Navigating Large-Scale Development: Lessons from Two Summers at Amazon

CS4991 Capstone Report, 2025

Ethan Christian
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
vya9tu@virginia.edu

ABSTRACT
Transitioning from academic coding to industry-scale software development presents challenges in collaboration, codebase navigation, and workflow adaptation. During two consecutive internships at Amazon, I engaged with enterprise-level systems, contributing to backend services and tools while refining my understanding of professional software engineering practices. Key experiences included rigorous code reviews, agile sprint cycles, design document collaboration, and problem-solving within large codebases. I synthesize lessons learned, emphasizing teamwork, iterative development, and the balance between autonomy and mentorship. Future applications include leveraging these skills in full-time roles and mentoring peers entering similar environments.

1. INTRODUCTION

Transitioning from academic projects to enterprise software development at Amazon marked a major turning point in my professional journey. In an environment where each coding decision affected a vast, interconnected system, traditional academic methods gave way to more flexible, carefully planned solutions.

During my first summer internship, I focused on developing a testing framework that closely mirrored live conditions. This task required balancing creative problem-solving with a strong commitment to system stability, as even small changes could impact critical operations.

The following summer offered the opportunity to refine deployment processes and set up continuous monitoring protocols. This stage demanded careful attention to both the fine details of code execution and a broader view of overall system performance. In doing so, it became clear that agile methods, rigorous testing, and teamwork are essential for success.

Altogether, these hands-on experiences have deepened the understanding of large-scale development, highlighting the need to combine innovative ideas with thoughtful planning and ongoing learning.

2. RELATED WORKS

Literature on agile methodologies forms a strong foundation for modern software practices. The Agile Manifesto (Beck et al., 2001) outlines principles such as iterative progress and continuous feedback, which are essential for managing complex systems and striking a balance between rapid innovation and system stability.

Research on code review processes reinforces the importance of collaborative evaluation in software development. Rigby and Storey (2011) demonstrate that systematic, peer-

driven reviews are crucial for identifying issues early, thereby reducing risk in large-scale projects

In addition, studies focusing on undergraduate experiences and internships provide valuable insights into the transition from academic learning to industry practice. Jones and Smith (2010) illustrate how internships serve as a critical bridge between classroom theory and real-world application, equipping students with the practical skills necessary for navigating complex systems. Similarly, Adams (2016) highlights the positive impact of early industry exposure on technical proficiency and professional readiness, suggesting that hands-on experience is indispensable for aspiring computer scientists.

3. PROJECT DESIGN

This section outlines the primary contributions made during the summer internships, detailing the structured approach followed to develop the project from concept to final presentation. The process represents the general timeline and methodology followed for intern projects across both years. This standardized approach ensured consistency in planning, implementation, and evaluation throughout the internship experience.

3.1 Design Document and Planning

The project began with the creation of a comprehensive design document. This document served as both an internal blueprint and an external communication tool, detailing the project's purpose, importance, and the proposed solution—including system architecture, coding strategies, and required packages. Before finalization, the document was reviewed collaboratively with the entire team, and suggested revisions were incorporated. This process ensured that the design was accessible and understandable not only within the immediate team but also for other teams across the company.

3.2 Implementation with Sprint Architecture

Once the design document was approved, the project moved into the implementation phase, which was organized using a sprint architecture. Tasks and milestones were clearly laid out on the scrum board, facilitating a structured and iterative development process. This approach allowed for regular assessment of progress and timely adjustments, ensuring that the project remained aligned with its initial objectives while accommodating evolving requirements.

3.3 Code Review and Final Presentation

Quality assurance was maintained through a rigorous code review process. Every code change was subjected to at least two team member reviews before integration, ensuring that the solution met high standards of reliability and performance. The final phase involved synthesizing the entire effort into a detailed presentation. This presentation showcased the project's outcomes, highlighted key challenges and solutions, and reflected on lessons learned throughout the internship.

