Carnegie Mellon University
Software Engineering Institute

FedCLASS: A Case Study of Agile and Lean Practices in the Federal Government

Nanette Brown
Jeff Davenport
Linda Parker Gates
Tamara Marshall-Keim

October 2018

SPECIAL REPORT
CMU/SEI-2018-SR-016

Software Solutions Division

http://www.sei.cmu.edu

REV-03.18.2016.0

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0578

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to thank the individuals who provided access to their federal development project. Department leadership gave the Carnegie Mellon University Software Engineering Institute broad access to all project documentation and the daily operations of the development team. Without this open and broad access, it would not have been possible to create this case study. The authors also wish to thank the members of the development team who talked with us openly about the project and allowed us to observe daily standup sessions and whole days of team activities.

Finally, the authors would like to express our appreciation for all those who contributed to or reviewed this case study and its predecessor. We extend our sincerest thanks to Eric Ferguson, Ken Nidiffer, Mary Ann Lapham, Annie Drazba, Kurt Hess, Pennie Walters, Gerald Miller, and Todd Loizes.

Executive Summary

Purpose of This Case Study

This anonymized case study tells the story of an actual development project for a new software system undertaken by a key program within an executive department of the U.S. federal government. The purpose of the project was to develop a new version of the software used by this department in the performance of its mission, and to do it using iterative, Agile, and Lean development methods, an approach recently recommended by the federal government at the time of this study. This case study documents the history and approaches used during the development of the new system and illustrates the successes and challenges of applying iterative, Agile, and Lean development methods in an organization that previously used more traditional development methods. The purpose of this case study is to inform other organizations about lessons learned from this project, both positive and negative, so that they may benefit from the department's experience in piloting iterative, Agile, and Lean practices. This case study was constructed from extensive access to the project and discussions with team members of the project, other staff in the executive department, and development and testing contractors; documentation reviews; observations of daily team activities; and analysis of work products from the project.

The Business Need for a New Software System

At the time of the project, the executive department was developing strategic plans to expand the context of its operations. However, department personnel recognized that the software they were using, referred to in this report as LEGACY, had reached capacity during peak processing times and could not easily accommodate changes in its functionality.

A 2010 Federal Chief Information Officer (CIO) report had called for reform of federal information technology (IT) management, outlining 25 points that federal agencies should address in order to produce greater returns on the government's investment in IT. The department recognized that three points of the plan were especially relevant to the department's views on modernization [Kundra 2010]:

- Point 3: Shift to a "cloud first" policy
- Point 6: Develop a strategy for shared services
- Point 15: Issue contracting guidance and templates to support modular development

Recognizing LEGACY's limitations and motivated by the Federal CIO's call to reform government IT practices, the executive department decided to replace the old system with a to-be-engineered system, referred to in this report as Federal Cloud-based Lean and Agile Shared Services, or FedCLASS, which would support these new considerations.

A New Approach to Software Development

In the past, the department used traditional waterfall approaches to software development. For FedCLASS, department executives decided to use a different approach that better aligned with modern software development practices, including iterative development, Agile and Lean practices, and cloud-based technologies.

In particular, the project decided to

- manage the project using the iterative lifecycle and risk-focused techniques of the Rational Unified Process (RUP) [RSC 1998]
- incorporate Agile practices and principles, including
  - integrating business owners into the development process with direct accountability for meeting delivery goals
  - having all FedCLASS team members be dedicated 100% to the FedCLASS Project
  - allowing the development team to be self-organizing
- use current technologies and tools, including
  - writing FedCLASS in Java
  - developing in the cloud
  - employing automation tools that support Agile and Lean development practices (e.g., automated test, continuous integration, and automated build)

Because of the sweeping nature of the changes being made, the department decided to hire consultants and coaches to help implement and institutionalize these new approaches, tools, technologies, and methods.

Implementation of Agile and Lean Principles of Development

After the project kickoff, the executive department began to implement Agile principles. For the first three months, all members of the FedCLASS Project were physically co-located. After that, virtual-presence software connected team members all day via video teleconferencing. All team members maintained a constant virtual presence, appearing in a “Hollywood Squares” type of visual projection that enabled them to be constantly accessible and aware of the needs of other team members. This allowed the team to work as though they were co-located no matter where they were.

The FedCLASS team did not function according to the traditional siloed structure but instead worked together as a cross-functional Agile team. All team members were fully dedicated to the project, were required to have varied skill sets or be open to learning, and performed their work with complete transparency. The FedCLASS team was responsible for completing all aspects of each development task (i.e., user stories) through coding, test, integration, and build. There was no handoff to a separate testing organization.

Individuals in three key positions—the project sponsor, product owner, and project manager—served as critical change agents for both trying new IT technology and methods and establishing new relationships within the organizations involved. The team included a few outside consultants who served as coaches for using the new development practices and introducing new development technologies and tools.

The concept of fully dedicated team member assignments was a dramatic change for both the executive department and for the testing and development contractors, as it represented a significant innovation and discontinuity in their historic relationship.

Integration with the Data Center

The cross-functional FedCLASS Agile team integrated the business owners and contractors as well as functional development and test. However, the operations team responsible for managing the executive department's data center and deploying FedCLASS remained aligned with the operations function rather than aligned with the development team's processes. Initially, the deployment team's personnel were unable to adapt to the Agile practices used by the FedCLASS development team. However, strong leadership support from the department helped the two teams figure out ways to bridge the gap between the development team's Agile methods and the data center's traditional methods.

For example, the development team treated the integration of their project into the data center as a parallel project rather than an inherent and integrated aspect of their Agile and Lean development process, which allowed the deployment team to retain its traditional approaches. The development team developed a more structured listing of necessary work, allowing the deployment team to better understand the system requirements.

The deployment team also instituted some new approaches to support the deployment of FedCLASS. Previously, they established and maintained operational environments using manual, often labor-intensive, processes. During the FedCLASS Project's development phase, however, the development team demonstrated the value and benefit of using the systems integration framework Chef to build the FedCLASS operational infrastructure more automatically. The deployment team adopted these techniques from the development team, which reduced building the FedCLASS operational environment from a months-long effort to an hours-long effort.

Finally, the executive department's leadership met regularly with the deployment team's leadership to ensure effective communication and progress, identify barriers, and assign actions for follow-up.

Key Findings

This executive department chose to follow a new direction in creating FedCLASS, making a fundamental break with previous practices and technologies. To accomplish its goal, the department explored these broad areas of innovation:

- *A new role for management:* The business owner had extensive experience with LEGACY and was willing to take direct responsibility for achieving the primary business goal of the FedCLASS Project. In the past, that responsibility rested with the contractor and was mediated via a formal contract.
- *New technology:* Cloud-based development, a new programming language, new commercial products, and new development environments and tools were introduced to build a foundation for future growth. These new technologies and methodologies were not part of the existing skills in the department.
- *A new development team concept:* The department embraced Agile development team concepts, such as having dedicated team members who were self-organized and operating in a virtual team room.
- *New software methodologies:* The project piloted and experimented with using an iterative development lifecycle (RUP), Agile team management (Scrum), and Lean flow (Kanban) as

well as Agile technical practices derived from XP, including automated test-driven development and pair programming. The department used external coaches to provide new knowledge rapidly for the innovative pilot.

What started as an innovative pilot of new technology and approaches became a broad new transformation developmental effort that effected changes across the executive department. Key enabling factors for the transformation included

- a business owner who had professional IT skills and operational experience in the business area
- a program manager who had experience with new technology
- an environment of government-wide IT reform and a push toward new technology and methods
- a senior leader who was willing to try something different

Many factors and influences came together at the start of the FedCLASS Project that helped it succeed. They supported and reinforced each other, making it possible for the project to start down an uncharted path and become a model for change within a department of the federal government.

Abstract

This case study tells the story of the development of a critical IT system within an executive department of the U.S. federal government, using iterative, Agile, and Lean development methods and cloud-based technologies. This study reports the successes and challenges of using this new development approach in a government software development environment so that other government entities can benefit from the experiences of this project. The study is based on conversations with team members, observations of team activities, and examination of work products, documentation, and program guidance. The report describes the organizations responsible for creating the software solution, establishing the development process, and structuring acquisition activities. It then details the product development process in chronological order and describes the development approaches and technologies. It also puts events into the context of external environmental influences to present a development effort as it confronts real-world challenges. The final section describes insights gleaned during the research of this case study and includes analysis of the organization's experiences with Agile and Lean adoption, technical approaches, and project leadership. These insights may benefit future Agile projects in the federal government and the software engineering community as a whole.

1 Introduction

This case study tells the story of a software-development initiative undertaken by a key program within an executive department of the U.S. federal government. The program had outgrown a legacy software system of critical importance to the executive department in the performance of its mission. It needed a new software platform that would support future growth. Breaking with years of experience using traditional methods with its legacy system, the department chose to follow an uncharted path and pilot innovative methods and technologies to develop the new system.

In this case study, we document the department's decision to use iterative,[1] Agile, and Lean development methods for the software solution and its experiences with these new methods from start-up to implementation. We describe the approaches used during software development and illustrate the challenges and successes of applying these approaches in the context of the federal government. We also discuss the new culture of leadership that was fundamental to the successful introduction of these new technologies and development methods.

1.1 Purpose and Overview of This Case Study

The primary purpose of this case study is to inform other organizations about lessons learned, both positive and negative, so that they may benefit from this experience. It describes both the development of a new version of mission-critical software and the use of iterative, Agile, and Lean approaches in an organization that previously used more traditional development methods. Using these new development approaches affected not only the development phase but also every activity from contracting with external organizations, to evaluating and managing progress, to deploying the software solution. When the project was over, the department had developed new management roles, new software practices, and new concepts for development teams, while using new technology and tools. What began as an innovative pilot of new technology and approaches became a transformative developmental effort that effected changes across this executive department. Figure 1 illustrates the innovations and changes undertaken by the project.

[1] Throughout this report, we mention some software development terms related to this project. Appendix F provides definitions.

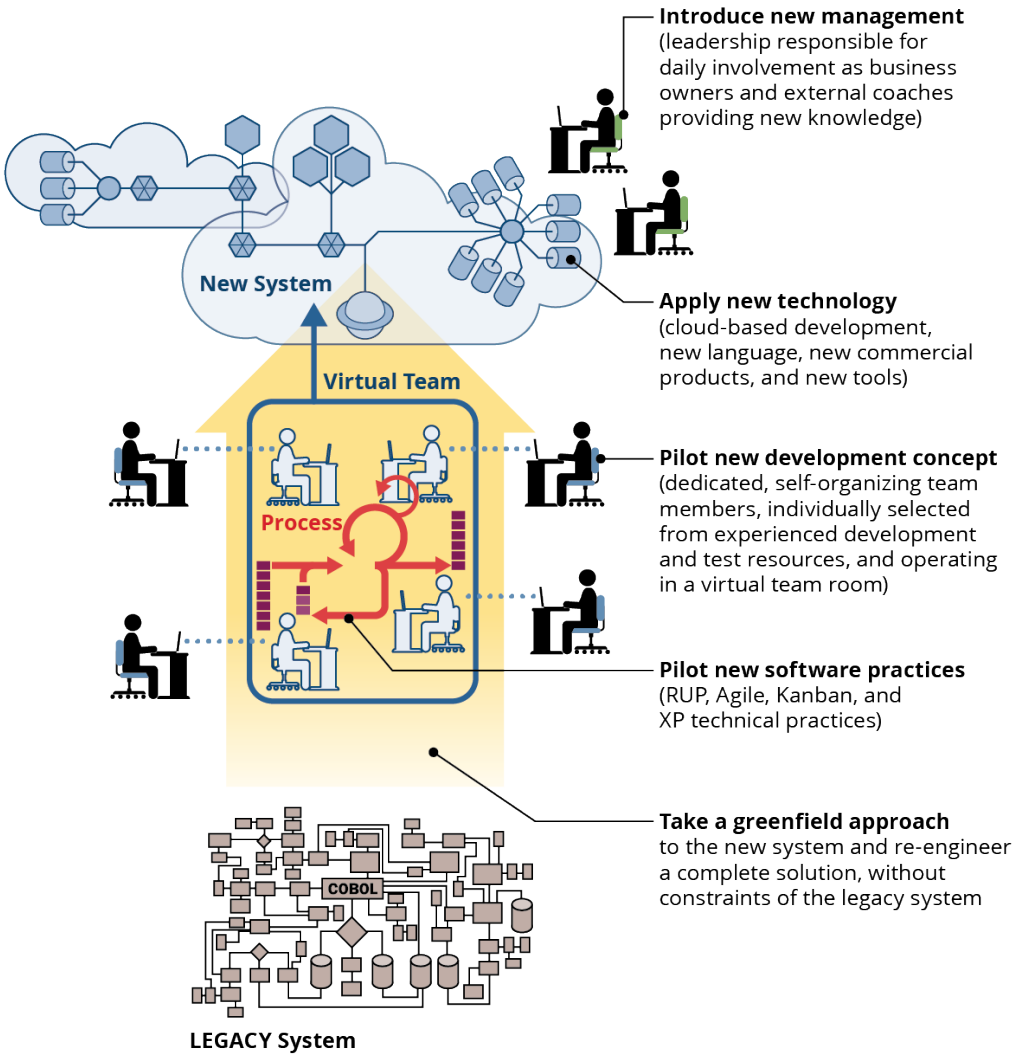


Figure 1: A Broad New Pilot

For this project, the executive department requested that the Carnegie Mellon University Software Engineering Institute (SEI) observe the development effort and provide guidance as needed in applying both new technologies and new software-development methods. The department wanted to shift to a "cloud-first" policy and use a new programming language and new commercial products. It also wanted to follow recent federal recommendations to break with traditional waterfall methods and instead apply management and technical practices drawn from iterative development, Agile and Lean, and Kanban.

We worked with this organization confidentially, and for that reason, we present this case study in an anonymized form. In addition to changing the names of participants and systems, we removed references to dates throughout the project lifecycle. We refer to the date of the project kickoff as "Kickoff," dates before that as Kickoff – *x* months, and dates after that as Kickoff + *x* months. The project timeline covered by this report spans Kickoff – 36 months to Kickoff + 35 months; the SEI's involvement spanned from Kickoff + 11 months to Kickoff + 35 months. This case study was constructed from extensive discussions with members of the software development

team, with other staff in the department, and with the staff of two external contractors; documentation reviews; observations of daily team activities; and analysis of work products from the development effort.

1.2 Organizations Involved in the Case Study

1.2.1 The Software Engineering Institute

The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University in Pittsburgh, Pennsylvania. The SEI helps advance software engineering principles and practices and serves as a national resource in software engineering, cybersecurity, and performance improvement. We work closely with national defense and federal government organizations, industry, and academia to continually improve software-intensive systems.

1.2.2 Department Omega

In this case study, we refer to the executive department as Department Omega. Department Omega has a broad mission to administer and enforce one category of federal laws and to ensure the reliability and security of one critical aspect of U.S. infrastructure. The secretary of Department Omega is a cabinet-level official, reporting to the president of the United States. The department has a strategic goal of maximizing its efficiency and expanding its scope. These goals are highly coupled as

- increases in efficiency allow for increases in scope
- increases in scope demand greater efficiency

One tactic the department has used to support this strategy is to seek new tools and methods to increase efficiencies. The software project that was the subject of this case study involved both new tools and new development methods.

1.2.3 Program Alpha

In this case study, we refer to the component of Department Omega responsible for the software development initiative as Program Alpha. Program Alpha is administered by Department Omega to perform a vital function of the U.S. federal government. It supports the mission of Department Omega by fostering efficient management of one of the primary resources that the department regulates. Program Alpha provides leadership and oversight of the information technology (IT) solutions used in performing its day-to-day work, such as the software developed during this case study. It also acts as the ultimate program manager responsible for balancing future investments in IT solutions with operational priorities and sustainment.

