An Analysis of Language Binding in Multi-Language Systems: Optimizing FAA Simulations through C++ and Python Integration

Challenges in Accelerating Technological Modernization in Government Agencies: Addressing Barriers to Updating Legacy Systems

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Ahbey Yared Mesfin
11/8/24

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Rider Foley, Department of Engineering and Society

Rosanne Vrugtman, Department of Computer Science

Introduction

Government agencies frequently rely on legacy systems—older technologies characterized by their age, lack of technical support, and replacement by more modern alternatives—to carry out critical operations. Agencies such as the Department of Defense, Department of Education, Health & Human Services, and Homeland Security continue to depend on these systems, which pose risks to national security and public well-being due to their heightened vulnerability to malicious attacks (Harris et al., 2019). Maintaining these legacy systems is also an extremely costly endeavor, with expenditures reaching billions of dollars nationwide (Harris et al., 2019).

The barriers to modernizing these aging systems are deeply rooted not only in technical limitations but also in social factors within government organizations. Employee resistance to change, fear of losing control over familiar systems, and insecurity about new technology all contribute to reluctance in adopting updated solutions (Elgohary & Abdelazyz, 2020). Addressing these social and organizational challenges is essential for enabling the modernization of legacy systems and reducing the operational and security risks associated with outdated technology.

Modern software development often involves multiple programming languages, each chosen for its suitability to specific tasks. However, this multilingual approach creates complex interfaces between languages, which can lead to decreased performance and limited flexibility (Grimmer et al., 2018). Language interoperability—the ability for different programming languages to work seamlessly together within a single program—presents a potential solution to these issues. One method for achieving interoperability is language binding, which allows code written in one language to directly access and execute code in another, circumventing

compatibility barriers (Wegner, 1996). By exploring language binding, this research seeks to optimize legacy systems and demonstrate how interoperability could enhance the functionality and adaptability of existing technology.

This prospectus examines the technical and social challenges of modernizing legacy government systems, focusing on language interoperability as a case study. Through this research, I aim to address the socio-technical barriers that limit innovation in public agencies and propose solutions for a more collaborative and flexible approach to technology adoption in the government sector.

PROGRAMMING LANGUAGE BINDINGS

During my internship at The MITRE Corporation, I focused on optimizing a flight simulation program for the Federal Aviation Administration (FAA). The simulation had two primary components: A C++ component, and a Python component. The C++ component was responsible for doing all of the intensive calculations for the simulation when provided with the appropriate parameters. The Python component, a REST interface (REST means Representational State Transfer, a known software architecture) handled other tasks for the simulation, such as displaying the data output from the C++ component in a clean, readable way (among other tasks). It also served as an interface to increase ease-of-use for the end-user, as working directly with the large C++ codebase could be cumbersome. The two components relate as follows: after the C++ code finished all of its calculations, it would write its results to a file within a shared filesystem. Once all the data was written, the Python REST interface would read in the data from that shared file and display it. This architecture was reliable with respect to actual functionality, but it was significantly slowed down by the overhead associated with

file-based data transfer. The read/write operations with the filesystem were particularly time consuming - taking as long or longer than the computational steps themselves. My job was to create a more efficient architecture by writing a “language binding” between the Python and C++ components. This would allow direct data exchange between the two, bypassing the filesystem altogether.

To achieve this goal, I first studied the existing codebase and noted any code that attempted to communicate with the filesystem. Next, I implemented replacement functions in those parts of the codebase that retained the role of the function, but without any filesystem communication. I then used open-source tools to develop the language binding that would enable direct data transfer between the two components. This approach dramatically improved the performance of the simulation by solving the file system bottleneck. The result of my work was an optimized simulation, with data passing between components seamlessly. Figure 1 below illustrates the before-and-after architecture of the simulation.

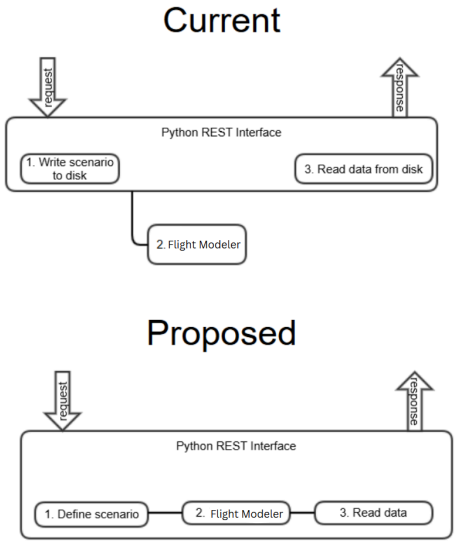

Figure 1: Flight Simulation Architecture Redesign

This technical project illustrates the core of my research question: technical overhauls alone are insufficient for modernization if they lack support through changes in organizational practice. This work was only possible because both major parties (The MITRE Corporation and the FAA) endorsed and encouraged the creation of this project. It is a common phenomena that employees resist changes to modernizing a system they commonly use (Khadka et al., 2015), and had that been the case here, this project would have never begun. Another major source of resistance to modernization is policy-driven rather than organizational practice (Alexandrova et al., 2015). By showing the practical benefits of improving a known system, language interoperability in this case, this project advocates for a reevaluation of accepted practice and policy, and promotes the positives of modernization by demonstrably improving a frequently used tool within the FAA.