In the first internship, the deliverable known as Nostradamus was developed to generate synthetic telemetry data. This tool provided realistic, fake data to the FleetWatch alarming service, enabling both routine tests of alarm functionality and integration tests for new features. By automating what was once a tedious, manual process at various data centers, Nostradamus streamlined testing and ensured that updates could be implemented with greater confidence in system stability.

During the second internship, the focus shifted to the NotificationHistoryService, which was designed to capture a comprehensive history of notifications sent from the FleetWatch team. Leveraging Amazon DynamoDB for data persistence and an SNS queue for event

handling, this service systematically records critical datacenter metrics such as voltage, current, temperature, and humidity. The robust architecture supports seamless code deployment across production, development, and beta environments, thereby enhancing system traceability and facilitating faster troubleshooting and proactive maintenance.

4. RESULTS

During my first internship, I developed Nostradamus—a project designed to automatically generate synthetic telemetry data for the FleetWatch alarming service. This deliverable enabled routine testing to verify that alarms were operational and supported integration tests for new features, ensuring that system updates did not break existing functionality. By automating what was once a tedious, manual process at data centers worldwide, Nostradamus greatly reduced testing throughput—from processes that previously took days to now being completed in just 2–3 hours.

For the second internship, preliminary outcomes indicated that the NotificationHistoryService significantly improved the traceability of system events, enabling faster troubleshooting and more proactive maintenance across teams. The centralized repository of historical data not only facilitates the identification of trends and recurring issues but also supports data-driven decision-making. As further refinements and advanced analytics are integrated, the service is anticipated to enhance operational efficiency and scalability, solidifying its role as a critical component in managing large-scale infrastructure monitor

5. CONCLUSION

This report has detailed a transformative journey from academic coding to enterprise-scale software development through two distinct internships at Amazon. The first internship's development of Nostradamus demonstrated how an automated testing framework can replace manual data alteration—reducing testing turnaround from days to just 2–3 hours—while ensuring system updates remain reliable. The second internship's NotificationHistoryService showcased a robust approach to monitoring, capturing critical datacenter metrics to improve event traceability and facilitate proactive maintenance. Together, these experiences not only enhanced technical competencies but also underscored the importance of agile practices, rigorous code reviews and effective cross-team collaboration in solving complex, real-world challenges.

By integrating industry best practices with innovative project design, these internships provided a solid foundation for addressing large-scale development challenges. The insights gained extend beyond technical improvements to emphasize adaptive planning, continuous learning and strategic communication—key factors that will inform future professional endeavors and drive the evolution of modern software engineering.

6. FUTURE WORK

Looking ahead, transitioning to a full-time role at Amazon presents an exciting opportunity to build on the lessons learned during these internships. The practical experience gained—from refining agile processes and ensuring rigorous code reviews to effectively managing cross-team communication—will inform daily operations in a full-time capacity. The intention is to scale these proven practices across a wider array of systems, further enhancing system reliability and operational efficiency.

In the next phase, efforts will focus on integrating advanced analytics and automation into existing workflows, allowing for proactive issue detection and more streamlined

deployments. These initiatives are expected to not only fortify the operational backbone of Amazon's infrastructure but also contribute to personal and professional growth, as the lessons from these internship experiences are directly applied to the challenges of full-time enterprise software development.

REFERENCES

Adams, T. (2016). Enhancing undergraduate computer science education through internships. *Journal of Computing in Higher Education*, 28, 3 (2016), 500–520.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for agile software development. Retrieved from https://agilemanifesto.org/

Jones, K. and Smith, A. (2010). Bridging the gap: The impact of internships on undergraduate computer science education. *IEEE Transactions on Education,* 53, 1 (2010), 65–72.

Rigby, P. and Storey, M. (2011). Understanding the role of code review in software development. Empirical Software Engineering, 16, 1 (2011), 3–27.