1.2.4 Service Providers

Two service providers have long-standing roles in support of both Department Omega and Program Alpha: a development contractor and a testing contractor. Neither contractor is a private contractor in the most common understanding of the term but rather stands somewhere between the public and private spheres. Department Omega and these contractors develop plans annually to define the scope and level of support that each contractor will deliver to Department Omega. Both contractors participated in the software development initiative.

1.2.5 The Development Team

A development team was established to carry out the development of the new software system. Including members of Program Alpha, the development and testing contractors, and consultants in iterative, Agile, and Lean methods, this team was cross-organizational and cross-functional. It planned to use new development methods to build new software systems to meet the department's future business needs.

1.2.6 The Deployment Team

The deployment team was drawn from a component of Department Omega that manages its data center and oversees its IT systems for compliance with security, configuration management, and deployment standards. While the development team created new software, the deployment team created new infrastructure at the data center. The new software would be deployed in the deployment team's infrastructure. And while the development team used iterative, Agile, and Lean methods, the deployment team used Department Omega's traditional methods until later in the deployment process, creating challenges when the new software neared its deployment phase.

1.2.7 Interrelationships

Figure 2 shows the relationships among the organizations associated with the new software project. The director of Program Alpha (the business owner) and the manager for IT development collaborated to oversee the project.

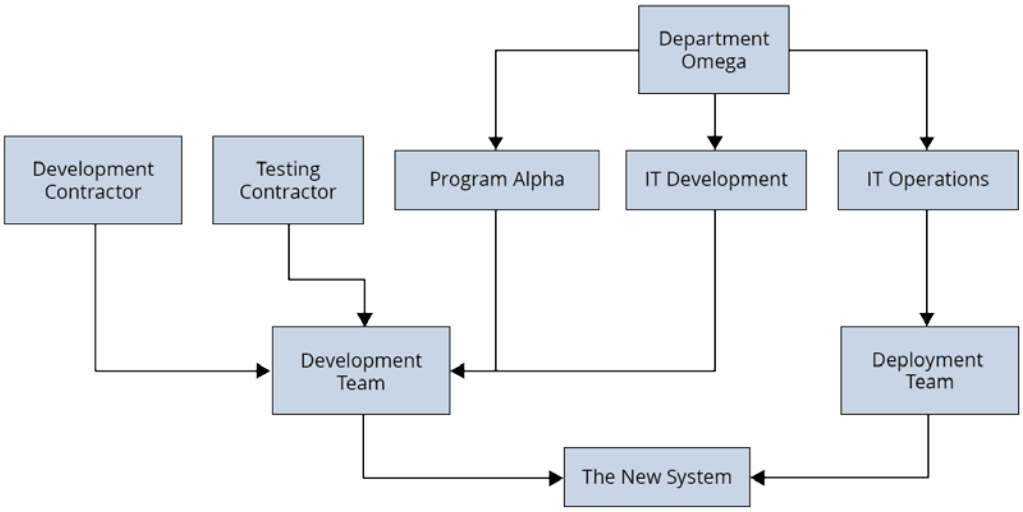


Figure 2: Interrelationships Among Organizations Involved in Developing the New System

2 Rationale for Change

This section describes the department's decision to develop a new system and move to new technologies and development methods. It also covers the factors driving this decision, including the inflexibility of the existing system, personnel shortages that strained maintenance of the existing system, and a desire to participate in government IT reforms.

2.1 The Inadequacy of the Existing System

The development contractor designed and built LEGACY in the mid-1990s with testing assistance from the testing contractor. LEGACY employed a combination of batch programs in Common Business-Oriented Language (COBOL) and, later, a web client on a Java 2 Enterprise Edition (JEE) platform. LEGACY was hosted in the operational data center administered by Department Omega.

As explained in a business case document prepared by Program Alpha, LEGACY had reached capacity in processing speed during peak processing times. The existing system could not easily accommodate changes in functionality. It was unable to grow in capacity sufficiently to continue supporting one of the goals of Department Omega—maximizing its ability to administer and enforce one category of federal laws. The inability to grow capacity in the system also impeded plans to expand services to support the needs of other federal and state agencies.

Program Alpha developed strategic plans to make multiple application improvements to LEGACY because it could no longer accommodate the growth goals of the program. These plans included

- increasing the maximum capacity of the system for processing data
- improving the correlation of data with data from other systems
- creating a single data repository for business intelligence queries
- providing better business intelligence for tactical decision making
- developing the ability to change business rules without code changes
- improving production and customer support

However, analysis of LEGACY's capabilities showed that the system could not support these goals without a huge investment in time, money, and personnel.

2.2 Personnel Shortages

Department Omega was formed from a consolidation of other government agencies, and the IT operations from those agencies merged into a single organization. This government-wide effort to consolidate data centers prompted Department Omega to decommission the data center that hosted LEGACY and to transfer its duties to another of its data centers in a different part of the country. This merger involved the decision to close the offices of one agency and move all operations to another agency. While personnel in the first agency were offered jobs in the second agency, many chose not to move—leaving the new, merged organization significantly under-

staffed. This personnel shortage severely affected the ability of Department Omega's IT Operations to overhaul the existing system or to stand up and operate a development and test environment for Program Alpha's new system in a timely manner.

2.3 Government Information Technology Reforms

IT plays a large part in federal agencies' ability to deliver products and services effectively and efficiently, and it offers immense capabilities to infrastructure and internal business systems. Unfortunately, IT system developments in the federal government are fraught with budget overruns and time delays in achieving initial operational capability. According to a July 2008 Government Accountability Office report, 48 percent of the federal government's major IT projects have been re-baselined at least once [GAO 2008].

From 2009 to 2011, the broad government oversight community generally recognized that the IT acquisition management processes were broken and the U.S. chief information officer (CIO) should aggressively change processes and policies. On December 10, 2009, the Federal CIO reported the following to the Federal CIO Council:

> *The government's management of information technology illustrates how a lack of enabling technology and transparency has led to poor results. Historically, the closed, secretive and compliance-based management approach, used to oversee more than $70 billion in federal IT investments, has not served taxpayers well. If an IT project was identified as being poorly planned or poorly managed, it was placed on a "Management Watch List," which was little more than a static PDF document on a website. This compliance-based approach was carried out behind closed doors with little evidence of improved performance. [Kundra 2009]*

In 2010, the Federal CIO issued a report calling for reform of federal IT management [Kundra 2010]. The plan outlined 25 areas that federal agencies needed to address to produce greater return on the government's investment in IT. Several of the 25 areas, or points, were highly inspirational to the development plans for Department Omega's new software system, including

- Point 3: Shift to a "cloud-first" policy
- Point 6: Develop a strategy for shared services
- Point 15: Issue contracting guidance and templates to support modular development

In addition, the White House issued a memorandum to the heads of executive departments and agencies on August 8, 2011, to clarify the primary areas of responsibility for agency CIOs across the government, as identified in the Federal CIO's 25-point plan for implementing IT reform. Issues in the reports included (1) the lack of usage of enabling technologies, (2) poor IT acquisition management processes, and (3) the failure to recognize that IT is and will be the principal enabler for advanced capabilities needed by customers.

2.4 The FedCLASS Project

To indicate the department's intention to address key points in the Federal CIO's call for reform [Kundra 2010], we refer to the project to replace LEGACY as FedCLASS (Federal Cloud-based Lean and Agile Shared Services). The change factors described in Section 2.3 influenced key decisions on the FedCLASS Project: developing a new system rather than overhauling the old system, overcoming personnel shortages, and participating in government IT reform.

2.4.1 Inadequacy of the Current System

As described in Section 2.1, Department Omega determined that continuing to enhance LEGACY would not support future business needs and decided to engineer a new solution. The department originally envisioned the FedCLASS Project as a major overhaul of an existing system—not the creation of an entirely new system. However, once the project was underway, it became apparent that very little of the old system could be salvaged for incorporation into the new system. Department Omega did, however, continue to maintain and operate LEGACY until it was replaced by the reengineered functionality and enhanced capability of FedCLASS.

2.4.2 Personnel Shortages

Data center personnel shortages were one of the factors that drove the FedCLASS team's decision to use a public provider of cloud services to host its development and test environments. The use of cloud services allowed the development team to begin development almost immediately, without the lag time associated with acquiring and building a development environment.

2.4.3 Government Information Technology Reforms

In addition to personnel shortages, the Federal CIO's report also drove the team's decision to utilize cloud computing. Though initially the use of the cloud environment was for development and test purposes, the project team hoped that Department Omega would also permit a production deployment to the cloud.

In addition to cloud computing, the Federal CIO's report was cited as a strong rationale for trying a different path than Department Omega's traditional Rapid Application Development (RAD) software development process. The RAD approach focused on delivery at the end of development, whereas the new approaches from the government's IT reform initiative focused on delivering working software early and throughout development. This drove the decision to pilot alternative development approaches including iterative, Agile, and Lean on the FedCLASS Project.

3 Project Initiation

This section describes the initial stages of planning and exploration for the FedCLASS Project. It includes determining project scope and requirements, assessment of the LEGACY system and its fit with FedCLASS Project needs, and selection of a development approach.

3.1 Key Events in the Project Initiation Process

This section describes key events from project initiation to the launch of the development effort. Figure 3 shows where the project initiation events fit into the development timeline (see Appendix B for a complete timeline). While the FedCLASS Project is the focus of this case study, it was executed in the larger context of Program Alpha's long-term IT strategy, and the capabilities developed for FedCLASS were designed to align with its future role in that overarching strategy.

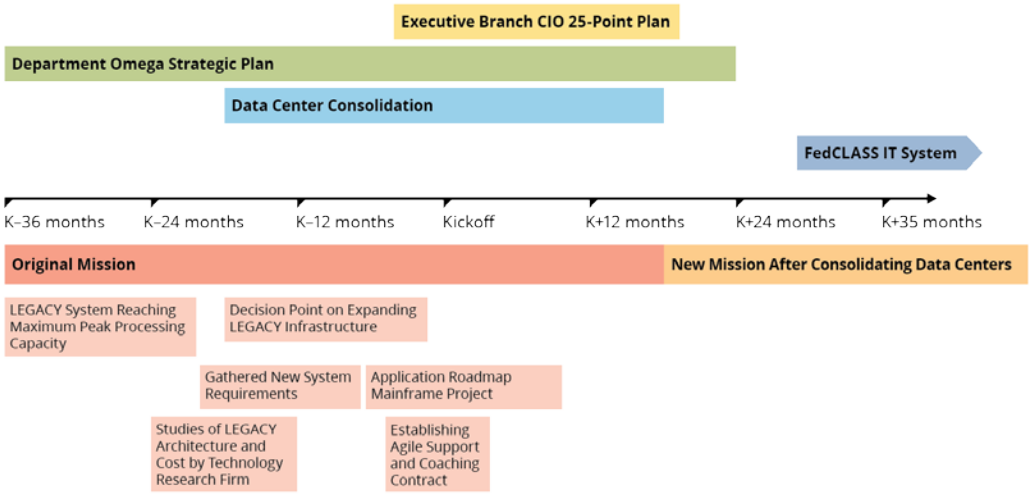


Figure 3: Key Project Initiation Events

3.1.1 Determining Software Requirements

Three years before the kickoff of the FedCLASS Project, LEGACY was reaching capacity in processing data during peak periods. This triggered a series of actions to identify and document the department's business needs for future growth capability. Requirements gathering and planning for the revision of the old system began at Kickoff – 18 months. By Kickoff – 6 months, Program Alpha had determined its system requirements for new capabilities. A spreadsheet summarized the program's new software requirements; included drivers and goals, business requirements, technical requirements, and constraints; and noted the source for each requirement statement. According to the Statement of Need in the project charter,

> *[The Program Alpha] system has reached capacity in processing [data] during peak periods due to its lack of flexibility to accommodate changes in the system. This [affects other state and federal] agencies, [Department Omega's other] systems, budgets for projects and pro-*

duction support as well as [Department Omega] operational personnel. The impact is a reduced operational window that impairs [Department Omega's] ability to [perform its mission] in a timely manner.

3.1.2 Assessing the Architecture

In parallel with the requirements gathering, a number of studies and analyses were performed to investigate options for improving LEGACY. Department Omega hired a technology research firm to conduct two architecture studies to examine alternatives for technical solutions that included a range of options from continuing to operate and enhance its application services to using commercial service providers. One of the studies was initiated at Kickoff – 24 months and one at Kickoff – 17 months.

Architecture Analysis 1

The technology research firm was hired to complete a case study to gain insight into how similar organizations handled the kind of business operational systems that support Program Alpha's software. The case study provided insight into alternative approaches to meeting Program Alpha's IT infrastructure requirement for growth in processing capacity. The *Program Alpha Architecture Review Case Study* was delivered at Kickoff – 20 months. A key part of the study was a comparison of the program's architecture approach with market research on similar batch workloads. The study's findings helped to shape the future direction for the program's infrastructure.

Architecture Analysis 2

After its architecture case study, the technology research firm was again hired to perform a cost assessment and benchmark study of Program Alpha beginning in Kickoff – 17 months. During this study, which lasted from Kickoff – 17 months to Kickoff – 12 months, the technology research firm conducted another architectural assessment and analyzed four alternatives.

3.1.3 Aligning the Development Effort with the Organization's Strategic Plan

A presentation—*Project Concept FedCLASS*—was prepared to gain the support of Department Omega's IT Governance Board for the FedCLASS Project. The presentation identified key points of contact for the project, high-level business needs that the project should meet, and desired outcomes for the project. The presentation also tied the project to the Department Omega Strategic Plan, spanning three years before and two years after the project kickoff, and made a specific connection to "Strategic Goal 3: Maximize [performance of Department Omega's mission]; Strategy: Seek new technologies to streamline, modernize, and improve business processes and systems."

3.1.4 Attempting to Reuse the Old System

To prepare to meet Program Alpha's future needs for IT infrastructure, Department Omega worked with the development and testing contractors on a number of studies and exploratory projects. Some of this work involved attempts to use LEGACY in ways that would overcome its weaknesses. The contractors, in partnership with Department Omega, started a project to move the old COBOL application to the mainframe as a first step to addressing performance and database deadlock issues. The direction to move to the mainframe was, in part, the recommendation of Architecture Analysis 2, completed by the technology research firm mentioned in Section 3.1.2.

Studies demonstrated that running significant components such as data correlation on the mainframe confirmed a gain in performance and a resolution of the deadlock issues. The development contractor proposed a rewrite "as is" of LEGACY in Java over three years, including moving it from UNIX to the mainframe. Department Omega's database administration group and its resources were involved in the proposal to move LEGACY to the mainframe and provided technical assistance and review.

Department Omega reviewed that proposal and a prototype but decided not to move forward with the project for several reasons:

- The development contractor's proposal solved some of the problems but was informally projected to cost $40 million and take four to five years to complete. Department Omega considered that estimated cost too high for the business value to be achieved from the project.
- The proposed solution would put the rewritten application on the same computing platform as other key Department Omega applications that were experiencing production issues.
- The proposal did not fully address business requirements for future flexibility and expandability.

3.1.5 Planning for the Future

By Kickoff – 12 months, Program Alpha faced the decision of how to address the needed expansion of its infrastructure capability. Its original software had grown through continual enhancements and additions, and the original U.S. Code requirements had expanded the applicable federal laws over the first 15 years of the Program Alpha system's operation to include providing services to state agencies and individuals. The consensus among Department Omega staff was that continuing incremental enhancements of LEGACY—a batch-job-based, 20-year-old system—would not achieve the goals of supporting the ever-expanding requirements of the department's mission through new laws and future business needs. LEGACY, which could not handle the increasing processing volume and throughput speed, was judged to lack the flexibility to accommodate future changes in the system. As we noted in Section 2.4, the goals of the FedCLASS Project were to enable additional sources of data, increase data volume, and improve the effectiveness of data correlation.

