.

3 Basic Research

Our research project began with several empirical studies examining the difficulties software engineers encounter when working with unfamiliar code. In 2005, while studying engineers at Microsoft, we conducted a series of interviews and surveys that revealed that existing documentation practices were inadequate, with developers often distrusting and ignoring the design documents they produced [8]. Instead, developers relied heavily on implicit knowledge to understand and edit code, which created challenges onboarding and sharing knowledge across teams. This suggested a significant opportunity: tools which more effectively linked code to design documents could bridge this gap and improve developer workflows.

Building on these insights, we created new forms of static analysis tools that effectively integrate documentation and code. We created ActiveDocumentation, which verifies documented design decisions against code. Each time a developer edits a file in their IDE, it actively checks the updated code against the documentation, immediately surfacing relevant design decisions in the documentation, linking to examples of code following the design decision, and highlighting design decision violations [12] (Figure 2). A user study with 18 participants showed that, compared to traditional documentation, ActiveDocumentation significantly improved developers' ability to work with unfamiliar code faster and more successfully.

To examine the type of static analysis that might be required to check design documents against code, we conducted an empirical study of 1,323 defects found during code reviews. This study demonstrated that static analysis tools could, in principle, detect up to 76% of the defects found in code review [11]. Many of these defects reflected project-specific rules of the sort which are commonly described in design documents. Among static analysis tools, AST Pattern Checkers showed the broadest coverage, with the potential

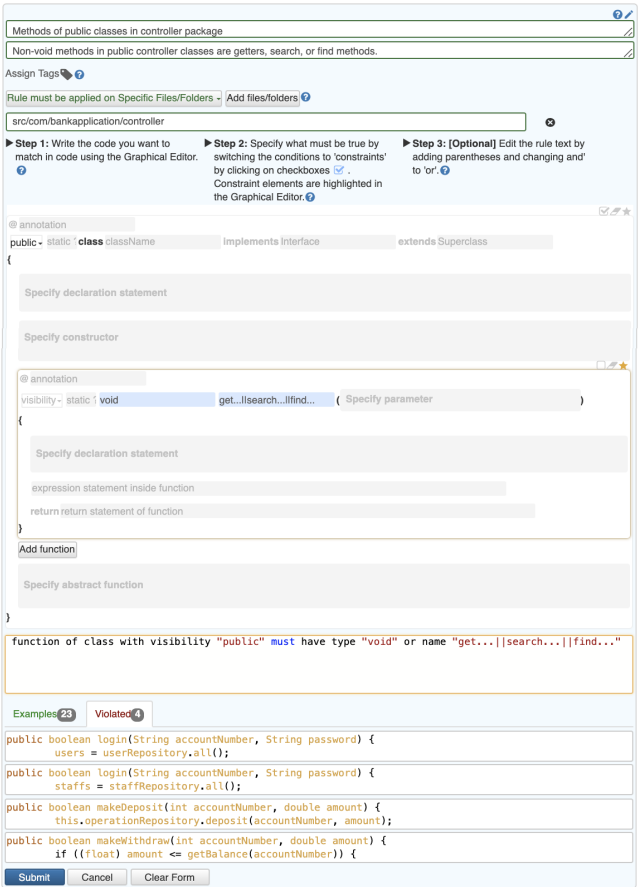

Figure 3: A developer using RulePad creates design decisions by specifying properties and refining rules in the Graphical Editor (top) with structured code snippets. They can verify and optionally edit rules in the Textual Editor (bottom) while viewing examples of code that satisfy or violate the rule.

to detect 25% of all code review defects. This study underscored that documentation tools not only support documenting and understanding design decisions but also play a critical role in reducing defects, making them a key resource for enhancing software quality.

We developed RulePad to help developers create design documents in a checkable format compatible with ActiveDocumentation [13] (Figure 3). Leveraging a novel structured editor and semi-natural language, RulePad supports documenting design decisions as AST patterns. In a user study with 14 participants, we compared RulePad to the PMD Designer, a utility for writing rules in a popular rule checker. Participants using RulePad successfully authored 13 times more query elements in significantly less time and reported greater willingness to adopt RulePad in their everyday work to capture their design decisions.

Finally, we created DesignRuleMiner to, with step by step support from the developer, identify and extract design rules directly from the code. By leveraging information from the codebase and the developer's work context, DesignRuleMiner suggests relevant and actionable design rules. In a user study with 16 participants working in an unfamiliar codebase, we found that it enabled developers to work 1.42 times faster and produce code that was 1.36 times more accurate. Participants emphasized that the inclusion of example code snippets illustrating each design rule was crucial for understanding and implementing the rules effectively.

4 Building a Company

Transitioning a university research project into a successful company requires addressing a number of key challenges. These include identifying funding and ownership of intellectual property, building awareness in the company through a branding and marketing strategy, finding and working with mentors to provide guidance and direction, and building an engineering team to build the product.