SOCIAL ISSUES IN MODERNIZING GOVERNMENT LEGACY SYSTEMS

Government agencies face significant challenges in modernizing legacy systems, which are deeply rooted in institutional practices and technical infrastructures. Modernization is the process of updating (and sometimes replacing entirely) outdated computer systems and software with more modern technologies. Despite their known security vulnerabilities and inefficiencies, these systems persist due to a complex web of social dynamics and regulatory constraints within the public sector (Abu Bakar et al., 2022). The slow rate of modernization in these agencies is not simply a technical problem; it is a socio-technical issue that requires a nuanced understanding of the interactions between technology, organizational culture, and policy frameworks. Modernizing these systems is not a trivial endeavor, requiring both technological

solutions and the willingness to address the institutional and social factors that reinforce their use.

A useful lens for examining this problem is path dependency, which explains how early technological choices create dependencies that are difficult to break, especially as systems become more deeply embedded over time (Mahoney, 2000). As agencies make initial investments in particular technologies, those technologies often become "locked-in" due to the high cost (financial and time) of transitioning to new systems , especially given the scale and complexity of public agency infrastructures. This "lock-in" effect contributes to the persistence of inefficient legacy technologies as the industry standard, allowing inefficiencies to endure for extended periods of time (Barnes et al., 2004). Path dependency also highlights how initial design decisions, reinforced by institutional policies and norms, shape the technological trajectory of agencies. For instance, once a technology is established, agencies develop procedures and training programs around it, making future changes more costly and disruptive which accrues technical debt (Monaghan & Bass, 2020). Understanding path dependency allows us to see modernization not just as a technical upgrade but as a transformation that challenges entrenched practices and investments.

Another framework that complements this perspective is technological momentum. This concept, introduced by Hughes (1987), posits that as a technology becomes integrated into social structures, it begins to shape society as much as society shapes it (Taylor, Johnsen, 1986). In the case of government legacy systems, technological momentum implies that these systems do more than simply support organizational functions; they actively shape agency routines and employee roles. Over time, legacy systems acquire their own "inertia", becoming increasingly difficult to overhaul/replace. This momentum reflects a paradox within government agencies:

while technology is intended to foster efficiency, the degree of its integration within organizational practices can actually significantly slow down modernization efforts. The more deeply entrenched a legacy system becomes, the more resistant it is to change - despite security and performance concerns. Technological momentum thus helps to explain why, as legacy systems mature, they become less adaptable and more resistant to external pressures for change (Schubert et al., 2013), making modernization efforts more complex as time progresses.

These two frameworks highlight the need for a comprehensive approach to modernization that considers both technical and socio-cultural roadblocks. Path dependency and technological momentum suggest that efforts to modernize aging government systems need to provide solutions to the social elements at play, such as institutional resistance, rather than purely technical solutions. Thus, these frameworks offer a comprehensive perspective for understanding and addressing the different complexities that come with modernizing government legacy systems.

RESEARCH QUESTION AND METHODS

This research has the aim of answering "What are the primary socio-technical barriers to modernizing legacy systems within government agencies" This question is significant because outdated government systems pose significant risk to security and efficiency - understanding the social and technical factors that hinder modernization efforts is essential for working towards effective strategies to transition these systems to more efficient and secure architectures. My research will begin by looking at the existing modernization steps, and observing the strengths and weaknesses of their implementations. For example, cloud computing (Chiang & Bayrak, 2006) , "modularized modernization" (Jain & Chana, 2015) and Service-oriented architectures

(Galinium, 2008) are three concrete steps for achieving modernization in aging systems, of which I can analyze and compare/contrast. The data will be collected from these case studies of legacy system modernization efforts in different government agencies. For the social elements of my research question, I will look to studies that primarily perform data collection via interviews with IT professionals and project managers (Schubert et al., 2013), as well as government policy analysis (Lam, 2005), (Waylen et al., 2015). I aim to observe how the two noted STS frameworks, technological momentum and path dependency, are illustrated in such studies. For example, there exist direct statements within these interviews that carry the sentiment of wanting to maintain the current systems simply because they are "tried and true", going hand-in-hand with path dependency (Schubert et al., 2013). The core idea among these studies is to investigate the "human element" of system modernization - exploring the reasons why certain workers are strictly against modernization (Wellar et al., 2011), (Alexandrova & Rapanotti, 2020).