3.1.6 Changing the Development Approach

After performing an analysis of alternatives, Department Omega decided to develop a completely new solution to the problems facing LEGACY and named it the FedCLASS Project. The organization decided to take a greenfield approach to the new system and start with a blank slate, without any constraints imposed by prior work. The primary business driver was to achieve a solution that provided flexibility to expand and change to meet future needs.

Within the context of broad government IT reform, Department Omega decided to pilot the use of innovative technology, software methodology, and development team concepts and to take on new management responsibility and relations. This pilot would include developing a major rewrite of LEGACY as a Java solution and using cloud technology for the development environment, in keeping with the Federal CIO's recommendation "Point 3: Shift to a 'cloud first' policy" [Kundra 2010]. In addition, the new effort would follow a tailored Agile approach that used risk-

focused RUP techniques as the basis of structuring the development team and managing the project.

In another significant break with traditional system development, Department Omega took direct responsibility for managing the project and for integrating business owners in the development process. Department Omega hired consultants, according to the contracting actions described in Section 4.1, for guidance on how to help its staff get directly involved using innovative management approaches and new technologies and tools.

3.2 Adopting New Development Practices

As part of project initiation, Department Omega reviewed its strategic goals for IT development, in particular the goal to seek new technologies to improve business processes and systems and the goal to act on the Federal CIO's report calling for reform of federal IT management [Kundra 2010]. For these reasons, Department Omega committed to incorporating iterative and Agile practices into its software development approach in order to improve efficiency and accelerate delivery.

Another driver for adopting new processes was Department Omega's desire to have deeper engagement with and responsibility for the FedCLASS Project. The department thought that the deeply collaborative, cross-functional teams used in Agile methods would facilitate this relationship shift.

Although Department Omega saw advantages in adopting Agile methods, it initially decided to institute an iterative RUP-based approach on the FedCLASS Project and hired a coaching firm with experience in this methodology. RUP is not typically considered to be an Agile approach, as it is based on defined lifecycle stages and uses considerably more formal documentation than Agile methodologies. The initial coaching contract called for the delivery of

- use cases
- a Microsoft Project plan
- formal test case documentation and reporting
- formal business rule documentation
- a formally defined quality assurance (QA) process
- architectural documentation (in line with the RUP view-based approach)

While not an Agile methodology, the iterative approach of RUP represented a move to incremental delivery as opposed to the waterfall approach previously employed by Department Omega. RUP's risk-based, architecture-centric perspective was also felt to be well-suited to address the needs of the FedCLASS Project. Although using RUP as a methodology, the project was committed to incorporating Agile values of collaboration and self-organization as well as selected Agile practices such as daily standups, the definition of done, and the product owner role.

4 Establishing the Team

Before development of the new system could begin, the department needed to acquire technical expertise, establish roles and team structure, and conduct training. This section describes these activities.

4.1 Contracting for Technical Expertise

Department Omega's IT development group recognized that to be successful they would need to augment the development and testing contractors' skills and knowledge of iterative and Agile software development methods and cloud-based technology. The process of finding and acquiring these skills and knowledge began before the project kickoff.

Contracting has sometimes been a problem area for government organizations that want to try iterative, Agile, Lean, or other nontraditional approaches. Department Omega follows the standard *Federal Acquisition Regulation* (FAR). Agile development calls for valuing "customer collaboration over contract negotiation" [Agile Manifesto 2001] and assumes the capacity for performers to respond rapidly to change. Department Omega's IT development group performed pre-work to resolve this process disparity so that contractor teams could use Agile methods and would not have to stop work mid-project to wait for new or renegotiated contracts. This pre-staging of flexible contract support would allow the FedCLASS Project to draw on special expertise quickly. With contract vehicles in place, task orders could be placed on contracts whenever special support was required.

Major contract actions required for supporting the project's nontraditional approach to software development included the following:

- contracts for coaching and technology transition support
- contracts for development tools
- contracts for a cloud service provider
- contracts for expert services, such as independent estimates and special tool support
- other contracts as required to rapidly address the needs of the development team

The following contracts and agreements were established for the FedCLASS Project:

- *coaching contractor:* This contractor would provide coaches to support the development team in using RUP and Agile development practices and train the team to work with new development technologies and tools. The contract was a direct award to a minority-owned business that provides professional services in IT areas. It was a firm-fixed price for services, with a not-to-exceed cost for reimbursable travel and other purchases, such as software, hardware, and licenses. The contract was based on a performance work statement.
- *external development and testing contractors:* The development contractor would provide software developers, system architects, and database design. The testing contractor would provide software testing services for the new system. The agreements with these entities were not contracts in the legal use of the term. Department Omega had a long-standing relationship

with these contractors established by statute. Annual plans were used to define the scope and level of support from each contractor.

- *Department Omega's IT operations:* The Omega data center was responsible for creating the infrastructure to support the new system and for performing deployment to production. The agreement for data center hosting of the FedCLASS production system was also not a contract in the legal use of the term. It was a service-level agreement established between Program Alpha and Department Omega's data center.

4.2 Creating the Development Team

After these foundational tasks were completed, a development team was established to carry out the FedCLASS Project. The team adopted roles and organizing concepts from two widely used methodologies. From the RUP approach, the FedCLASS Project adopted the practice of grouping work in development increments based on risk. From the Scrum approach, the FedCLASS Project adopted the concept of dedicated team members working on a self-directed development team and the roles of product owner and team coach.

The concept of assigning fully dedicated team members was a dramatic change for both Department Omega and the development and testing contractors. The traditional approach was to matrix an individual with a specific skill onto multiple projects at the same time. Department Omega also historically assigned development projects fully to the one of the contractors. The contractor's leadership would then take responsibility for managing the project and accomplishing daily tasks. The use of a dedicated, empowered, and self-organizing team within Department Omega represented a significant innovation in the relationship between the department and these contractors. Figure 4 shows a hierarchical view of the development team.

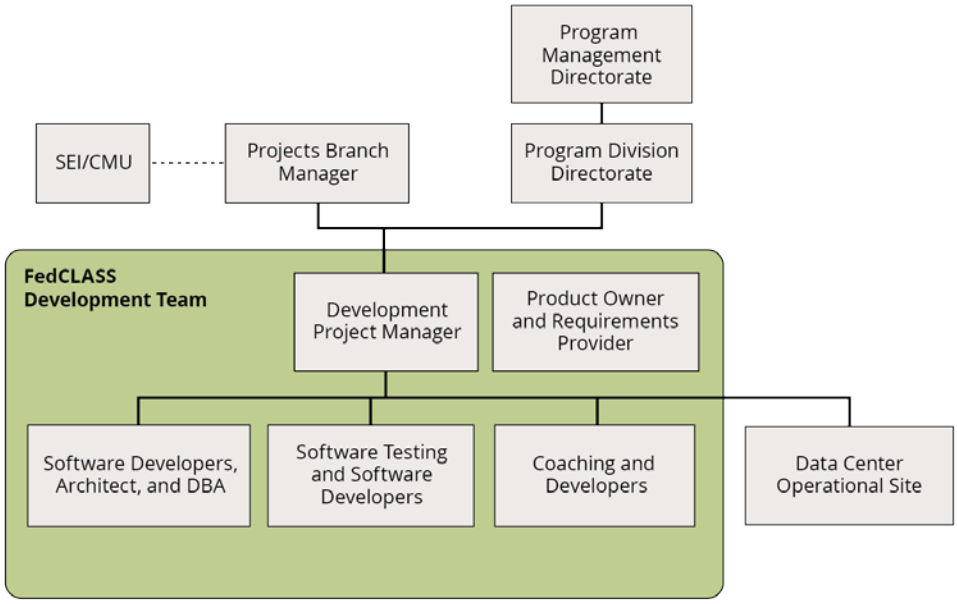


Figure 4: Hierarchy of the Development Team

4.2.1 Team Structure and Roles

The FedCLASS Project had to address the needs of the project owner and sponsor, the interfacing agencies (stakeholders) who would use the new system, and Department Omega's data center for hosting the production applications. Although Figure 4 shows a hierarchical relationship, the project did not function on a day-to-day basis according to the traditional hierarchical structure. The team operated consistently with Agile and Lean development methodologies. Members completed work by collaborating in cross-functional teams, based on integrated development that included requirements (user stories), coding, testing, and deployment.

The broad roles within the dedicated cross-functional team, shown in Figure 5, included

- project sponsor (not shown), who provides direction and access to resources
- product owner, who has the responsibility of the business owner
- project manager, who has the responsibility of overseeing the development of the system
- federal and state agency stakeholders, who interfaced with the development team through the product owner
- members of the development team, including an architect, developers, a database administrator, and testers from within Department Omega and from integration contractors
- coaches and technology subject-matter experts (SMEs)

These broad roles are consistent with the Scrum approach to management concepts. We describe each role in more detail below to provide insight on how the development team accomplished work and delivered functioning software capability to the Program Alpha business owner.

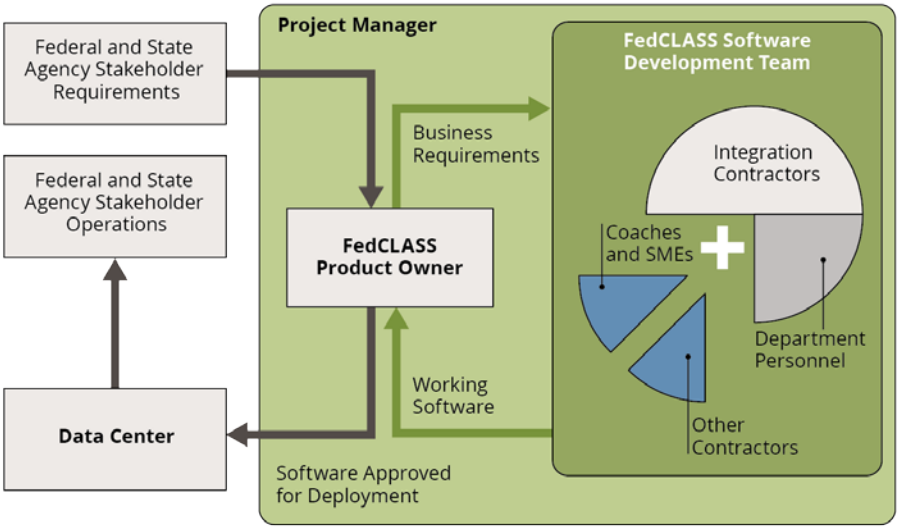


Figure 5: Team Role Interfaces of the Development Effort for the New System

4.2.2 Project Sponsor

At the highest level in Department Omega, the project sponsor was the Assistant Commissioner of the department. The project sponsor was not involved in the daily management of the FedCLASS Project but provided funding, resources, and political support for the team at a higher level.

4.2.3 Product Owner

The Agile Scrum methodology defines a role called product owner. “The product owner has responsibility for deciding what work will be done” [Baker 2014]. At the start of development, the role of product owner was assigned to the director of Program Alpha, who was responsible for the performance of Program Alpha. The product owner was supported by members of Program Alpha’s operational staff, who were assigned the role of product owner delegates and were involved in the business aspects of the program’s day-to-day operation.

The product owner, with support from the product owner delegates, maintained the business requirements for FedCLASS in what Agile methodology calls the product backlog, “an ordered list of everything that might be needed in the product” [Schwaber 2013]. One of the product owner’s primary functions in an Agile environment is to ensure that the product backlog is organized and ranked by business value. The development team’s responsibility is to work on the items of highest priority in the product backlog, in accordance with the business owner’s priorities.

4.2.4 Project Manager

The role of project manager was assigned to a technically experienced federal employee of Department Alpha, who was responsible for overseeing the development of the system. Activities included overseeing the development team, addressing roadblocks, and reporting to the product owner and upper management.

Together, the project sponsor, project owner, and project manager served as key change agents [London 1988] for both trying new IT technology and methods and establishing new relationships among the organizations involved. The three roles and the individuals filling those roles remained key influences in the move to adopt new and innovative approaches to performing IT projects.

4.2.5 Federal and State Agency Stakeholders

Program Alpha focuses on providing services to the federal and state agencies. With this broad customer base, the program has a large number of data partners who depend on Program Alpha’s services for their organizational health. The existing stakeholder communities of Program Alpha participated in the transition from LEGACY to FedCLASS. The goal of developing the new system was to retain the existing application interfaces and limit any impact on the external data partners.

4.2.6 Team Members

The development team included 16 people:

- 4 members from Department Alpha, including the product owner and project manager
- 5 members from the development contractor, including software developers, system architects, and database designers
- 2 members from the testing contractor, which provided testing services and software developers

- 5 coaches and SMEs on commercial software (e.g., Oracle[2]) from a coaching contractor, who served as coaches on using Agile and Lean development practices and introducing new development technologies and tools

The development team operated as a cross-functional team, in which each member was aware of every role and team members contributed based on the needs of the work to be performed. This required each team member to have varied skill sets or at least be open to learning new skills. The team controlled the work through user stories that were placed in a backlog. Initially in accordance with RUP, these stories were prioritized by architectural risk. Later in the project, prioritization shifted to focus on business value as determined by the product owner. The team members worked together with complete transparency. They did not "hand off" user stories within the team. Once a team member took a story from the backlog, he or she handled it until it was accepted by the product owner or placed back in the backlog. All stories that had been started in the work flow were addressed in every daily standup until they were accepted. The output from the development effort was working software and supporting information.

A significant number of team members who started with the dedicated FedCLASS development team had experience with LEGACY. At various points in the team's life, new members and coaches were added to support the development needs or to replace individuals who moved to other commitments. With additional support for testing, the team size grew to approximately 20 members by Kickoff + 18 months.

Formation of the development team began at Kickoff – 2 months. When Department Omega chose to directly manage this team, it set aside past practices for personnel assignment. In the traditional process, Department Omega and the development and testing contractors assigned people and managers to the work project. Now, candidate team members were selected from across the United States through interviews. Criteria for team selection included candidates' openness to the use of Agile methods.

4.3 Training the Development Team

At the kickoff, team members attended the first team meeting via teleconference and discussed the travel logistics for starting the work. It was the first session of a three-month training period and team-forming experience. Appendix D contains a list of the training events and activities. The team's first in-person meeting occurred at the development contractor's facility at Kickoff + 11 days.

The 16 individuals tentatively selected for the FedCLASS Project were instructed in the concepts of Agile practices and forming a dedicated team working in a common team room. (See Section 4.2.6 for information about the makeup of the development team.) All team members were fully dedicated to the FedCLASS Project and were physically co-located for three months. Later, a virtual team room was established, enabling all team members to work collaboratively. By the time the project was up and running, the virtual team was connected all day via video teleconferencing

[2] Throughout this report, we mention some of the tools used by the development team. Appendix C describes their typical uses.

from 9:00 a.m. to 7:00 p.m. Eastern Time to allow for collaboration that crossed four time zones. The daily team standup occurred at 11:15 a.m. Eastern Time.

As team members got to know each other, they also began to learn the practices for working in an Agile environment. The team received training in Agile practices and the RUP through an Agile Development Simulation Workshop. Additional training was provided during the first iteration of the Inception Phase, including modules that covered iterative development, phases and iterations, daily standup concept and content, releasing iteratively, and estimating effort.

5 Project Implementation

The development team initially used a RUP lifecycle with multiple phases and iterations and planned for three releases: two internal releases and a third release for external deployment. The RUP approach was blended with Agile practices, primarily drawn from Scrum. As the project progressed, the RUP approach was discontinued and the team adopted the use of Lean and Kanban. Release plans were also modified as the project progressed.