4.1 Funding and Intellectual Property

One of the first key questions is who will fund the new company. Building a company requires substantial time—to interact with potential customers and understand the market needs, build a minimal viable product (MVP), find and gather feedback from potential customers, and iterate the design of the MVP. We considered three possible funding mechanisms. Our state government as well as the US government offer non-dilutive grant funding, which provide funds for the startup without taking any equity (ownership) in the resulting startup. Venture capital firms and angel investors focused on developer tools, such as BoldStart[2] and Devtool Angels[3], offer dilutive investments, which provide funds in return for equity (ownership) of the startup. Bootstrapping involves the founders themselves funding the startup, either by working a second job or through their own personal savings.

A closely related issue is intellectual property. For potential investors, a key question is who owns the intellectual property (IP) being commercialized. For a potential developer tools company, the key IP is often the core idea, the research innovation that enables the product, as well as an early stage prototype. In our case, as we had previously published all of the key innovations, they were no longer eligible to be patented and were all in the public domain. The code for our prototype was distributed under a permissive open source license. While we discussed our plans for commercializing our research with our university's office of tech transfer, they ultimately agreed that the university did not own either of these and that our startup was largely free to commercialize this work without sharing ownership with the university.

We applied for a state program (VIPC CCF) which offered non-dilutive funding and promised a short and simple application process. After discussing the program's aims with a funding officer, it seemed our project was an ideal fit. However, changes to the program leadership resulted in changed expectations. We received feedback that our commercialization effort was not yet mature enough, as it lacked a sufficiently clear go to market strategy for reaching potential customers. We did receive a small amount of non-dilutive funding through a second program which was more focused on supporting early commercialization efforts, the NSF National I-Corps.

[2]https://boldstart.vc/
[3]https://www.devtoolangels.com/

We considered at length applying for dilutive funding. However, this came with substantial and meaningful strings. With their preferred equity, taking funds from a VC would mean relinquishing substantial leadership autonomy, with the VC able to dictate the direction and pace of our commercialization. As our goals were to build an independent developer tool company, rather than facilitate a future sale of our company, we decided against this.

4.2 Branding and Marketing

Branding and marketing are central to communicate a company's vision, connect with potential customers, and establish credibility in a competitive market. In software engineering, branding goes beyond a logo or tagline—it is about crafting a consistent narrative that aligns with the company's values, mission, and the problems it seeks to solve. Marketing, in turn, translates this narrative into actions that build awareness, foster trust, and engage stakeholders.

As a developer tools startup, we recognized that we needed to create a brand that resonated with developers and decision-makers alike. The tension lay in balancing technical credibility with accessibility. Our audience ranged from deeply technical engineers to engineering managers and decision-makers, each with unique concerns. Branding and marketing became tools to bridge this gap, crafting a unified identity that spoke to diverse stakeholders while maintaining authenticity.

Our branding journey began with identifying our core values, collaboration, transparency, and simplicity, and ensuring that they informed every decision. However, with limited resources and no in-house design expertise, we faced the challenge of translating these abstract values into a visual identity. Without in-house design expertise, we used Fiverr to develop the logo. This pushed us to clarify our brand values and resulted in a design that conveyed simplicity and connection. Relying on Fiverr meant we had less direct control over the creative process, and aligning external design input with our vision required constant communication. Yet, it also pushed us to articulate our brand values more clearly, ultimately strengthening the final outcome.

An integral aspect of OurCode's branding strategy was the design and implementation of a website that clearly communicated our mission and engaged our target audience. The website served as a central hub for information, reflecting our core values of collaboration, transparency, and simplicity. Beyond showcasing our project's purpose, it was designed to foster interaction, build a sense of community, and encourage ongoing engagement.

Equally important was the decision on the website's Call-to-Action (CTA). Guided by discussions with mentors and internal brainstorming, we considered various options: "Join a Demo" to showcase the product in action; "Give Feedback" to gather user insights and refine our offering; "Follow Us on LinkedIn" to grow our social media presence and network; "Sign Up for Updates" to build a mailing list for ongoing communication. Ultimately, we chose "Sign Up for Updates" as the primary CTA. This reflected our commitment to transparency and community-building, providing visitors with a simple way to stay informed about our progress and future developments.

Marketing OurCode was shaped by experimentation and learning. We chose LinkedIn as our primary channel, recognizing its strength in reaching technical decision-makers and engineering leaders. We created content that served several different aims.

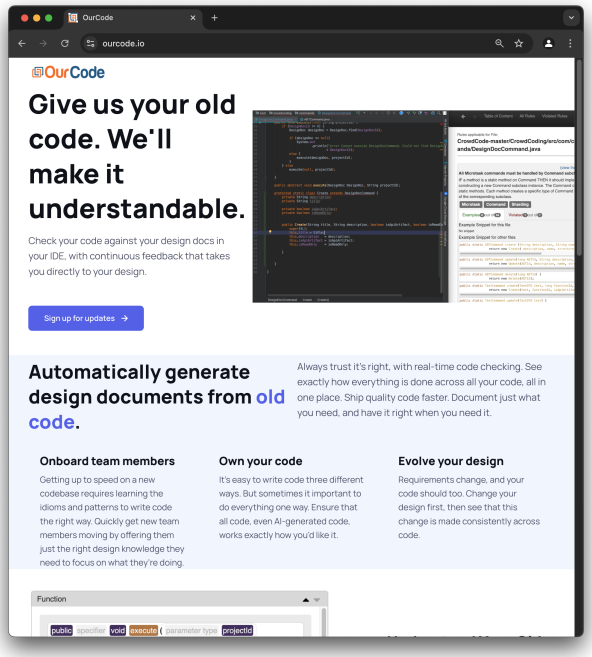

Figure 4: The OurCode.io website, providing a concise statement of the value proposition, a call to action ("Sign up for updates"), and screencasts demonstrating the tool.