CONCLUSION

This paper addresses the rising issue of outdated legacy systems within government agencies, which pose a threat to security, efficiency, and the general public good. My technical work, a language binding interface between Python and C++ components, demonstrates the potential of interoperability to optimize runtime performance in systems that depend on multiple programming languages. The STS deliverable lays the groundwork for my research by providing frameworks, namely path dependency and technological momentum, for understanding the socio-technical barriers that hamper modernization efforts and how to begin addressing them. This work aims to encourage more adaptable, secure, and efficient technology solutions within the public sector. If implemented accordingly, these deliverables could illustrate a more flexible

approach for technological adoption in the government, ultimately benefiting the efficacy and security of said systems. The research is meant to offer real, actionable insights that bolster technological advancements as well as socio-organizational adoption in seemingly resistant government institutions.

REFERENCES

Abu Bakar, H., Razali, R., & Jambari, D. I. (2022). A Qualitative Study of Legacy Systems Modernisation for Citizen-Centric Digital Government. Sustainability, 14(17), Article 17. https://doi.org/10.3390/su141710951

Alexandrova, A., & Rapanotti, L. (2020). Requirements analysis gamification in legacy system replacement projects. Requirements Engineering, 25(2), 131–151. https://doi.org/10.1007/s00766-019-00311-2

Alexandrova, A., Rapanotti, L., & Horrocks, I. (2015). The legacy problem in government agencies: An exploratory study. Proceedings of the 16th Annual International Conference on Digital Government Research, 150–159. https://doi.org/10.1145/2757401.2757406

Barnes, W., Gartland, M., & Stack, M. (2004). Old Habits Die Hard: Path Dependency and Behavioral Lock-In. Journal of Economic Issues, 38(2), 371–377.

Chiang, C.-C., & Bayrak, C. (2006). Legacy Software Modernization. 2006 IEEE International Conference on Systems, Man and Cybernetics, 2, 1304–1309. https://doi.org/10.1109/ICSMC.2006.384895

Elgohary, E., & Abdelazyz, R. (2020). The impact of employees' resistance to change on implementing e-government systems: An empirical study in Egypt. THE ELECTRONIC

JOURNAL OF INFORMATION SYSTEMS IN DEVELOPING COUNTRIES, 86(6), e12139. https://doi.org/10.1002/isd2.12139

Galinium, M., & Shagbaz, N. (2009). Factors Affecting Success in Migration of Legacy Systems to Service-Oriented Architecture (SOA).

Grimmer, M., Schatz, R., Seaton, C., Würthinger, T., Luján, M., & Mössenböck, H. (2018). Cross-Language Interoperability in a Multi-Language Runtime. ACM Transactions on Programming Languages and Systems, 40(2), 1–43. https://doi.org/10.1145/3201898

Hughes, (1987). The Evolution of Large Technological Systems, 51-81.

Information Technology: Agencies Need to Develop Modernization Plans for Critical Legacy Systems. (n.d.). Retrieved October 19, 2024, from https://apps.dtic.mil/sti/citations/AD1156632

Jain, S., & Chana, I. (2015). Modernization of Legacy Systems: A Generalised Roadmap. Proceedings of the Sixth International Conference on Computer and Communication Technology 2015, 62–67. https://doi.org/10.1145/2818567.2818579

Khadka, R., Shrestha, P., Klein, B., Saeidi, A., Hage, J., Jansen, S., van Dis, E., & Bruntink, M. (2015). Does software modernization deliver what it aimed for? A post modernization analysis of five software modernization case studies. 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), 477–486. https://doi.org/10.1109/ICSM.2015.7332499

Lam, W. (2005). Barriers to e‐government integration. Journal of Enterprise Information Management, 18(5), 511–530. https://doi.org/10.1108/17410390510623981

Mahoney, J. (2000). Path Dependence in Historical Sociology. Theory and Society, 29(4), 507–548.

Monaghan, B., & Bass, J. (2020). Redefining Legacy: A Technical Debt Perspective (pp. 254–269). https://doi.org/10.1007/978-3-030-64148-1_16

Schubert, C., Sydow, J., & Windeler, A. (2013). The means of managing momentum: Bridging technological paths and organisational fields. Research Policy, 42(8), 1389–1405. https://doi.org/10.1016/j.respol.2013.04.004

Taylor, Johnson, Resisting Technological Momentum. (n.d.). https://doi.org/10.1177/016146818608700512

Wegner, P. (1996). Interoperability. ACM Computing Surveys, 28(1), 285–287. https://doi.org/10.1145/234313.234424

Waylen, K. A., Blackstock, K. L., & Holstead, K. L. (2015). How does legacy create sticking points for environmental management? Insights from challenges to implementation of the ecosystem approach. Ecology and Society, 20(2). https://www.jstor.org/stable/26270192

Wellar, B., Garrison, W. L., MacKinnon, R., Black, W. R., & Getis, A. (2011). Research Commentary: Increasing the Flexibility of Legacy Systems. Int. J. Appl. Geosp. Res., 2(2), 39–55. https://doi.org/10.4018/IJAGR.2011040104