5.1 Starting with RUP

This section summarizes the initial RUP-based approach. The RUP process has four phases, each with a different focus [RSO 1998]:

- Inception Phase, focused on establishing project viability and mitigating business risk
- Elaboration Phase, focused on establishing a technical approach and mitigating architecture risk
- Construction Phase, focused on developing a usable solution and implementing risk mitigation
- Transition Phase, focused on successful release and mitigating deployment risk

Each phase contains a varying number of iterations in which to accomplish the work. An iteration is a set of activities with a plan and evaluation criteria, comprising a complete development cycle, from requirements gathering to implementation and testing. The coaching contractor defined an iteration as "like a small project producing a stable integrated software product, assessed to evaluate its success in meeting the clear objectives defined at iteration start."

The technology transfer contract with the coaching contractor included specific training and support on iterative development concepts and risk-driven development in the context of RUP phases. To help organize and train the team, the initial coaching contract included creating an iteration plan by following a template.

From Kickoff + 2 weeks to Kickoff + 10 months, the process in use was largely based on RUP, although the team also incorporated some Agile Scrum-based practices. The RUP-based Inception Phase of approximately seven weeks consisted of 5 iterations of varying length. This was followed by a RUP-based Elaboration Phase of approximately 7.5 months, consisting of 14 iterations of varying length.

The process used during this period was risk centric in keeping with the RUP approach to development. Nonfunctional requirements were elicited, prioritized, and captured in an architectural concerns survey. The development team defined risks to mitigate and tracked them in each of the 14 iteration plans of the Elaboration Phase. The team captured requirements using use cases. Elaboration efforts focused on constructing and proving a viable candidate architecture. The team pursued Pattern-Enabled Development as a design approach, pushing functionality into the architectural framework with commensurate constraints on the design latitude of the development team.

5.1.1 Release 1: Inception Phase

The Inception Phase consisted of five iterations from Kickoff + 2 weeks to Kickoff + 3 months. During this phase, iterations were scheduled for time periods of varying length, and sometimes there was a delay between the early iterations. The development team started working as a dedicated team with training woven into the work iterations. The team created a plan associated with each iteration of the Inception Phase to guide the work. The goals for this phase included understanding project objectives, choosing tools, setting up the development environment, and creating release plans.

During the first iteration, the development team worked with an external coach and identified 73 risks related to FedCLASS, 45 current system problems, and 24 desired outcomes with evaluation criteria. The team also started identifying stakeholders for the FedCLASS Project and holding meetings with them to settle on project outcomes.

During the Inception Phase, the team accomplished the following tasks:

- reviewed the business requirements, identified stakeholders, and agreed on the scope of the FedCLASS solution
- identified architecturally significant requirements and defined a candidate architecture
- identified and installed multiple project development and tracking tools
- developed functional and nonfunctional test ideas, strategy, and candidate tools
- received training on
  - use cases, Pattern-Enabled Development, software architecture principles, and development approaches
  - estimating and project-sizing techniques, including planning poker, domain analysis, and use-case points
- started release planning based on the identified technical risks
- visited cloud-service providers
- began using a fail-fast approach[3] to trying and learning new ideas

The fifth and final iteration in the Inception Phase of Release 1 ended at Kickoff + 3 months. By that time, team members had set up cloud-based environments for development and QA and chosen software development tools. They determined means and methods to measure acceptable quality levels, agreed on the estimated size (i.e., effort) for the project, and created a release plan with initial content for each release.

5.1.2 Release 1: Elaboration Phase

In the Elaboration Phase, iterations were scheduled for different time periods, and sometimes there was a delay between the early iterations. Release 1 was planned as the output of the Elaboration Phase. It was intended as an internal release and not for external deployment. Figure 7 illustrates the team's release plan.

[3] "Failing fast is a non-intuitive technique: 'failing immediately and visibly'" [Shore 2004]. The team used this idea in the context of trying something new to learn what works by doing.

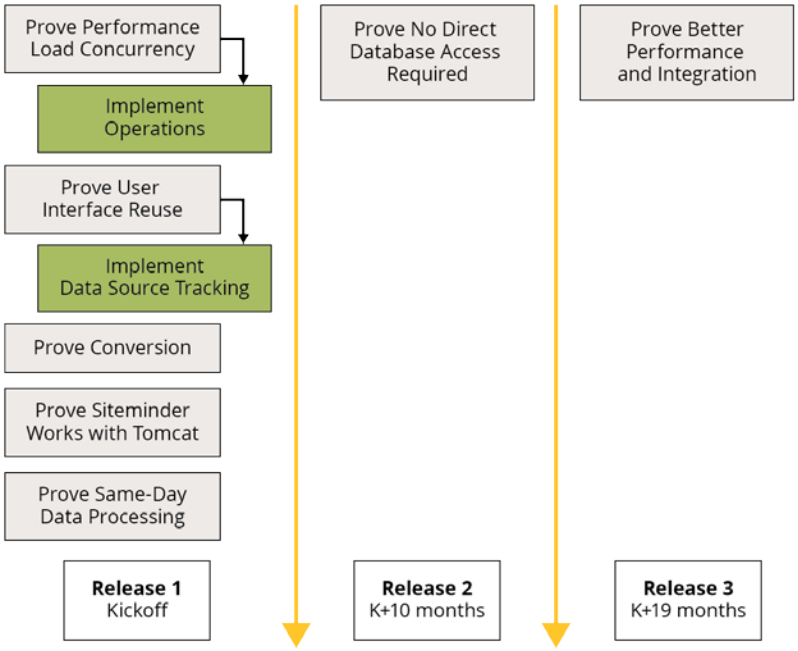


Figure 6: Release Plan Diagram

Release 1 focused on addressing technical risks in two areas: (1) those identified as high risk in relation to how LEGACY performed and (2) those related to technology for the new FedCLASS architecture. Figure 7 illustrates the iterations of Release 1 and its fit within the overall project timeline.

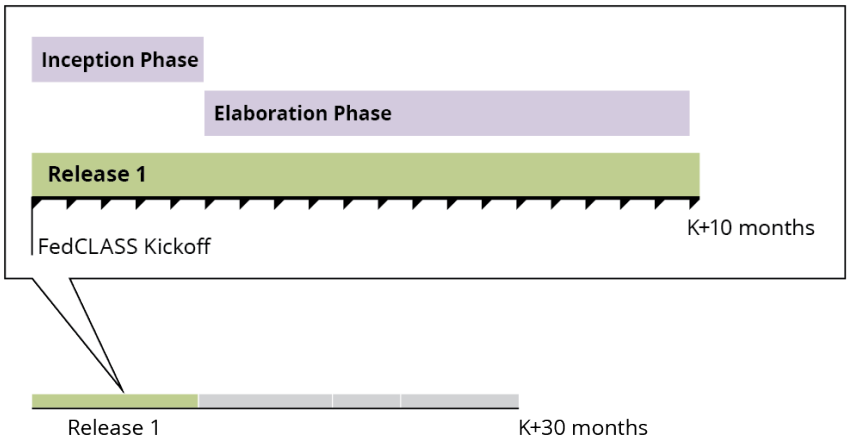


Figure 7: Release 1: Establishing a Candidate Architecture

The Elaboration Phase of Release 1 consisted of 14 iterations of varying length that ran from Kickoff + 3 months to Kickoff + 10 months. As in the Inception Phase, the development team created a plan associated with each iteration of the Elaboration Phase to guide the work.

Iteration 1 focused on proving that the team could implement and test a significant scenario in the cloud-hosting environment using the candidate architecture. The team detailed all the requirements with the scenario, wrote the code, and created more than 70 unit tests. The code was executed in the cloud (QA environment) with only one outstanding Severity 1 defect.

During the second iteration, the final development team was chosen based on the skill sets needed for the project, including those for team compatibility. The initial team was formed during the Inception Phase, with the collaborating organizations selecting the members. After team members worked together during the early iterations, they determined that the team was too large, so only a subset of the initial team moved forward into the next phases.

Tasks for the Elaboration Phase were selected in accordance with the architecture-centric, risk-based approach of RUP. During this phase, the team accomplished the following:

- identified, prioritized, and validated architecturally significant and high-risk business requirements and use case scenarios. These included scalability, external interfaces, performance, data conversion, and reuse of the LEGACY graphical user interface (GUI).
- performed design, coding, and testing to address the prioritized high-risk requirements and scenarios. The team proved feasibility in the following areas:
  - proved it was possible to integrate the LEGACY GUI with the new FedCLASS architecture
  - demonstrated that the system could perform primary operations concurrently and during updates without locking issues (a problem in LEGACY)
  - demonstrated that FedCLASS could process files under data-volume conditions that matched reasonable near-future projections.
  - made a significant breakthrough that resulted in the system responding much faster than required for large files (51 minutes to process a 3-million-record file; the requirement was 180 minutes). The development team determined that multi-threaded processing for key steps maximized the use of CPUs for parallel processing.
- began work on establishing an automated test infrastructure with a focus on performance and scalability testing. As part of this work, the team successfully generated random, production-like data for testing in the cloud-hosted environment to address security concerns about personally identifiable information (PII).

By the end of the Elaboration Phase, the team had demonstrated that the chosen architecture supported scalability requirements and met the stakeholders' projected data volumes. The team also integrated the LEGACY GUI with the FedCLASS architecture, which would allow the business owner to reuse that GUI if necessary or desired.

5.2 Moving to Lean and Kanban

After the end of the Elaboration Phase and the completion of Release 1, a team meeting was held at the testing contractor's office to assess progress and plan the next release. At this point, the development team had about eight months of experience working as dedicated team members from four separate organizations, meeting daily in a virtual team room. While establishing a cross-functional, self-directed team was the goal, the group had not yet reached that level of performance. This meeting turned out to be a key event in the evolution of this team, because they decided to switch to new coaches and a new coaching style.

The new coaching team brought with it a different coaching approach to coordinating development and helped the team begin applying Lean thinking and Kanban techniques. The coaching

style moved from a command-and-control, directive style to a style focused on encouraging new behaviors and helping team members address roles and responsibilities in a new way. The team also addressed topics related to release planning, determined the definition of done for development, and defined requirements on the data center environment for the operational FedCLASS software and architecture.

Other key changes included the following:

- moved from an emphasis on risk-centric, architecture-centric development to an emphasis on delivering functionality with business value: The team eliminated formal risk tracking and embraced an emergent approach to architecture.
- moved away from Pattern-Enabled Development: A degree of functionality was extracted from the architectural framework.
- moved to a Kanban-style, continuous-flow model for development: The team maintained fixed iterations of three weeks in length but primarily used them to provide a cadence for demonstrations to the product owner. The team did not implement the Scrum practice of committing to a fixed set of user stories for a given iteration based on team velocity.
- discontinued planning poker and the use of story points for estimation
- moved from use cases for requirements specification to user stories and test scenarios with Cucumber, a tool for running automated acceptance tests: The team emphasized requirements that emerged from conversations and collaboration among the product owner, developers, and testers. They also adopted an acceptance test–driven development (ATDD) approach.
- adopted pair programming and refactoring as developer practices
- implemented continuous build, integration, and test: The team implemented on-demand environment configuration and deployment.
- generated automated metrics for code and design quality on a nightly basis
- shifted from story prioritization based on risk to prioritization based on business value

The Agile and Lean practices adopted by the development team are summarized in Table 1. The practices come from Scrum, Kanban, and XP.

Table 1: Agile and Lean Practices Used by the Development Team

Scrum Practices	Kanban Practices	XP Practices
Product backlog Standup meeting Retrospectives	Visualize the work Limit work in progress (WIP) Manage flow Evolutionary change Improve collaboratively Evolve experimentally Standup focusing on stories rather than individuals	ATDD Pair programming Refactoring Continuous integration Collective code ownership Simple design Sustainable pace User stories Collaborative work space On-site customer

The development team continued to use the Lean and Kanban framework as its approach throughout all subsequent releases and deployment to production.

5.2.1 Release 2: Developing Core Functionality

Release 2 consisted of 11 iterations that ran from Kickoff + 10 months to Kickoff + 19 months. The team focused on developing user functionality as defined in the user stories and writing the corresponding Cucumber tests.

Figure 8 illustrates the iterations of Release 2 of FedCLASS and Release 2's fit within the overall project timeline.

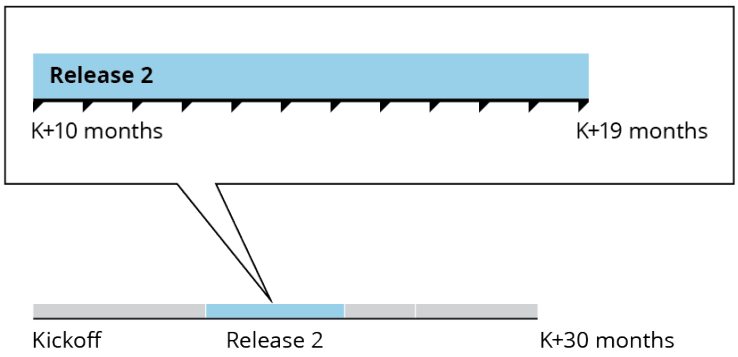


Figure 8: Release 2: Core Functions

During Iteration 1 of Release 2, the team started building new functional scenarios. Functions added during this iteration included validating data format, recording data receipt, status of receipt, and completion of schedule. In addition, the team restructured the Maven project configuration to include separation between code for the GUI and code for FedCLASS. They also fixed existing code with optimistic locking of transactions, a database technique for avoiding update collisions resulting from simultaneous updates to the same data by two concurrent users.

With the move to Kanban, the team stopped using the iteration plan template. The 11 iterations in Release 2 were moved to a standard three-week cycle with a Wednesday start and Tuesday end. During Release 2, the team implemented an extensive set of user and technical stories. User stories covered both GUI and batch operations. Technical stories covered quality attributes (e.g., performance testing), modifications to the application architecture (e.g., investigating Oracle TimesTen), and development infrastructure improvements (e.g., beginning to use ATDD and the Cucumber tool and establishing a development staging environment within the data center).

5.2.2 Release 3: Developing Additional Core Functionality

Release 3, shown in Figure 9, consisted of seven iterations that ran from Kickoff + 19 months to Kickoff + 23 months. The development team continued developing functional requirements and preparing for deployment and operation. The development team interfaced more with the deployment team and focused on the readiness of the data center's operational and disaster-recovery environments. (Note that the iteration numbers continued to increment even though the release shifted to Release 3.)

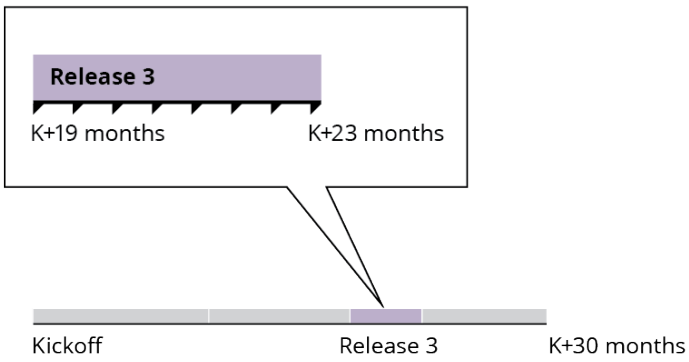


Figure 9: Release 3: Core Functions

During the first iteration of Release 3, Iteration 12, the team completed functionality for determining calculations of data sets and allowing users to add, view, and update data. The team also implemented several configurability functionalities and refactored data reports. The last feature completed during this iteration was functionality for capturing additional data for primary operations. Related to the environment build-out in the new consolidated data center (introduced in Section 2.2 and discussed further in Section 5.3.2), in this iteration the team installed Nexus and configured it to pull build artifacts from the Nexus Cloud server. This gave the team a mechanism for moving approved builds from the cloud into the data center's staging and other environments. The team also installed a Chef server on the Nexus server so it had a central place to manage and view the deployments to the various environments.