4.2.1 Educational Content. Posts addressing developer pain points, such as onboarding challenges and documentation practices, sparked meaningful discussions and positioned us as thought leaders. The primary goal of our posts was twofold: to engage and build a community while providing actionable reminders and simple tools to promote industry best practices. By fostering discussions and offering practical insights, we reinforced value to the audience and also aligned with our broader mission of empowering developers to work more effectively and collaboratively. To make the content more engaging, we introduced humor and referenced academic papers foundational to our work, ensuring credibility and relevance. We leveraged tools such as ChatGPT to draft initial posts, enabling us to iterate quickly and maintain a consistent posting cadence. A structured approach allowed us to balance accessibility with technical depth, making complex topics more relatable.

4.2.2 Transparency and Updates. Sharing milestones, such as progress in the National I-Corps program, helped build trust and keep our audience engaged. This transparency aligned with our value of openness and also worked to build both our company brand and personal brand, as suggested by our mentors. Highlighting these milestones reinforced our credibility, demonstrating progress and commitment to solving real developer challenges. However, we occasionally struggled with how much detail to include—too little felt vague, while too much risked overwhelming the audience. Striking the right balance became a key focus.

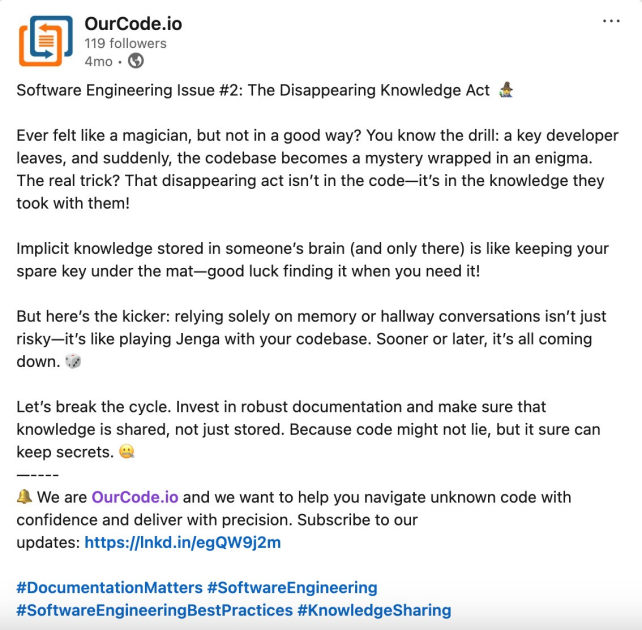

Figure 5: Example of a LinkedIn educational post.

4.2.3 Interactive Engagement. Instead of directly asking for interviews or feedback, we used open-ended questions like "Who should I talk to?" or "What's your team's biggest challenge with documentation?" This approach worked because it leveraged the power of personal referrals and fostered organic growth. However, it required patience, as building trust and getting responses took time.

Our branding and marketing efforts increased visibility, strengthened relationships with our target audience, and provided valuable feedback that informed both our messaging and product development. The process also clarified our values and vision, helping us communicate our project and goals more effectively while establishing guiding principles for the entire process. Throughout, we could quantify simple success metrics around engagement, in terms of views or likes or the number of interview participants found through these channels, but could not easily quantify the impact on our thinking about the problem.

4.3 Mentorship Resources

Mentorship provided critical guidance and encouragement, bridged gaps in knowledge, challenged assumptions, and fostered growth through shared expertise. For OurCode, mentorship came through a combination of structured programs, informal conversations, and networking, each contributing uniquely to our development.

Building a startup is as much about learning as it is about execution. While our team had strong academic and technical expertise, we lacked experience in business development. We sought mentorship because we recognized the importance of having someone who could not only guide us but also point out blind spots we might otherwise miss. Mentorship provided us with external perspectives, expanded our network, and gave us access to individuals who had navigated similar paths. By learning from their experiences, we avoided common pitfalls, identified new opportunities, and grew more confident in our decisions. Our participation in formal mentorship initiatives such as VMT, National I-Corps, and Endeavor provided targeted support that went beyond expectations.

4.3.1 The Venture Mentoring Team Organization (VMT). We initially joined VMT looking for a single mentor to support us as we prepared to begin the NSF National I-Corps program. We found an entire group of mentors who became a consistent source of guidance and encouragement. Our first interaction with the group was a pivotal moment: we presented a pitch to convince them why they should invest their voluntary time to help us. This milestone not only shaped our mindset as entrepreneurs (forcing us to articulate our vision and goals clearly) but also laid the foundation for a deep, trust-based relationship. Over the course of a year, we had monthly calls to review updates, address obstacles, and explore new opportunities. Beyond their practical advice and connections to valuable resources, this group provided emotional support, consistently demonstrating trust in us and our potential. Their belief in our work helped sustain our motivation through challenges. The program culminated in a meaningful closure call, reflecting on our progress and fostering a lasting relationship that continues.