Release 3 iterations continued the standard three-week cycle with a Wednesday start and Tuesday end. The team used Kanban tracking of work in progress during all seven iterations. Release 3 focused on user functionality stories, including both GUI and batch operations. In addition, the team achieved a significant milestone in the build-out of the data center environment—the successful deployment of FedCLASS in the development/staging environment.

5.2.3 Release 4: Addressing Non-Core Functionality

The concept of releases was employed more loosely in this phase because the development team shifted its focus to the projected deployment (go-live and go-parallel) into the government data center run by Department Omega. Some presentations and status material used the term *Release 4* to represent the work associated with deployment into the data center.

Between Kickoff + 23 months and Kickoff + 30 months—the time period informally called Release 4—the team decided to reuse LEGACY functionality, rather than rewriting it for FedCLASS. During this period, 11 iterations occurred that each lasted 3 weeks, as shown in Figure 10. The team implemented an extensive set of user and technical stories. User stories covered both GUI and batch operations. Technical stories covered quality attributes (e.g., security), modifications to the application architecture (e.g., install Oracle TimesTen), and development infrastructure improvements (e.g., publish Cucumber scenarios using Relish).

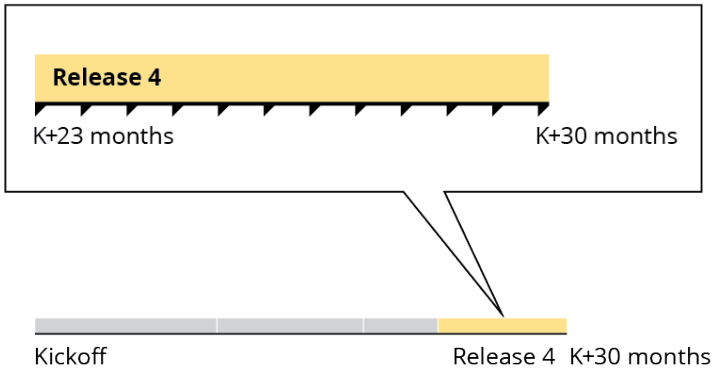


Figure 10: Release 4: Non-Core Functions

Iteration 29 occurred at Kickoff + 30 months. The iteration review was held near the end of this month and declared to be the last one. During the iteration review, the development team provided an update on parallel processing and answered questions. The review gave the team a chance to thank everyone for their support over the past few years.

5.3 Deployment

This section covers the activities to complete the move to hosting the FedCLASS production software in the government data center. The move to deployment uncovered additional work and complexity. The development team had to deal with this complexity as they began parallel work paths and focused on operational deployment in the data center.

The release plan diagram (Figure 6) had caused miscommunication between the development team and Department Omega's leadership. When they adopted Agile and Lean practices, the team moved away from using the release plan and milestones but did not formally replace them with the team's projections of completion dates. However, the leadership assumed that the dates on the release diagram were a high-level delivery schedule. Based on this assumption, leadership expected the team to deliver FedCLASS by Kickoff + 22 months.

As a result of this expectation, the development team began to interface more actively with outside organizations, such as the production data center. However, the outside organizations did not use Agile and Lean approaches. The concepts of user stories, daily standup team meetings, and full dedication to the program were all foreign to the outside organizations. The development team found the difference in work processes between Agile and Lean approaches and traditional approaches very frustrating. For over two years, team members were dedicated to completing the user stories needed to provide value to the business owner. The outside organizations had multiple priorities, and the individuals in the outside organizations had multiple tasks and priorities—no one was dedicated solely to the FedCLASS Project.

Because organizations outside the project followed a traditional approach to performing work, the development team adjusted its processes to accommodate their expectations. The team started to work along the parallel paths shown in Figure 11 to meet the go-live requirements and start of production by a target date of Kickoff + 29 months.

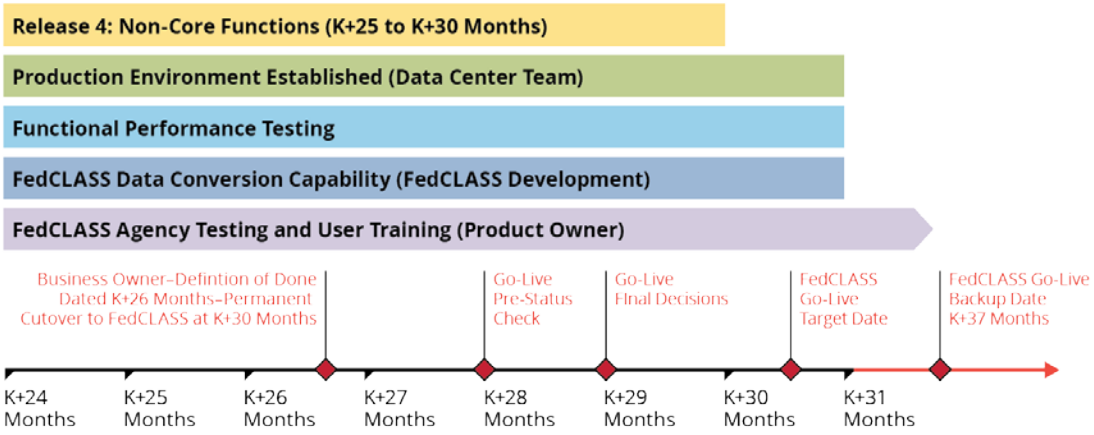


Figure 11: Moving to Go-Live Deployment

Working on parallel paths meant that some subteams (subsets of the development team) continued to work on functional user stories, while others focused on data center infrastructure. One subteam (supplemented by new team members) focused on writing programs that would convert the LEGACY database contents into the FedCLASS database structure. Converting the LEGACY database was a critical action during the go-live deployment. Another subteam focused on testing the performance aspects of the functional application. Meeting the service-level agreement for processing data files was a critical factor for the business owner.

In addition to these parallel activities, the business owner established an independent testing effort involving members of the Program Alpha business operations staff. The business owner staff performed user-oriented testing of FedCLASS, worked with the interfacing agencies, and educated other operational staff about the FedCLASS functionality.

5.3.1 Experiences with Converting the Old Databases

Although the functional user stories in the backlog targeted for the Kickoff + 29 months deployment date continued to be addressed at a steady rate, converting the legacy data to the FedCLASS format became a problem for the development team. The conversion programs ran too slowly. The conversion subteam was unable to determine a process and software programs to convert and validate the legacy data within the time allotted for the three-day go-live window at Kickoff + 29 months. This problem rapidly became the top focus of the business owner and project management. The development team took a multipronged approach to reduce the conversion time:

- use TimesTen, an in-memory Oracle database product whose purpose was to double FedCLASS performance
- have an independent small group of experienced Java developers, not involved with coding the conversion, work in parallel with the regular conversion coders to review the code and experiment with performance improvements
- get a higher performance server and move LEGACY to it for the conversion processing window

While some actions did help speed up the conversion, the team still had difficulty converting the legacy DB2 database to the FedCLASS format within the required time frame. When the SEI's

engagement with Department Omega ended at Kickoff + 35 months, this problem remained unsolved.

5.3.2 Integrating with the Data Center

During Release 2, the requirements for hosting FedCLASS were determined. The option of deploying to operations in a cloud environment was shelved due to security concerns with processing PII data outside the government-owned data center. Participation of the deployment team in the development team's daily standup session began sporadically at Kickoff + 14 months and matured during the final development push to reach a go-live milestone. The deployment team's participation gradually became a way for the development team to synchronize its work with the creation and stabilization of the data center infrastructure.

In Release 3, work included technical interchanges with the deployment team and provided insight into the dependency of the FedCLASS Project on a fully functional data center. Although the development team had been preparing to integrate FedCLASS into the larger IT ecosystem of Department Omega, concerted efforts toward this integration did not begin until Release 4 at Kickoff + 23 months. The deployment team's organization also oversaw IT systems' compliance with security, configuration management, and deployment standards for Department Omega. The development team began documenting its de facto configuration management plan, performing an inventory on its security and other IT controls, and assisting the deployment team with building the new production environment for FedCLASS.

As FedCLASS moved toward deployment, the Agile-based development team started working more closely with the non-Agile deployment team developing infrastructure at the data center. The deployment team at first continued to use the traditional processes that were the means of daily operation at the data center. These processes used service request tickets to authorize and track work. Without a service request, data center staff were not authorized to perform work.

One of the primary difficulties of working with the deployment team was the inability of deployment team personnel to adapt to the Agile practices used by the development team until later in the deployment process. Specifically, the deployment team was unable to regularly attend standup meetings, assist with the development of pertinent user stories, prioritize those user stories, or execute tasks in terms of user stories. The data center staff, including the deployment team, were still completing a project to consolidate the data centers, as mentioned in Section 2.2. Consequently, the ability of this new consolidated organization to provide timely service to projects such as FedCLASS was greatly diminished. These complications further jeopardized the chances for FedCLASS to become the system of record at Kickoff + 29 months—the target date for going operational.

The development team overcame these difficulties by adapting to the more traditional approach and treating the data center integration as a separate project rather than an integrated aspect of its own Agile and Lean development process. The development team continued with user stories for identifying all work to be done. However, for the data center's user stories, the team adjusted its practices to translate them into the deployment team's work tracking system as service requests. The team established a Kanban work flow for work stories related to the data center and supplemented the user stories with special action lists and tracking for the service requests. In addition,

they also created a more structured listing of necessary work, allowing the deployment team to better understand the requirements.

The data center's traditional approaches put an additional workload on the development team, which had to be very specific about what it asked the data center staff to do. Also, data center staff were organized around specific systems, applications, or services, and the FedCLASS user stories tended to cross these boundaries. As a result, completing a data center user story resulted in many handoffs from person to person, with each person closing out his or her action. This contrasted with the development team's processes, which did not perform handoffs and which considered a user story closed only after it passed the associated tests.

To keep the communication open and the focus on completion of the data center, the leadership of the data center's Data Management Services met weekly from Kickoff + 26 months to Kickoff + 30 months. These routine touch-point meetings helped participants ensure effective communication and progress, identify barriers, and assign needed actions for follow-up.

Late in the project, Department Omega leadership added the requirement that the government data center use new approaches to support the deployment of FedCLASS. Traditionally, the operational environment was established and maintained by manual and often labor-intensive processes. The FedCLASS Project demonstrated the value and benefit of using Chef and "cookbooks" to more automatically build the operational infrastructure. The deployment team learned to use Chef and write "recipes" to build the data center environment. By Kickoff + 30 months, the deployment team had created Chef recipes for building the environments for production and was working to implement recipes for the other environments: DevStaging, System Acceptance Test (SAT), and Simulation. The process for building an operational environment was shortened from a few months to a few hours. Also, with Chef recipes, the work steps were both repeatable (automated to a larger degree) and more auditable.

5.4 Estimating System Completion

At Kickoff + 24 months, concerns about the development team's ability to deliver the new software by the target date of Kickoff + 29 months became more acute. As stated earlier, some parts of Department Omega viewed the release plan milestone of Kickoff + 22 months as the completion date for FedCLASS, although the development team moved away from using the release plan and milestones following the end of Release 1.

After Release 1, the development team managed its estimation process informally. While an estimating and progress spreadsheet was made available to the entire team, members did not make active use of these estimates or the estimation process. They did include an estimated completion date as part of status reporting to the standing Management Steering Committee. This practice would cause problems later when the Steering Committee expected the development team to adhere closely to its estimates.

Department Omega leadership kept the development team focused on a go-live target of Kickoff + 29 months. With this focus, both the development team and the deployment team worked the issues related to establishing the government data center. The two groups collaborated to enhance the daily standup as a mechanism to support the work that each group accomplished. They expanded the Kanban board to address user stories for the government data center, in addition to the

normal work-management processes in the data center. The cross-flow of communication at the daily standup helped resolve impediments to operational deployment of the FedCLASS solution.

5.5 Changing Definitions of Success

At project launch, the program definition of done was described as "fully tested code, deployed and working in a production-like data center environment, with converted data and integrated with existing interfacing systems." To fully refine this definition, the business owner had to determine whether the team's end goal for FedCLASS would be go-live or the system of record. Go-live means that a system is running, performing real processing on real data. Two systems handling the same data can be live at the same time, running in parallel. But only one system can be the system of record—the authoritative data source for a given data element or piece of information—for a type of data in a specific agency at a time.

At Kickoff + 24 months, the development team started creating a series of scenarios for going live on the target date of Kickoff + 29 months, which was the expectation of Omega leadership. The team also looked at alternative scenarios by which FedCLASS could be described as a successful conversion from LEGACY. These scenarios ranged from a low threshold of having all coding finished, with the system presumably ready to deploy, to a high threshold of declaring FedCLASS to be the system of record at Kickoff + 29 months. After the team presented these scenarios to the business owner, Department Omega's leadership firmly and unequivocally decided at Kickoff + 25 months that nothing short of declaring FedCLASS to be the system of record at Kickoff + 29 months could be categorized as success.

This was the development team's first experience with a firm deadline for operational delivery. Team members began a series of discussions and actions to better understand specifically what they needed to complete to meet the business owner's needs. They created a specific Go-Live Checklist presentation to reach the Kickoff + 29 months target. The business owner and Department Omega's leadership reviewed and accepted these go-live criteria, and the development team established incremental checkpoints for Kickoff + 27 months and Kickoff + 28 months to evaluate the progress on completing work to confidently go live with FedCLASS and its infrastructure.

To gain confidence in the development team's ability to estimate the completion of work, Department Omega's leadership commissioned external studies of the estimated completion date. The team estimated that they could complete work on functional software requirements in time for the Kickoff + 29 months target. However, when they included the work necessary to also have FedCLASS operational in the government data center, that target was unrealistic. The external studies of the estimated completion date also concluded that the target date was high risk.

The development team and the deployment team continued to work together collaboratively to achieve the go-live target date of Kickoff + 29 months. As they accomplished the required work, they realized that they could not achieve high confidence for a successful go-live transition to the new FedCLASS software. During the checkpoint review at Kickoff + 28 months, Department Omega's leadership released the two groups from the go-live target date. However, attention and focus remained on finishing the build-out of the government data center for deploying FedCLASS by that date.

5.6 Preparing for Deployment

The move to FedCLASS as the system of record did not occur at Kickoff + 29 months. The next go-live window would not occur until after a seasonal period of heavy use of LEGACY ended, so the old system continued to be the system of record during this time frame. Beginning at Kickoff + 30 months, the development team focused on getting FedCLASS to process matching files in parallel with LEGACY. Work continued on all parallel paths (see Figure 11), but the priority was to establish a stable operations environment for moving FedCLASS to production. To meet that goal, the development team focused on establishing the Chef recipes that would rebuild the operational environment if a failure occurred. They also worked with the deployment team to ensure that the backup and recovery policies would function correctly if needed.

During this time period, the development team continued to struggle with converting the legacy DB2 database to the new FedCLASS database structure. Conversion was a "one-time" event, necessary to complete the shift to FedCLASS; however, it could not be completed in the 20 hours allotted for the go-live window. The business owner's requirement was to halt production using LEGACY and restart production using FedCLASS. The cutover had to occur over a three-day weekend and be completed within the three-day weekend. The team continued to work on the software applications to rapidly convert the DB2 database. They also started an additional path to acquire a faster mainframe and larger amount of storage in an attempt to meet the conversion window.

While the development team worked on issues related to the database conversion, Program Alpha's business operations staff both supported LEGACY operation for daily use and performed user-oriented testing of FedCLASS. The operations staff also started performing end-to-end tests of the interfacing systems. The need to support the end-to-end testing put additional workload on the development team.

5.7 Automated Delivery Pipeline and Continuous Integration

FedCLASS development moved toward addressing changes in the application code and in the computing infrastructure as part of an integrated system. The development team defined a process and roles to get new and modified code into production using a continuous-integration approach. This new approach required changes to the infrastructure to be addressed concurrently with the code that the infrastructure supported. This meant recognizing the interdependence of code and infrastructure, taking a unified approach to making changes, and performing integrated testing of the code and infrastructure before deployment to production.

The release-management process defined by the development team began when either the deployment team or Department Omega identified a change. Types of changes included applying patches, installing additional software, changing a parameter, and changing the code. Development work always began in the Amazon Web Services (AWS) cloud environment. Changes made by developers were checked into the formal version control system (Git) in AWS multiple times per day. Upon every check-in or commit, the continuous-integration server created a build and assigned a build number in AWS.

The continuous-integration server in AWS ran the unit and acceptance tests and failed the build if all tests did not pass. The continuous-integration server also ran quality and security scans using

Fortify and SonarJ and failed the build if the code violated security and quality thresholds established by the deployment team. Once a build passed all automated tests, the development team updated the build number in AWS Chef, and the change was propagated to the AWS deployment environment. The team tested the deployment process to ensure that Chef recipes worked as expected.

In addition to being deployed to AWS, successful builds were also deployed to the data center's infrastructure. This allowed the deployment team to view the differences between the currently deployed build and the new release. To deploy a new release, the team's system administrator simply updated the build number on the team Chef server for the FedCLASS Dev-Staging environment, and the build underwent testing there. In addition to Dev-Staging, builds were also deployed to and tested in the SAT environment and the Simulation environment.

The data center's Service Operation Branch deployed and verified all releases to production. Releases to production were deployed to both the production environment and the disaster-recovery environment. In the development team's release-management process, FedCLASS production deployments were planned for twice per week, with the intention of gradually moving toward daily deployments.

5.8 Sustainment

The initial FedCLASS deployment replicated the LEGACY functionality in the FedCLASS architecture and infrastructure. Post-deployment, the development team's focus shifted to developing new functionality and features. FedCLASS was designed to be more flexible and to enable additions and changes. As Program Alpha's needs changed over time, this flexibility for growth would support new functionality to increase sources of data, data volume, and data-matching effectiveness.

Agreements between Department Omega and the development and testing contractors for supporting FedCLASS were incomplete at the time that we completed our study. But they expected the same roles followed during the development phase to continue in the post-deployment phase. The business owner and the business owner's staff continued to plan the post-deployment work.

While this case study was being written, the development team was planning to move to a DevOps strategy of small, frequent software releases. In this approach, the development team and operations team work closely together to ensure the controlled and systematic release of new business functionality on very short release cycles. The development team planned to use DevOps in the future, but the operational data center had not yet approved this approach.

6 Project Analysis

The history of the FedCLASS Project described in the preceding sections yields some helpful insights on an early foray into Agile development in the federal government context. These insights may benefit future Agile projects in the federal government and the software engineering community as a whole. This section describes in detail the insights gleaned during the research of this case study, particularly regarding Agile and Lean adoption, technical approaches, and leadership.

6.1 Analysis of Agile and Lean Adoption

During Release 1, the FedCLASS team implemented a RUP-based iterative approach supplemented by selected Agile practices from Scrum and XP. While this approach proved effective to a degree, the team believed that the development culture and environment did not yet effectively embrace the underlying values and principles of Agile, particularly with respect to the concept of a self-organizing and self-managing team. This was the primary driving force behind the change of coaches and development approach that occurred between Release 1 and Release 2.

Beginning with Release 2, the team focused on adopting Agile as a philosophy and culture. They also embraced Lean and Kanban, while continuing to implement selected Agile practices from Scrum and XP. Certain Agile practices in use during Release 1 were discontinued. Of particular significance was the discontinuation of story point estimation and the concept of team commitment to the completion of a set of stories for a given sprint.

The FedCLASS development team was successful both in embracing the Agile culture and principles and in implementing many key Agile practices, as described in the subsections below on

- requirements and test
- enhancing collaboration
- the contracting environment

However, as with any change initiative, some practices were candidates for improvement, as described in the subsections below on

- estimation practices
- metrics and continuous improvement

6.1.1 Requirements and Test

The areas of requirements and test illustrate how the FedCLASS Project's adoption of Agile practices was informed by the Agile mind-set. Agile calls for incremental delivery of customer-valued functionality. The development team consistently delivered incremental releases of the system on a three-week cadence. The prioritization and specification of the system was based on a close collaboration between the development team and the product owners. A three-person product owner team was dedicated to the project and was available to the development team at all times via an "always-on" video teleconference (VTC) connection. The product owners all had deep subject-matter expertise and long-established relationships with the user community and other key stakeholders.

In keeping with Agile practices, the requirements were specified as user stories and pulled from the backlog in accordance with product owner priorities. Agile practices regard user stories as "placeholders for conversations" between developers and product owners. The development team deeply embraced this concept. Every user story resulted in in-depth discussions between a product owner and a developer. In most cases, a tester was also included in the conversations. This multi-functional approach ensured comprehensive, in-depth analysis of stakeholder requirements. A further practice innovation that the FedCLASS Project adopted was to produce executable requirement specifications (tests) of the user stories. These scenarios were written by the product owners and served as both requirements specifications and automated test scripts for execution by the developers. Using executable scenarios to specify stories is a form of ATDD and advances the Agile goal of "building quality in."

Involving the developers in story analysis and scenario creation ensured that they wrote the code for FedCLASS with an in-depth understanding of the system's expected behavior and output. This understanding is reflected in the success of the stakeholder demonstrations conducted at the end of each iteration and in the limited number of defects that were fed back to the team for rework.

6.1.2 Enhancing Collaboration

Overall, the team displayed a strong sense of collaboration, trust, and mutual support as well as a commitment to overcoming obstacles and delivering customer value. This was achieved even though the team was broadly distributed across multiple locations. The tool environment used to support the distributed team included an always-on VTC and instant-messaging chat rooms. VTC and instant messaging provided different communication options depending on the subject matter and the communication preferences of individual team members. Team members also had the option to intersperse work in an office location with work at home, which helped offset the difficulties of distributing work across multiple time zones. The tool environment allowed flexibility that supported work–life balance (related to the Agile goal of sustainability) while maintaining a consistent and rigorous development focus.

More specifically, the collaborative approach to story specification was a major achievement for the FedCLASS Project from both a practice perspective and a cultural perspective; collaboration is one of the key cornerstones of any Agile method [Cockburn 2007, Highsmith 2004]. The organizational boundaries that frequently appear in traditional projects between developers and product owners were almost nonexistent in the project at the team level. Several team members cited collaboration as an aspect of Agile development that they greatly enjoyed and valued. Most stated that they would be reluctant to work in any other way on future projects.

Another form of collaboration, the use of pair programming, also contributed to the goal of building quality in. Team members frequently cited pair programming as a practice that was challenging to adopt. However, after the initial period of learning and adjustment, it was well accepted by the team. In addition to improved code quality, team members appreciated the fact that by facilitating collective code ownership, pair programming allowed more flexibility because it reduced the likelihood of one person becoming a single point of failure for a given piece of code. Programming pairs were rotated, which facilitated knowledge transfer within the team and strengthened team interactions.

While collaboration and communication between the development team and product owners were excellent, some communications gaps resulted in negative impacts to the project. In particular, there was a misunderstanding between the team and executive leadership about release milestones and a lack of clarity between the team and operations about the viability of using AWS as a production environment. Both misunderstandings arose at a later stage in the project and resulted in rework, delays, and considerable stress. As a result, the team learned that a focus on internal cohesion and business stakeholders should not obscure the need for communication with other key organizational stakeholders.

6.1.3 The Contracting Environment

In addition to changing its development culture and practices, the FedCLASS Project focused on transforming its contracting environment. This transformation manifested itself both in contracting practices for obtaining external services and in Department Omega's relationship with the development and testing contractors.

Traditional approaches to contracting practices often present a barrier to Agile adoption. Department Omega follows standard FAR, and the contracts for the FedCLASS Project were firm fixed-price contracts for services. Agile development assumes the capacity to respond rapidly to changes that may include the need to obtain initially unanticipated expertise, software, hardware, training, manpower, and other products and services. For the project, the government contracting officer supported the creation of flexible agreements designed to enable responsiveness to changing development needs. This empowered management to source technical expertise quickly to resolve infrastructure issues, obtain new tools, and acquire coaching and consulting resources. This capability, coupled with tight feedback loops between management and the development team, enabled management to remove impediments and assist the team in timely course correction.

The Agile Manifesto calls for valuing "customer collaboration over contract negotiation" [Manifesto 2001]. On previous Program Alpha projects, as Department Omega's agents, the development and testing contractors' role was to provide solutions for software development. The FedCLASS Project transformed the nature of the relationship between Department Omega and these contractors. The department took a leadership role in the Agile adoption effort, bringing in outside expertise and hiring external coaches. Department Omega leadership, project managers, and product owners engaged directly with development resources on a daily basis rather than at prescribed times within the software lifecycle.

6.1.4 Improving Estimation

After Release 1, the development team discontinued the use of team-based estimation practices, which previously had been done in story points using planning poker. Instead, estimation was done in story points by the coaches and was used primarily for status reporting to upper management. Changing estimation practices disrupted communications about delivery projections between the team and upper management, which reduced upper management's confidence in the team's ability to forecast.

As the development focus moved to operational deployment, senior leadership of Department Omega expected projected milestone dates. Because team members were not comfortable with their estimates, senior leadership tended to set dates, and those dates put the team under pressure.

Independent estimation reviews made by external experts helped to reset expectations for work completion.

6.1.5 Metrics and Continuous Improvement

The project did not emphasize disciplined metrics collection and the display of "information radiators," a recommended Agile practice. Although the team generated automated metrics for code and design quality on a nightly basis, we did not see evidence of metrics collection and analysis in other areas. A rigorous data orientation can serve as a platform for continuous improvement, which is central to Agile and Lean practices. While the development team conducted iteration retrospectives, the concept of data-driven continuous improvement was not well integrated. Data recording and monitoring were not part of the team's day-to-day operations, and the team tended to bypass retrospectives when confronted by delivery deadlines.

6.2 Analysis of Technical Approaches

The FedCLASS Project exposed the development team to a variety of new technologies and technical methods. The team adopted many of them successfully, as described in the subsections below on

- cloud development
- automation of test, integration, and build processes
- layered architecture
- the use of open source frameworks
- data conversion

Although the development team successfully adopted and integrated a number of technologies, gaps in project communication led to complications in technical implementation, as described in the subsection on cloud development. In addition, not all of their development was without technical issues, as described in the subsection on data conversion.

6.2.1 Cloud Development

Early in the project, the development team chose to use publicly available cloud services during development, hoping that Department Omega would also permit a production deployment to the cloud. The development team chose a cloud approach because it sought the abilities to

- spin up and spin down virtual servers at will while paying only for services used
- tune virtual servers as desired and install software on those servers at will

This use of cloud services allowed the development team to begin development almost immediately, without the lag time associated with acquiring and building a development environment. It also aligned Department Omega with the Federal CIO initiative to start moving the government toward the use of cloud technology.

The circumstances, however, began to change when preparing to move FedCLASS to a production environment. Department Omega neither had a cloud environment ready for production systems nor would it allow production systems to be deployed to public clouds. Consequently, production was deployed on a traditional premise-based IT infrastructure. The development team,

having become accustomed to the flexibility associated with cloud services, had difficulty reacclimating themselves to this more traditional approach to provisioning and deploying IT infrastructure. The work necessary to complete the production data center took more time and was one of the justifications for delaying deployment to production status for FedCLASS.

6.2.2 Automation

The development team chose to automate acceptance testing and the release delivery pipeline (continuous integration). Using these automated practices allowed for short and frequent iterations. For example, without the ability to perform automated tests, a non-automated, full-regression testing cycle could easily increase the length of an iteration. The automated tests were run on multiple code bases each time a new check-in occurred.

The team succeeded in using tools to automate tasks. Tool selection was based mostly on recommendations of the coaches. The selected tools included

- Subversion (later Git) for configuration management
- Jenkins, Hudson, and Bamboo servers for continuous integration
- Cucumber for automated acceptance testing
- Chef for automated deployments and computing hardware configuration management
- CAST, Fortify, and SonarJ to review code for adherence to project standards

6.2.3 Layered Architecture

One of the most established and widespread best practices in software architecture is to divide software into layers with clean and clear boundaries. This is achieved by assigning each code unit (e.g., a class) to one and only one layer that handles one category of job, such as business decisions. Each layer remains agnostic about how other layers handle their jobs even as it interacts with those layers to accomplish the overall goal of the use case it supports. FedCLASS employed such a layered architecture.

The benefits of a layered architecture are as follows:

- *Maintenance* is improved because predictable patterns for fulfilling user stories are always honored, which allows programmers to quickly find and update code units when modifying a specific story.
- *Testing* is improved because specific types of functions, such as saving data to a database, are isolated to specific code units and consequently can be tested in isolation.
- *Interoperability* and *portability* are improved because the functions associated with interactions with another system, suite of libraries, or platform are isolated to one layer of code, which can be replaced or rewritten to adapt to a new system, suite of libraries, or platform.

Because of this layering, FedCLASS exhibited a strong degree of portability, particularly between JEE engines (e.g., WebLogic, WebSphere) and between relational databases. Additionally, the system exhibited a strong degree of maintainability because user stories tended to follow similar paths through the architecture.

6.2.4 Open Source Frameworks

The development team used proven open source frameworks such as JEE, EclipseLink, Struts, Spring, Cucumber, Git, Jira, and Chef. Open source software has the following benefits:

- Many developers have used, tested, and offered improvements to open source software.
- Open source software implements reusability, not only within a project but outside of it.
- Open source software brings the collective knowledge of the software engineering community at large to a project.
- Use of open source software often reduces costs because many of these tools are free.

For example, using JEE relieved the development team of coding the detailed mechanisms of transmitting data from one server to another over TCP/IP or HTTP. Those mechanisms were available to them as Java code that was already familiar to many of the team members and was so well tested that its correctness could be trusted and did not require specific testing.

6.2.5 Data Conversion

One of the most challenging areas for the development team was converting its legacy data into the format required for the new system. The challenge was not in relating the legacy data model to the new data model but rather in extracting, transforming, and loading the data in the span of downtime allotted for the conversion. Conversion, for the purposes of fully migrating from one system to another, is different from standard development:

- Conversion code is meant to be used only once, requiring less focus on maintainability or other best practices. The overriding considerations are correctness and performance.
- Coders for conversions do not usually need to understand the functioning of the primary system well. They are essentially converting data from one format to another.

In hindsight, to better address the performance issues discovered in the conversion programs for converting the legacy DB2 data, the team could have begun work on the conversion earlier—theoretically as soon as the new data model was nearly finalized. Additionally, the team could have dedicated a small team to conversion that was integrated with deployment efforts rather than development efforts.

6.3 Analysis of Leadership

Department Omega leadership on the FedCLASS Project exhibited best practices in project management [NIST 2013] by

- setting the team's vision and direction, such as choosing Agile methods for the development approach
- communicating with the workforce, such as engaging in daily standups with the entire team
- creating an environment and culture for high performance, such as allowing for continuous improvement, discarding practices that did not work, and adopting those that did

Section 6.3.1 describes how Department Omega leadership exhibited these best practices through managing cultural changes. Sections 6.3.2 and 6.3.3 describe improvements that the leadership team could have made in the areas of Agile interface and risk management.