4.3.2 National I-Corps Program. While the structured classes in the I-Corps program were invaluable, the biggest benefit was the opportunity to schedule 1:1 calls with experts. These were not academic mentors but engineering leaders or entrepreneurs who had built their own businesses and could offer firsthand insights. Their advice extended beyond customer discovery, helping us understand the practical realities of scaling a startup and making strategic decisions under uncertainty. Additionally, at the beginning of each group encounter, we had dedicated time to present our progress and receive targeted feedback. This iterative process helped us refine our approach, incorporating inputs from mentors and peers to improve continuously. The combination of structured guidance, personalized mentorship, and actionable feedback made the program a transformative experience for our team.

4.3.3 Endeavor. Consuelo participated in the Women in Entrepreneurship program by Endeavor, which offered a tailored approach to mentorship. Through weekly classes, 1:1 mentorship sessions with local entrepreneurs, and a network of female founders, the program provided inspiration and practical tools for navigating the complexities of running a startup. The exposure to diverse perspectives and experiences among female founders was particularly empowering, reinforcing the importance of resilience and innovation.

Beyond formal programs, informal mentorship and networking played a pivotal role in our journey. Conversations with colleagues, peers, and industry connections often provided unexpected insights. For example, discussions at industry meetups or introductions through LinkedIn expanded our perspective and reinforced the value of community-driven learning. The mentorship we received shaped every aspect of OurCode, from product development to marketing and branding strategies. It enabled us to validate ideas and assumptions, gain clarity on critical decisions through external perspectives, and build confidence and resilience by learning from the successes and failures of others.

4.4 Building a Team

Building the OurCode team was as much about fostering collaboration as it was about finding the right people. Our team grew through the addition of six interns, all of whom were recent graduates from the same institution as two of our core team members. They were eager to gain real-world experience while navigating the transition to full-time engineering roles. This dynamic introduced both opportunities and challenges, as we worked to align individual aspirations with the collective mission of OurCode. From the outset, we emphasized that the team was not just a collection of individuals working toward a product but a community united by shared values. The onboarding process reflected this philosophy.

Our first All Hands meeting focused on introducing the team to OurCode's vision and values. This meeting was more than just a procedural formality; it was an opportunity to connect on a human level, build trust, and foster a sense of belonging. To foster connection and reflection, we experimented with onboarding activities like journaling and personal sharing sessions. While reactions were mixed, they helped spark interpersonal connections and discussions around team roles and contributions.

However, the process of integrating interns was far from seamless. Many were accustomed to a classroom dynamic, with clear tasks and assessments. Shifting this mindset to a process-oriented, collaborative approach required significant patience and reinforcement. Encouragement to view tasks as part of a larger team effort, rather than isolated assignments, was a persistent struggle. We frequently emphasized the importance of communication and iterative learning, but translating this into action proved challenging.

While we had previously evaluated our tool through lab studies with dozens of users, no users had yet regularly used the tool as part of their work. When an intern tried, they identified barriers to using the tool, including documentation that offered an incomplete picture of how to start using the tool, usability issues, and performance issues using some of the tools' features. While we hoped to learn about how it would fit into their workflow and refine our notion of where it would help most, we instead only learned that more engineering work was first needed. Another aim for our interns was to serve as a showcase for good documentation practices. However, this goal was only partially met. While some progress was made, they struggled to understand what was important to document and to attain the expertise necessary to document it.

While engineering tasks were assigned, interns often gravitated toward the easiest ones, avoiding larger, more complex commitments. This lack of ownership and accountability limited the team's overall progress. To address these challenges, we experimented with various strategies, such as pair programming, brainstorming sessions, and task-oriented challenges. However, even these initiatives had mixed success, as motivation and engagement levels varied widely. Some activities, like journaling and documentation participation, offered insights into team dynamics and preferences, but they fell short of driving substantial progress on core objectives.

5 Customer Discovery

Transitioning from company-building to customer discovery, our mindset shifted dramatically. Initially, our efforts were inward-focused: branding, building a team, and translating our research into a product. But once we began interviews, we realized that our differentiators often did not resonate with customers. This marked a critical turning point—from projecting value to actively listening and reshaping our offering based on real-world needs.

Most startups fail not due to the challenges engineering a product but because there are not enough customers who choose to buy their product. The concept of product-market fit, creating a product that satisfies a pressing need of its customers, is central to the journey of a startup. It is often the case that what startups think customers need differs from what customers actually need. Building a successful startup can be seen as a search for a business that can successfully, repeatedly, and scalably sell to customers [1]. In our project, we used techniques from the Lean Launchpad to talk to potential customers, learn about their key practices and pain points, and carefully examine how our product might or might not be the best way for customers to meet these needs.

5.1 Identifying a Value Proposition

Identifying our customer and value proposition was a pivotal aspect of OurCode's journey. Early on, our assumptions about potential customers and their pain points were broad and, as we learned, incomplete. Initially, we conflated users with customers, focusing on the value our tool offered to developers, the end users, while overlooking the priorities of decision-makers who controlled purchasing. This misalignment became evident during our interactions with mentors in the Venture Mentoring Team (VMT). They challenged us to think beyond functionality and consider the business value for customers, such as return on investment (ROI) and alignment with organizational goals.