6.3.1 Cultural Change

The FedCLASS Project leadership faced the challenge of managing the cultural and process adjustments necessary for the players to move from traditional, hierarchical management techniques to being a self-directed team. As noted earlier, Agile processes were well adopted by the project leadership. The development and testing contractors eventually adjusted well to the Agile practices through their participation in the FedCLASS development and were mostly receptive to Agile and eager to experiment with new techniques and approaches.

To promote the benefits of Agile practices, the project leadership adopted a successful team-empowering style by supplying the development team with the tools and resources necessary to achieve progress and removing obstacles that would hinder the team's progress. Ultimately, organizational leaders are responsible for guiding an organization to produce the results associated with its goals, and project leadership fulfilled this responsibility by implementing innovations, initiating improvements, and guiding the program toward its strategic objective of delivering the FedCLASS system.

6.3.2 The Agile Interface

During the development of FedCLASS, some discord arose around the "Agile interface." That is, the Agile methods sometimes conflicted with the traditional governance processes surrounding them. Some of the traditional governance processes that the project interfaced with were

- Department Omega IT policies, such as security and configuration management for the operational data center environment
- Department Omega Technical Architecture Review Board
- the development contractor for supporting the LEGACY implementation

Conflicts around the Agile interface mostly manifested as communication breakdowns, particularly a lack of proactive communication between the project leadership and the Department Omega configuration management and operational data center personnel as well as back-channel communications with the development contractor.

The project leadership successfully addressed many of these communication breakdowns by insisting on defined and regular meetings with project stakeholders and by engaging coaches to mentor stakeholders outside of the core development team. However, a major communication gap persisted between the team and the Management Steering Committee regarding the release plan, team estimates, and the projected go-live date.

6.3.3 Risk Management

The most common risk associated with software development—inadequate delivery of product—is mitigated by Agile practices through short iterations and frequent deliveries [Cohn 2010]. However, risks can be associated with *all* aspects of a software development project—including hardware, requirements, and more—not just the software per se.

The development team dealt with risks through informal discussion and strategy sessions and did not develop or institute a formalized Risk Management Plan. As a result, the team tended to analyze the program and enterprise-level risks that manifested during the development of FedCLASS

in an ad hoc manner as they arose, rather than anticipating and planning for them using a systematic and documented process. As a result, leadership needed to develop mitigations quickly for highly anticipatable risks, such as the manifested risk that FedCLASS could not be deployed in a cloud and would require a dedicated hardware infrastructure internal to Department Omega. Developing this hardware infrastructure in cooperation with Department Omega production personnel proved to be one of the most difficult tasks for the development team and would have benefited from a longer, more deliberative planning process. While informal means may be sufficient to mitigate risk at the level of the software development team in an Agile context, a greater degree of formality is usually prudent when scaling to the enterprise.

7 Summary

Department Omega chose to follow an uncharted path to creating the FedCLASS capability. After years of experience with traditional and incremental changes in LEGACY, the department's leadership was willing to take a risk and pilot innovative methods and technology. The primary goal that drove the reengineering of LEGACY toward a completely new solution was to make a fundamental break with the 15-year strategy of evolutionary, incremental change to the existing platform. Program Alpha needed a software platform for future growth. LEGACY needed the capability and capacity to grow and support expanded users of the system, as outlined in Department Omega's strategic planning goals. The broader 2010 federal IT reform environment supported taking a risk on innovative development and change. As part of its "take a chance" strategy, Department Omega's leadership charted a greenfield approach to innovation and change.

Figure 1 shows the broad areas of innovation and change that are briefly summarized below:

- *a new management role:* The business owner had extensive experience with LEGACY and was willing to take a role with direct responsibility for achieving the primary business goal. As a result, a unique relationship was established between Department Omega and managers from the development and testing contractors. Department Omega leadership was responsible for the development team's day-to-day work, and the business owner was directly involved with the user stories. Also, external coaches were introduced into the project with the goal of rapidly providing new knowledge for team members. The new technologies and methodologies were not part of the existing skills of Department Omega staff.
- *new technology:* Cloud-based development, a new programming language, new commercial products, and new development environments and tools were introduced as part of building a foundation for future growth.
- *a new development team concept:* The department embraced new Agile and Lean development team concepts, such as having dedicated team members who were self-organized and operating in a virtual team room.
- *a new software methodology:* The development team experimented with software methodologies including RUP, Agile, Lean, Scrum, and Kanban. They also experimented with integrating practices from different methodologies such as daily standups, Kanban boards, story point estimation, user stories, pair programming, and continuous integration. Throughout the experimentation, the team remained focused on Agile and Lean flexibility in support of business owner needs and value delivered.

The pilot program did not remain simply a pilot of Agile and Lean methods. What started as an innovative pilot of new technology and approaches became a broad new transformational development effort that effected changes across the organization. Some critical and key enabling factors for the project included

- a business owner who had professional IT skills and operational experience in the business area
- a program manager who had experience with new technology
- an environment of government-wide IT reform and a push toward new technology

- a senior leader who was willing to try something different

Many factors and influences came together at the start of the FedCLASS Project that helped it succeed. They supported and reinforced each other, making it possible for the project to become a model for change and opportunity for learning within the organization.

Appendix A Project Stakeholders

Program Alpha focused on providing services to the federal and state agencies. With a very broad customer base, Program Alpha had a large number of data partners who depended on its services. At the start of the FedCLASS Project, 18 key project stakeholders were identified. These key stakeholders and their application interfaces were the same as those for LEGACY.

A goal of the project was to retain the existing application interfaces and limit the impact on the external data partners. Interfaces were kept unchanged to limit the complexity of managing interfaces with external agencies. The existing Program Alpha stakeholder communities participated in the transition from the legacy infrastructure to the FedCLASS system. The identified stakeholder listing is categorized below into broad areas of concern and influence.

Project Oversight

The Sponsor (Department Omega, Assistant Commissioner), who would ultimately accept or reject FedCLASS as the system of record.

The Senior Project Manager (Department Omega Projects Branch), who supported the project manager for FedCLASS development, provided the interface with Governance Oversight, and oversaw contractor support through the **Contracting Officer Technical Representatives**.

The FedCLASS Project Manager, who had responsibility for development work.

Product Owner

Program Alpha Director (Primary Business Owner) ran Program Alpha to perform a vital function of the U.S. federal government for federal and state agencies. The director marketed the Program Alpha system to expand its use among these agencies and train users on Program Alpha software. The director set the business needs that determined what the FedCLASS system should do and accepted or rejected the incremental deliveries of demonstrated capabilities. The director then made a recommendation to the Sponsor to accept FedCLASS as the system of record for LEGACY. The director also established **Program Alpha Business Owner Delegates** to act on behalf of and with the authority of the primary business owner as integral members of the development team.

Governance

Governance for the FedCLASS development effort was relatively complex because of the extensive list of stakeholders. The governance covered Department Omega, the Sponsor's direct oversight of FedCLASS development, and governance on how FedCLASS fit within the larger federal IT enterprise and information security controls. These controls included the IT Governance Board, Information Systems Security Officer Certification and Accreditation process, Enterprise Architect, and CIO Configuration Management, Operations, and Policies.

The legal and regulatory governance covered how Program Alpha software and program operations met the legal and regulatory requirements for handling PII. These legal requirements flowed

into the design and testing of FedCLASS. Because the development and testing contractors were involved in development and support of both LEGACY and FedCLASS, the contractors' auditors helped enforce compliance with controls.

Development Team

The Department Omega Projects Branch had a project manager role for FedCLASS development and provided the interface between the Governance Oversight and the FedCLASS Project.

The Program Alpha Business Owner Delegate had the Product Owner role on the development team.

A development contractor handled software development, systems architecture, the database, and maintenance of LEGACY.

A testing contractor handled client stress/load testing and LoadRunner.

A coaching contractor provided coaching and new technology subject-matter experts on the software development team.

The FedCLASS Project maintenance team will consist of members from the same organizations that participated in development of FedCLASS.

Interfacing Systems

Interfacing systems are external users and systems that pass information to or receive information from the Program Alpha system. The FedCLASS Project needed to address requirements of 13 interfacing systems during development and integration of the new system.

Operational Data Center

Department Omega's Information and Security Services provided the infrastructure for hosting the FedCLASS applications. It also hosted and supported the LEGACY system being replaced by FedCLASS.

Platform Engineering: engineering for Unix, Middleware (web, application, database), Mainframe z/OS, Mainframe z/Linux, Storage (Mainframe and AIX), Single Sign-On (SiteMinder), Disaster Recovery (Unix and Mainframe), Intel (Windows), and Citrix

Network Engineering/Operations

Platform Operations

Configuration Management: operations for Unix, Mainframe, Service Desk, and Desktop/LAN support

Applications Server: WebSphere application server and WebFocus reports for Program Alpha web client

Database Administration: current Program Alpha DB2 database administrator

Disaster Recovery: for Distributed Systems IBM P-Series using Global Mirroring

IBM Tivoli Identity Manager (**ITIM)\DACD:** provision of user accounts and passwords

Middleware: operational and engineering support for Middleware software (Oracle, DB2, Web-Sphere, MQ, SiteMinder, WebFocus, Wily)

Monitoring: performance monitoring, system status, and alerts using CA Wily and CA Unicenter

Scheduler: automation of batch job scheduling on the mainframe and distributed platforms

Single Sign-On (SSO)/SiteMinder: how users currently sign on to the web client through SiteMinder. The team assists developers with implementing SSO within the Program Alpha client using SiteMinder.

Storage: allocation of data

Virtualization: engineering virtual hardware and operating systems for IBM P-Series and AIX platforms

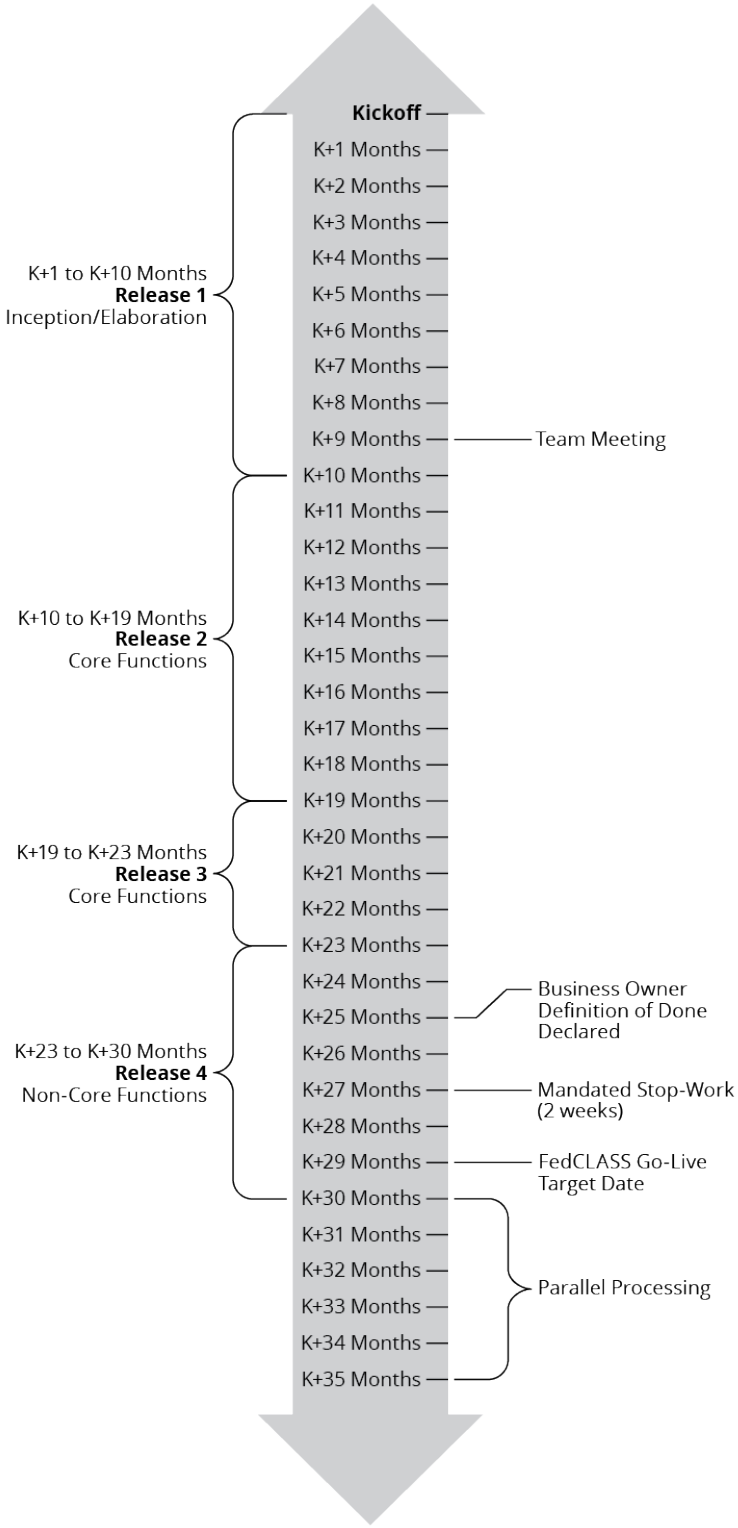
Operational Support

Operational support included all those who supported the daily operation of Program Alpha.

Program Alpha Operations: system support and use

Program Alpha User Acceptance Testing Team: system supports, user acceptance, and end-to-end testing to verify new users

Appendix B Project Timeline



Appendix C Development Tools

Throughout this case study, we cited many tools used by the development team. This appendix lists those tools with their typical uses.