One suggestion was to target organizations migrating legacy COBOL projects, as these teams faced acute documentation and maintenance challenges. While this niche presented clear opportunities, it was far removed from our network and expertise, making it difficult to access and validate. This underscored balancing opportunity with practicality when defining a target market.

Through iterative feedback, we began differentiating between adding value to the customer, such as reducing operational costs or improving compliance, and enhancing the experience for users by addressing pain points like inefficient onboarding or poor documentation practices. This dual focus enabled us to refine our messaging and prioritize features that bridged both perspectives.

One software architect at a mid-sized company explained that while they regularly adopted developer tools, it was challenging to quantify productivity gains clearly enough to justify new purchases. Developer tools were easiest to approve when they (1) supported compliance (e.g., for security or auditing), (2) reduced costs in large work-in-progress projects (e.g., migrations), or (3) reduced cloud expenses. After several painful migrations between paid documentation tools, his team decided to avoid additional vendor lock-in and prioritized open-source alternatives with no licensing costs.

5.2 Customer Discovery Process

Finding and engaging customers and users is one of the most challenging aspects of building a product or business. This requires not only identifying customers who might benefit from the product, but also customers who are representative of a broader market segment,

have decision-making power, and experience the problem being solved. For us, this meant connecting with engineering managers, team leads, and software engineers.

Initially, we believed our solution would appeal to engineering managers across companies of all sizes. However, we found that small to medium-sized companies—particularly those with challenges in knowledge transfer for long-term maintenance—were a better fit. This realization came after dozens of conversations that exposed the nuanced needs and barriers of different organizational contexts. We started by reaching out to our network—former colleagues, friends, and family in the software industry—before targeting specific individuals and roles. Beyond potential customers and decision-makers, we also spoke with influencers and potential partners who could improve our understanding of the market.

To connect with these stakeholders, we used a mix of strategies. Attending trade shows, conferences, and meetups such as the GitLab DevSecOps World Tour, Chicas en Tecnología Festival, and local Code & Coffee events allowed us to meet potential users in informal yet impactful settings. On LinkedIn, personalized messages and posts asking "Who should I connect with?" proved more effective than directly asking for interviews. Referrals from interviewees expanded our network exponentially, adding credibility and opening doors to more meaningful conversations.

Engaging stakeholders effectively meant approaching each interaction as an opportunity to learn. We adapted our communication to suit the audience, diving into specific workflows with technical users and focusing on cost-saving benefits with managers. Inviting participants to reflect on their top challenges fostered thoughtful discussions and helped establish goodwill, laying the foundation for long-term relationships.

Through over 100 interviews, we honed our questioning techniques. Starting with broad, open-ended questions like "What are the biggest challenges in your workflow?" encouraged free sharing, while follow-up queries such as "How do you document design decisions, and what's most frustrating about it?" uncovered deeper insights. Avoiding leading questions ensured authentic responses, and hypothesis testing validated or refined our assumptions. As patterns emerged, we focused our scripts on key areas like adoption processes, budget constraints, and decision-making dynamics.

Undertaking this process required empathy, adaptability, and structure. Through extensive networking, deliberate interactions, and iterative questioning, we identified our target audience and gained a deeper understanding of their challenges. These experiences shaped our market perspective and provided a foundation of lessons that will guide future endeavors, offering a blueprint for bridging the gap between user needs and business opportunities.

5.3 Onboarding and Software Documentation

Through extensive customer interviews, we gained valuable insights into the challenges organizations face with onboarding and software documentation practices. These learnings provided a deeper understanding of pain points, the variability of approaches across companies, and opportunities to improve developer workflows.

A recurring theme in our interviews was the difficulty organizations encounter when onboarding new developers. Many engineering managers highlighted how onboarding is often a slow and resource-intensive process, particularly for teams working with legacy codebases or complex projects. Key challenges included:

Knowledge Transfer: Developers frequently need to explore unfamiliar parts of the codebase to become productive. This process is hindered by insufficient documentation, resulting in reliance on oral knowledge transfer or informal mentoring, which disrupts senior developers' workflows.

Fragmented Processes: Onboarding often lacks a cohesive structure, with new hires piecing together information from disparate sources, including outdated documentation, code comments, and direct guidance from teammates.

Time to Productivity: Organizations expressed concerns about how long it takes for new hires to make meaningful contributions. Poor documentation and inefficient onboarding processes were cited as major factors slowing this timeline. Software documentation emerged as both a critical enabler and a persistent pain point for teams. While most organizations recognize the value of documentation, several patterns of challenges were consistently noted:

Perceived Low Priority: Documentation is often deprioritized compared to feature development, leading to outdated or incomplete materials. This creates a cycle where developers distrust documentation and rely instead on their memory or colleagues.

Switching Costs: Teams that adopt or transition between documentation tools often face significant disruption. One interviewee described multiple painful migrations between paid documentation tools, leading to a preference for Open Source solutions that minimize costs and offer flexibility.

Disconnected Documentation: A major pain point was the lack of integration between documentation and code. Static documents that fail to update with code changes result in mismatches that further erode the trust in documentation.

Adopting new tools or practices to address onboarding and documentation challenges faces several obstacles. Often teams struggle to quantify the return on investment of documentation improvements, often seeing it as an additional cost rather than a long-term enabler of productivity. At the same time, developers may resist adopting documentation tools if they perceive the workflow as cumbersome or misaligned with their existing practices.

These customer interviews underscored the critical importance of addressing onboarding and documentation practices as part of improving developer workflows. By focusing on tools and processes that reduce friction, support knowledge transfer, and integrate seamlessly into existing systems, organizations can enhance developer productivity and streamline onboarding.

5.4 Adoption of Developer Tools

The adoption of developer tools in organizations is shaped by how tools are sourced, the processes driving their adoption, and the challenges associated with demonstrating their value. Understanding this is critical for positioning new tools in a competitive market.

Organizations typically acquire developer tools in one of three ways. In-house development is common in larger companies that build custom tools tailored to their specific needs. These tools are often based on existing Open Source solutions, combining community-driven innovation with company-specific adaptations. While this provides flexibility and integration, it demands significant internal

resources for development and ongoing maintenance. Purchases from established vendors are also common, with many organizations selecting tools from well-known companies due to their perceived reliability, robust support, and long-term stability. Startups face challenges, as they are perceived as riskier due to their limited resources, lack of reputation, and uncertain longevity. This hesitation is particularly pronounced in industries with strict compliance requirements or during economic downturns when risk tolerance is low. Finally, mixed purchases from multiple vendors are prevalent among smaller and medium-sized companies. This allows organizations to select tools addressing specific needs without committing to a single vendor, with OSS solutions often crucial. OSS tools offer affordability and flexibility but can lead to integration challenges and tool redundancy, where overlapping functionality creates inefficiencies. Balancing the benefits of OSS innovation with the need for a cohesive toolset remains a common challenge.

Adoption processes for developer tools typically follow one of two models: bottom-up adoption or top-down adoption. Bottom-up adoption is driven by developers or team leads who independently discover tools, often through social media, open-source communities, or personal projects. These individuals experiment with tools, share their experiences, and advocate for adoption within teams. In contrast, top-down adoption is initiated by management as part of strategic priorities, such as reducing the risk of security vulnerabilities or increasing release cadence. While top-down adoption ensures alignment with organizational goals, it can face resistance if the tool does not meet developers' practical needs or integrate effectively with existing workflows. Both models have strengths and weaknesses, and successful adoption often requires bridging the gap between individual enthusiasm and organizational priorities.

One of the biggest challenges in adopting developer tools is making a compelling business case. Organizations often struggle to quantify the productivity impact of new tools, as it is difficult to measure how they influence efficiency or developer output. Usage uncertainty—how many developers will actively use the tool and how often its benefits apply—further complicates justifying the investment. Consequently, developer tools are frequently perceived as an added expense rather than a cost-saving measure, especially in cost-conscious environments.

Our interview process identified specific scenarios where the business case for developer tools is strongest. Tools that support compliance requirements, such as security or auditing, are prioritized because their results are essential for organizations to sell their products. Similarly, tools that significantly reduce the costs of work in progress, such as migration projects, are easier to justify due to their direct financial impact. Another area where tools gain traction is in their ability to reduce cloud spend, a growing priority as companies look to optimize operational costs. However, organizations remain cautious of tools that require disruptive transitions or fail to deliver clear results. For instance, one interviewee noted that their team had switched paid documentation tools multiple times, each migration causing significant disruption. This experience led them to prefer OSS solutions, which are perceived as cost-effective and flexible alternatives.

To succeed in this landscape, new developer tools must directly address these challenges. Communicating a clear value proposition—such as reducing knowledge transfer time, enabling faster onboarding, or solving specific pain points—can help shift perceptions from cost to value. Tools that align with existing workflows and integrate seamlessly into current systems can reduce resistance and foster adoption. Demonstrating tangible outcomes through pilot projects or case studies builds trust and makes the business case more compelling, ensuring a smoother path to adoption.

6 Challenges Starting a Company

Our challenges were not just technical, they were existential. One of the most pivotal moments came when we sat down to review the data from our 100+ interviews and saw a simple, painful pattern: only two people said they would pay to try our product. We were bootstrapping, working multiple jobs, and navigating financial constraints. The effort required to keep building, both in time and money, no longer made sense. This turning point reframed everything. From that moment on, our decisions were shaped not just by optimism, but by realism.

As we began to discover through our conversations with potential customers, we began our company in challenging times for developer tools. Due to the economic conditions, many companies were focused on cutting costs. We discovered that, while no one felt that onboarding worked well or that software documentation was up to date and trustworthy, neither of these were top priorities for most decision makers. Developer tools were often perceived as a new cost, and the focus was more on how to reduce these costs by removing developer tools that were unnecessary, infrequently used, or which overlapped in functionality with other tools.