Table 2: Development Tools Used in the FedCLASS Project

Tool	Use
Amazon Web Services (AWS)	Commercial, proprietary IT services. Cloud-based infrastructure as a service (IaaS) and software as a service (SaaS). Provides application programming interfaces (APIs) for mature, widely used web services; open standards that can be used to provide or integrate AWS, third-party, or custom-developed storage; and computing, networking, and other infrastructure services into customer applications. Used for development and testing. http://aws.amazon.com
Bamboo	Automates tasks for continuous integration
CAST (static analysis tool)	Structural quality measurement Maintenance cost model Estimation of technical debt Run weekly to check total code set
Chef	A systems integration framework, with both open source and proprietary versions, built to bring the benefits of configuration management to the entire infrastructure. Key enabler for continuous integration, in which modified source code is recompiled, retested, and verified upon check-in. Automates configuration, integration, and deployment of software. Insulates developers from the specific details of the target hardware or operating platform, allowing them to focus more attention on the application code. Provides “automated infrastructure” by allowing developers to write software modules that describe the target operating hardware and software platform and how it should be deployed, configured, and managed. http://www.opscode.com/chef/#how-works
Connect:Direct	Transfers files between mainframe computers and midrange computers
Cucumber	Non-commercial open source product for running automated acceptance tests. Allows software development teams to capture requirements in simple plain-text, human-readable scripts that can be compiled and tested. The scripts describe the software’s intended behavior and become a primary source for documentation, automated tests, and specs for developers. http://www.cukes.info/
DB2	A relational database management system from IBM for storing, analyzing, and retrieving data efficiently
Eclipse	A free Java development environment http://www.eclipse.org/
EclipseLink	An extensible framework that allows Java developers to interact with various data services, including databases, web services, and enterprise information systems http://www.eclipse.org/eclipselink
Fortify Tool Suite	Security testing of software under development Run weekly to check for security issues Includes the HP Fortify Static Code Analyzer, a static analysis tool for source code. Scans source code for patterns and indicators of security vulnerabilities and malicious software. Helps reduce security risks by identifying vulnerabilities early in the development lifecycle. Used to improve the security of developed software. http://www8.hp.com/us/en/software-solutions/software.html?compURI=1338812

Tool	Use
Git	Source code management, configuration management
Jenkins and Hudson	Jenkins and Hudson are both continuous-integration servers. They watch for changes in the source control. When they see a change, they check out the changes, build the software, and run all the tests. If there are failures, they notify the team by sending emails, updating a webpage, and turning on a red lava lamp. The goal is to minimize the time between a check-in that accidentally breaks something and the time it is discovered and fixed.
Jira	Commercial product for project management and defect and issue tracking http://www.atlassian.com/software/jira
Maven	Non-commercial, open source software project management tool that can manage a project's build, reporting, and documentation from a central source of information. Allows developers to set up automated, repeatable scripts to "build" (compile) source code into executables. Helps automate reporting and documentation of source code configurations. http://maven.apache.org/maven-features.html
Nexus	The development team's artifact repository. When team members needed to have the Oracle jar available for their build, they put it in Nexus. When they needed to add the Spring jar to the build, it went in Nexus. By centrally managing all the artifacts, developers need not spend time downloading them manually (Maven does this for them), and the source code repository stays small because it is just the source code—not all the .jar files of the libraries and frameworks being used. Nexus is also used to transfer code and other files into the operations center. Its Nexus server is set up to mirror the development team's AWS server. The development team recently purchased the professional version of Nexus and is getting it installed and configured. It will allow them to analyze the open source libraries used for security and licensing issues.
Nexus Cloud servers: • primary server • satellite server • cloud	A mechanism to move approved builds from the cloud into the staging and other environments. The team also installed a Chef server on the Nexus server, providing a central place to manage and view the deployments to the various environments.
Oracle TimesTen In-Memory Database	A relational database that runs in the application tier, storing all data in the main memory and thus dramatically reducing latency and increasing throughput https://www.oracle.com
Relish	A tool that displays testing scripts in formatted output for understanding
SonarJ	A static analysis tool for testing open source software code. Checks source code quality. Run daily on every check-in of code.
Struts	A free, open-source framework, developed by Apache, for creating Java web applications https://struts.apache.org
Spring	A tool that provides support to increase developer productivity in Java when using Apache Cassandra https://spring.io
Subversion	Non-commercial open source product for software version control. Used to track, control, and manage software changes. http://subversion.apache.org/

Appendix D Training for the Agile Development Team

Training for the development team was provided incrementally during the early phases of the FedCLASS Project, as part of starting up the dedicated team. The team was trained by the coaches in the general topics as shown in Table 3. Most training was done as part of the normal flow of work by the team. Limited formal external training was for Scrum master training.

Table 3: Identified Training Events for the FedCLASS Project

Training Event	Focus of Training
Agile Project Phases	Training on Agile project phases, iterations, and understanding the development problem
Use-Case Modeling	During use-case modeling training, the development team modeled use cases (diagrams and outlines).
Software Architecture Principles	Completed an architectural concerns survey for the FedCLASS system. Identified the requirements that are architecturally significant and defined candidate architecture for the FedCLASS system.
Estimating Techniques	Estimating the FedCLASS Project size. Included • planning poker • domain analysis • techniques for estimating use case points • analogy
Introduction to Pattern-Enabled Development	Introduced the concepts of using proven patterns to guide the development. http://patternenabled.com/
Cross-Trained Each Other	Adopted three different technologies and processes, including • AWS (cloud) • Maven build and deployment technology • new multi-threaded capabilities of Program Alpha application
Acceptance Test–Driven Development	Training using a test-focused development approach
Kanban and Lean Systems Thinking	Training on use of the Jira Tool and Kanban methods applied to software development projects
Cucumber Tool	Training on use of the tool, skills, and knowledge needed to write the features to be developed and tested
Git Tool Training	Git-focused training was done as Lunch-and-Learn Sessions, to help the team move to the new tool. Also, topic-specific training was provided when needed. http://docs.opscode.com/
Chef Training	Formal Chef training was provided for the integration. Chef is a tool and approach to automating the configuration of the infrastructure.

Appendix E Acronyms and Glossary

Term or Acronym	Definition
Agile	"A set of methods and practices based on the values and principles expressed in the Agile Manifesto. Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context." https://www.agilealliance.org/agile101
API	Application programming interface
ATDD	Acceptance Test–Driven Development http://www.acceptancetestdrivendevelopment.org/
AWS	Amazon Web Services http://aws.amazon.com/
business owner	The person with the role of defining what is important to the business
CIO	Chief information officer
coaching contractor	A teaming relationship between the two named companies to support the FedCLASS Project
COBOL	COmmon Business-Oriented Language
construction	In RUP, the construction phase involves designing, writing, testing, and completing the product. https://techterms.com/definition/rup
definition of done	A list of criteria that must be met before a product increment or user story is considered complete. In Agile practice, the development team defines this criteria. https://www.agilealliance.org/glossary/definition-of-done
DevOps	Continuous small software releases, with development and operations teams working together to ensure controlled, systematic releases of new business functionality on very short release cycles http://theagileadmin.com/what-is-devops/
Extreme Programming	A software development methodology that is intended to improve software quality and responsiveness to changing customer requirements http://www.extremeprogramming.org/rules.html
fail fast	Failing immediately and visibly; used in the context of trying something new and learning what works by doing http://martinfowler.com/ieeeSoftware/failFast.pdf
FAR	Federal Acquisition Regulation
FedCLASS	Program Alpha software, a major reengineering of LEGACY
go-live	The time when a system becomes available for use. Code moves from the test environment to the production environment, and the system becomes operational.
greenfield approach	Starting with a clean sheet of paper, without any constraints imposed by prior work http://www.webopedia.com/TERM/G/greenfield.html
GUI	Graphical user interface
HTTP	Hypertext Transfer Protocol
inception	In RUP, the Inception Phase involves articulating the concept of a project; determining if it is worth doing; and, if so, what resources are required. https://techterms.com/definition/rup

Term or Acronym	Definition
incremental development	A method of software development modeled on a gradual increase in feature additions and a cyclical release and upgrade pattern https://www.techopedia.com/definition/25895/iterative-and-incremental-development
information radiator	“A large, highly visible display used by software development teams to track progress.” [Atlassian 2014]
IT	Information technology
iteration	A set of activities with a plan and evaluation criteria that results in a release. Each iteration is a complete development cycle, from requirements gathering to implementation and testing. https://www.agilealliance.org/glossary/iteration/
JEE	Java Platform, Enterprise Edition
Kanban	A method for managing knowledge work with an emphasis on just-in-time delivery, while not overloading the team members http://en.wikipedia.org/wiki/Kanban_(development)
Lean	The core idea is to maximize customer value while minimizing waste. Simply, Lean means creating more value for customers with fewer resources. http://www.lean.org/WhatsLean/
phase	The span of time between two major milestones of the development process. In each phase, defined objectives are met and artifacts are completed.
PII	Personally identifiable information
pair programming	A practice in Extreme Programming in which two programmers team up and assume joint responsibility for a set of source code in order to deliver higher quality software than if each programmer were to assume individual responsibility for some subset of that code https://www.agilealliance.org/glossary/pairing/
Pattern-Enabled Development	A set of eight principles targeted specifically for Java/JavaScript application development with a pattern language http://pedcentral.com
planning poker	A free online Scrum tool that enables sprint planning through a consensus-based, gamified technique for estimating effort or relative size of development goals in software development https://www.planningpoker.com
product backlog	A prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it’s not necessary to start a project with a lengthy, up-front effort to document all requirements. Typically, a Scrum team and its product owner begin by writing down everything they can think of for Agile backlog prioritization. This Agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers. http://www.mountaingoatsoftware.com/agile/scrum/product-backlog

Term or Acronym	Definition
product owner	Has responsibility for deciding what work will be done, the single individual who is responsible for bringing forward the most valuable product possible by the desired date. The product owner does this by managing the flow of work to the team and selecting and refining items from the product backlog. The product owner maintains the product backlog and ensures that everyone knows what is on it and what the priorities are. The product owner may be supported by other individuals but must be a single person. Certainly the product owner is not solely responsible for everything. The product owner provides the requirements for the product; it is a specific role in the Scrum management process framework. For FedCLASS development, the Director of Program Alpha, as the business owner, served as the product owner. http://www.scrumalliance.org/why-scrum/core-scrum-values-roles
Program Alpha	A centralized program administered by Department Omega to perform a vital function of the U.S. federal government for federal and state agencies
RAD	Rapid application development, a development approach that emphasizes process over planning. A RAD development team adjusts requirements as it gains knowledge about users' needs. http://en.wikipedia.org/wiki/Rapid_application_development
recipe	In Chef, a recipe is the most fundamental configuration element within the organization. Authored in the programming language Ruby, a recipe is a collection of resources that defines everything required to configure part of a system. Recipes are stored in the project "cookbook." https://docs.chef.io/recipes.html
refactoring	A disciplined technique for improving the design of an existing code base, altering its internal structure without changing its external behavior https://www.agilealliance.org/glossary/refactoring/
release	A term used to group tasks that deliver some business capability. A release is the delivery of a complete set of artifacts to a user. https://www.agilealliance.org/glossary/frequent-release/
RUP	Rational Unified Process https://techterms.com/definition/rup
SAT	System acceptance test
Scrum	A management process framework defined by the Scrum Alliance http://www.scrumalliance.org/
SEI	Software Engineering Institute
service request	A service request, part of the standard process for requesting and managing the work within the data center
standups	A specific meeting within the Scrum management framework. Team members report to each other their work progress and blockers. http://www.scrum-institute.org/Daily_Scrum_Meeting.php
system of record	An information storage system that is the authoritative data source for a given data element or piece of information [Inmon 2008]
TCP/IP	Transmission Control Protocol/Internet Protocol
TDD	Test-driven development
UI	User interface

Term or Acronym	Definition
use case	A way to “describe the system’s behavior under various conditions as it responds to a request from one of the stakeholders, called the primary actor” [Cockburn 2000]
user story	User-visible functionality that can be developed within one iteration https://www.agilealliance.org/glossary/user-stories/
velocity	The amount of work done in a sprint https://www.agilealliance.org/glossary/velocity/
WIP	Work in progress, a Lean manufacturing concept of measuring work flow through the process http://en.wikipedia.org/wiki/Work_in_process
XP	Extreme Programming, an Agile process that stresses customer satisfaction, focusing on early testing, frequent incremental delivery, and responsiveness to changing customer requirements. http://www.extremeprogramming.org/

Appendix F Key Project Documents

Analysis of Alternatives: Cost and Schedule Analysis. Internal presentation. Kickoff + 12 months.

ARB Presentation Program Alpha Enterprise Architecture. Internal presentation. Kickoff – 12 months.

Business Case for the New Program Alpha System. Internal Report. Kickoff + 12 months.

Department Omega Data Center Consolidation Plan. Update. Kickoff + 2 months.

Department Omega Program Alpha FedCLASS Project Proposal. Internal presentation. Kickoff + 12 months.

Development contractor. *Top Application Roadmap*. Internal presentation. Kickoff year.

Enterprise Governance Portfolio and Project Management. Internal Presentation. Kickoff + 24 months.

Improving [Data Processing] at Department Omega. Internal presentation. Kickoff year.

Program Alpha Cost Assessment and Benchmark. Internal report. Kickoff – 12 months.

Program Alpha: Next Generation. Internal presentation. Kickoff + 12 months.

Program Alpha Production Platform Approval. Internal Presentation. Kickoff + 12 months.

Program Alpha FedCLASS Project into Agile. Internal presentation. Kickoff + 24 months.

Program Alpha System Requirements – Final. Internal report. Kickoff – 24 months.

FedCLASS Project. Internal memorandum. Kickoff year.

FedCLASS Project Status Reports. Internal status reports. Kickoff + 24 months.

Technology research firm. *Program Alpha Architecture Review – Case Study*. Internal Report. Kickoff – 24 months.

Three-Year Business and Financial Plan. Internal report. Kickoff – 48 months.

References

URLs are valid as of the publication date of this document.

[Atlassian 2014]
Information Radiators. *Atlassian*. 2014. https://confluence.atlassian.com/jira064/displaying-a-dashboard-as-a-wallboard-720416968.html

[Baker 2014]
Baker, David. Has Scrum Killed the Business Analyst? *Scrum Alliance*. September 2014. https://www.scrumalliance.org/community/articles/2014/september/has-scrum-killed-the-business-analyst

[Cockburn 2000]
Cockburn, Alistair. *Writing Effective Use Cases*. Pearson Education. 2000.

[Cockburn 2007]
Cockburn, Alistair. *Agile Software Development: The Cooperative Game (Second Edition)*. Addison-Wesley. 2007.

[Cohn 2010]
Cohn, Mike. Managing Risk on Agile Projects with the Risk Burndown Chart. *Mountaingoat Software*. 2010. https://www.mountaingoatsoftware.com/blog/managing-risk-on-agile-projects-with-the-risk-burndown-chart

[GAO 2008]
Agencies Need to Establish Comprehensive Policies to Address Changes to Projects' Cost, Schedule, and Performance Goals. GAO Report GAO-08-925. Government Accountability Office. July 31, 2008.

[Highsmith 2004]
Highsmith, Jim. *Agile Project Management*. Pearson Education. 2004.

[Inmon 2008]
Inmon, W. H.; Strauss, D.; & Neuschloss, G. *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Elsevier. 2008.

[Kundra 2009]
Kundra, Vivek. *Statement of Vivek Kundra, Federal Chief Information Officer, Administrator for Electronic Government and Information Technology, Office of Management and Budget, Before the Senate Committee Task Force on Government Performance*. U.S. Senate Committee on the Budget. 2009.

[Kundra 2010]
Kundra, Vivek. *25 Point Implementation Plan to Reform Federal Information Technology Management*. The White House. December 9, 2010.

[London 1988]
London, Manuel. *Change Agents*. Jossey-Bass Publishers. 1988. ISBN 1-55542-107-5.

[Manifesto 2001]
Manifesto for Agile Software Development. 2001. http://agilemanifesto.org

[NIST 2013]
National Institute of Standards and Technology Baldrige Performance Excellence Program. *2013–2014 Criteria for Performance Excellence*. NIST. 2013.

[RSC 1998]
Rational Software Corporation. *Rational Unified Process: Best Practices for Software Development Teams*. White Paper TP026B Rev 11/01. RSC. 1998.

[Schwaber 2013]
Schwaber, Ken & Sutherland, Jeff. *The Scrum Guide: The Definitive Guide to Scrum*. Scrum Alliance. July 2013. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

[Shore 2004]
Shore, Jim. Fail Fast. *IEEE Software*. Volume 21. Number 5. 2004. Pages 21–25.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	May 2018	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
FedCLASS: A Case Study of Agile and Lean Practices in the Federal Government	FA8721-05-C-0003

6. AUTHOR(S)

Nanette Brown, Jeff Davenport, Linda Parker Gates, Jon Gross, and Tamara Marshall-Keim

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2018-SR-016

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This case study tells the story of the development of a critical IT system within an executive department of the U.S. federal government, using iterative, Agile, and Lean development methods and cloud-based technologies. This study reports the successes and challenges of using this new development approach in a government software development environment so that other government entities can benefit from the experiences of this project. The study is based on conversations with team members, observations of team activities, and examination of work products, documentation, and program guidance. The report describes the organizations responsible for creating the software solution, establishing the development process, and structuring acquisition activities. It then details the product development process in chronological order and describes the development approaches and technologies. It also puts events into the context of external environmental influences to present a development effort as it confronts real-world challenges. The final section describes insights gleaned during the research of this case study and includes analysis of the organization's experiences with Agile and Lean adoption, technical approaches, and project leadership. These insights may benefit future Agile projects in the federal government and the software engineering community as a whole..

14. SUBJECT TERMS	15. NUMBER OF PAGES
Agile, case study, federal government, incremental development, Lean, software development	68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102