An additional challenge was the positioning of our technology itself. As we finished the basic research that enabled our tooling and began the work of starting the company, many developer workflows were quickly changing with the rapid adoption of generative AI. Whereas creating software documentation had always previously been perceived as a manual and labor-intensive process, generative AI based tools offered the promise of replacing this with automation, where a tool could create documentation (e.g., Swimm[4]). Other tools promised to remove the need for creating software documentation entirely. Instead of referencing a design document, a developer might instead interact with a generative AI chat tool, which draws on knowledge stored in chat logs, issue trackers, or wikis.

Generative AI tools did not entirely remove the need for our technology. As developers generate more code through AI, we expected that there would be increased need for tooling to help establish trust in this code. Moreover, while generative AI tools can ingest text from knowledge stores, this information may itself, like software documentation, be out of date and incomplete. Our technology could offer a compelling solution to this problem, making it easy for a developer to establish a clear design and architecture for a codebase, captured in checkable rules, and ensure that all AI-generated code follows this design and architecture whenever it is generated. But building this tooling would require a pivot—a rethinking of our key value proposition and the features which support this— which would require substantial engineering time to realize.

Our startup also faced more practical challenges. As we found from having our interns attempt to dogfood our product, our tool needed more work to address defects, improve its documentation,

[4]https://swimm.io/

and address performance issues. In addition, we had designed the tool to support Java. But few of the prospective customers we found wanted to use it for Java. One potential market for our tool was to assist in working with really old codebases, such as those written in Cobol. Or, we could alternatively support more popular languages for new projects, such as JavaScript or Python. However, our tool was built on top of an AST query engine [6] which did not support any of these languages. Migrating to a modern query engine would address this, but would require months of work. In the meantime, our tool lacked feedback from users to guide its direction.

A final challenge was the nature of our funding. We had funding for a short 7-week customer discovery effort through the National I-Corps, but this did not include follow-on funding to do meaningful engineering work to refocus our product based on what we had learned from interacting with potential customers. Our primary funding source was bootstrapping. One of us worked part time on this project while spending most of their time seeking outside consulting jobs. Another was employed in a faculty role, which left only part time commitments available for this project. And our third member had no employment at all. All of this meant that there was strong pressure to obtain income fast, with little time available for addressing the many challenges we faced.

As a result of these challenges, our company ultimately shutdown in December 2024, just over 13 months after incorporation. This decision resulted in the "Intellectual Honesty Award", honored by NSF during the I-Corps program, recognizing our commitment to relying on observed data and addressing real user needs rather than building for the sake of building. Rather than build a company, we planned to instead continue to offer our technology as an open source project. However, building a successful open source developer tool requires much the same work as building a commercial company: marketing work to build awareness, engagement with users to understand market needs, and engineering work to build a product. While a community might assist with some of these tasks, providing leadership and guidance can be just as time consuming as running a company, and difficult to balance with the many other commitments of researchers.

7 Lessons from a Failed Startup: Insights on Tech Transfer for Researchers

Transitioning academic research into a startup is often portrayed as a success story when it ends in acquisition or scale. But even when a startup does not reach that outcome, it can serve as a powerful learning tool, both for the researchers involved and the broader academic community. Our experience surfaces insights often absent in success narratives: what misalignments matter, how researchers can improve their approach, and what systemic barriers remain.

One of the hardest lessons we learned was that solving a substantial problem for many users does not automatically translate into a successful business. This realization emerged as we grappled with the fundamental gap between the value experienced by end-users and the priorities of decision-makers. Developers, who interact directly with tools, are not always the ones making purchasing decisions. Decision-makers prioritize business value, like return on investment, risk mitigation, or cost savings, while developers focus on usability and seamless integration into their workflows.

The gap between research and industry in the developer tools space is vast and shaped by diverging incentives. Research prioritizes long-term goals, novelty, and incremental contributions to knowledge, while industry emphasizes immediate solutions and rapidly evolving trends. Academic prototypes, often referred to as "researchware", are designed to demonstrate technical capabilities and rigor but may fail to address the broader, practical workflows that industry values. For example, while our prototype rigorously captured documentation design rules, we questioned whether this approach fully aligned with the real pain points of software teams. This tension underscored the need for iterative feedback and real-world validation, both of which are essential for bridging this divide.

Engaging users early was critical to uncovering practical insights. While academic rigor often prioritizes large sample sizes or controlled experiments, even brief, informal conversations revealed obvious yet transformative lessons about user workflows and priorities. These non-rigorous interactions, though undervalued in academic circles, were foundational to refining our approach. Rigor, we learned, is not solely about increasing sample sizes but about adopting a broader perspective that includes iterative learning through real-world feedback.

Recruiting participants for these conversations required a shift from traditional academic approaches. Developers are often treated as a homogeneous group based on years of experience or roles, but their needs vary widely depending on organizational context, team dynamics, and personal work styles. Networking became our primary recruitment strategy, relying on personal contacts, referrals, and trust. This hands-on approach helped us find participants who could provide meaningful, contextual feedback while also maintaining valuable social capital within our network.

Finally, the industry's focus on trends and immediate applicability presented additional challenges. Companies prefer tools that have been vetted by peers and align with established trends. As a startup with an unproven tool, building credibility in a competitive market was difficult. Generative AI, for instance, reshaped developer workflows during our journey, presenting both new challenges and opportunities. These disruptions further emphasized the importance of adaptability in aligning with market needs.

Transitioning from research to industry requires balancing rigor with practicality, fostering early and iterative user engagement, and understanding the dynamics between users and decision-makers. While our journey was marked by challenges, it offered invaluable lessons that future teams can use to avoid similar pitfalls. Our experience shows that a startup, even when it does not succeed commercially, can still generate critical insights for aligning academic innovation with industry needs. In this way, our "failure" becomes a valuable outcome: one that can inform more grounded, impact-oriented research practices and better prepare future academic entrepreneurs for the realities of commercialization.

8 Acknowledgments

This work was supported in part by the US National Science Foundation and Innovation Corps (I-Corps) program under grants 1703734, 1845508, and 2433803. We thank our mentors, interview participants, and the broader software engineering community for their invaluable insights and support throughout this journey.

References

[1] Blank, S., and Dorf, B. *The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company.* K & S Ranch, 2012.

[2] Calcagno, C., and Distefano, D. Infer: An automatic program verifier for memory safety of c programs. In *NASA Formal Methods* (Berlin, Heidelberg, 2011), M. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, Eds., Springer Berlin Heidelberg, pp. 459–465.

[3] Chilana, P. K., Ko, A. J., and Wobbrock, J. From user-centered to adoption-centered design: A case study of an hci research innovation becoming a product. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2015), CHI '15, Association for Computing Machinery, p. 1749–1758.

[4] Chilana, P. K., Ko, A. J., and Wobbrock, J. O. Lemonaid: selection-based crowdsourced contextual help for web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, Association for Computing Machinery, p. 1549–1558.

[5] Chilana, P. K., Ko, A. J., Wobbrock, J. O., and Grossman, T. A multi-site field study of crowdsourced contextual help: usage and perspectives of end users and software teams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, Association for Computing Machinery, p. 217–226.

[6] Collard, M. L., Decker, M. J., and Maletic, J. I. srcml: An infrastructure for the exploration, analysis, and manipulation of source code: A tool demonstration. In *2013 IEEE International Conference on Software Maintenance* (2013), pp. 516–519.

[7] Friedman, N. Welcome semmle to the github family. https://github.blog/news-insights/company-news/github-welcomes-semmle/, Nov. 2019. Accessed: 2025-04-16.

[8] LaToza, T. D., Venolia, G., and DeLine, R. Maintaining mental models: a study of developer work habits. In *International Conference on Software Engineering (ICSE)* (2006), pp. 492–501.

[9] Ltd., M. Facebook to acquire monoidics' assets! https://archive.ph/20130718144452/http://www.monoidics.com/uncategorized/facebook-to-acquire-monoidics-assets, 2013. Accessed: 2025-04-16.

[10] Markman, G. D., Gianiodis, P. T., Phan, P. H., and Balkin, D. B. Innovation speed: Transferring university technology to market. *Research Policy 34*, 7 (2005), 1058–1075. The Creation of Spin-off Firms at Public Research Institutions: Managerial and Policy Implcations.

[11] Mehrpour, S., and LaToza, T. D. Can program analysis tools find more defects? a qualitative study of design rule violations found by code review. *Empirical Software Engineering (EMSE) 28*, 1 (nov 2022).

[12] Mehrpour, S., LaToza, T. D., and Kindi, R. K. Active Documentation: Helping Developers Follow Design Decisions. In *Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2019), pp. 87–96.

[13] Mehrpour, S., LaToza, T. D., and Sarvari, H. Rulepad: interactive authoring of checkable design rules. In *European Software Engineering Conference and International Symposium on the Foundations of Software Engineering (ESEC/FSE)* (2020), pp. 386–397.

[14] Planview, I. Planview announces strategic acquisition of tasktop. https://newsroom.planview.com/planview-announces-strategic-acquisition-of-tasktop/, May 2022. Accessed: 2025-04-16.

[15] Postman. Postman acquires akita for automated api observability. https://blog.postman.com/postman-acquires-akita-for-automated-api-observability/, Jan. 2025. Accessed: 2025-01-14.

[16] Redwine, S. T., and Riddle, W. E. Software technology maturation. In *Proceedings of the 8th International Conference on Software Engineering* (Washington, DC, USA, 1985), ICSE '85, IEEE Computer Society Press, p. 189–200.

[17] Rogers, E. M. *Diffusion of innovations.* Free Press, New York, 2003.

[18] Rothaermel, F. T., Agung, S. D., and Jiang, L. University entrepreneurship: a taxonomy of the literature. *Industrial and Corporate Change 16*, 4 (07 2007), 691–791.

[19] Sawers, P. Sapienz: Facebook's push to automate software testing. https://venturebeat.com/business/sapienz-facebooks-push-to-automate-software-testing/, December 2018. Accessed: 2025-04-16.

[20] Siegel, D. S., Veugelers, R., and Wright, M. Technology transfer offices and commercialization of university intellectual property: performance and policy implications. *Oxford Review of Economic Policy 23*, 4 (01 2007), 640–660.

[21] Synopsys, I. Synopsys enters software quality and security market with coverity acquisition, 2014. Accessed: 2025-04-16.