TREASURY INSPECTOR GENERAL FOR TAX ADMINISTRATION

The Individual Tax Processing Engine Project Is Making Progress

September 14, 2020

Reference Number: 2020-20-062

TIGTACommunications@tigta.treas.gov | www.treasury.gov/tigta | 202-622-6500

To report fraud, waste, or abuse, please call us at 1-800-366-4484

HIGHLIGHTS: The Individual Tax Processing Engine Project Is Making Progress

Final Audit Report issued on September 14, 2020
Reference Number 2020-20-062

Why TIGTA Did This Audit

The Customer Account Data Engine 2 Program, chartered in 2009, is one of the most complex modernization programs in the Federal Government and involves major changes to core IRS tax processing systems. This audit was initiated to determine whether the IRS is effectively and efficiently managing the Customer Account Data Engine 2 program's Individual Tax Processing Engine project with a focus on velocity estimates and development.

Impact on Taxpayers

The IRS Integrated Modernization Business Plan states that a key project supporting the Customer Account Data Engine 2 is the Individual Tax Processing Engine project, which will convert lines of legacy Assembly Language Code to Java, a modern software language. This code conversion is a major milestone towards retiring the Individual Master File.

The Customer Account Data Engine 2 is intended to provide state-of-the-art individual taxpayer account processing as well as data-centric technologies to improve service to taxpayers. However, deployment delays and cost overruns can decrease stakeholder and public confidence in the IRS's ability to develop, monitor, and use its resources effectively to deliver improved taxpayer services.

What TIGTA Found

The primary goal of the Individual Tax Processing Engine project is to reengineer the Individual Master File, written in an old programming language (Assembly Language Code), into a modern programming language (Java). The IRS implemented a scenario-based approach for Java code development. TIGTA determined that this is an effective approach given the size and complexity of the Individual Master File.

The IRS is effectively monitoring the progress of the Individual Tax Processing Engine project. Project planning meetings, project monitoring meetings, and Integrated Project Team meetings occur at regular intervals. The Integrated Project Team meetings are intended to discuss project status, hot topics, and next steps. The results of these meetings are documented. In addition, comparisons of planned work versus actual work completed are reported weekly. As a result of their monitoring efforts, IRS management has identified and mitigated development challenges. For example, the IRS approved additional resources to increase project velocity (*i.e.*, how much work can be completed in each product increment iteration) and extended the project's schedule to compensate for the time lost during the Government Shutdown.

The IRS is using a Trajectory Model to estimate the planned velocity goals of the project and to track whether the goals are met. An updated Trajectory Model was used in September 2019 and will be updated approximately every seven and a half months. In our next review, TIGTA will fully analyze the effectiveness of the updated Trajectory Model.

The IRS developed Java code that complied with documented guidelines for the Java declaration and statement standards and are applied in the 58 files reviewed. The IRS developed Java code that generally conforms to industry best practices, but some best practices were not followed. For example, Java code files contained lines in excess of 100 characters, files are longer than 2,000 lines, and opening comments are incomplete or missing. According to the IRS, these deviations from best practices do not affect the quality of the code or runtime, but future maintenance could be inefficient.

What TIGTA Recommended

TIGTA made no recommendations as a result of the work performed during this audit.

TREASURY INSPECTOR GENERAL
FOR TAX ADMINISTRATION

U.S. DEPARTMENT OF THE TREASURY

WASHINGTON, D.C. 20220

September 14, 2020

MEMORANDUM FOR: COMMISSIONER OF INTERNAL REVENUE

FROM: Michael E. McKenney
Deputy Inspector General for Audit

SUBJECT: Final Audit Report – The Individual Tax Processing Engine Project Is Making Progress (Audit # 202020015)

This report presents the results of our review to determine whether the Internal Revenue Service (IRS) is effectively and efficiently managing the Customer Account Data Engine 2 program's Individual Tax Processing Engine project with a focus on velocity estimates and development. This review is part of our Fiscal Year 2020 Annual Audit Plan and addresses the major management and performance challenge of *Modernizing IRS Operations*.

Management's complete response to the draft report is included as Appendix V.

Copies of this report are also being sent to the IRS managers affected by the report. If you have any questions, please contact me or Danny R. Verneuille, Assistant Inspector General for Audit (Security and Information Technology Services).

Table of Contents

Background

The Customer Account Data Engine (CADE) 2 Program, chartered in 2009, is one of the most complex modernization programs in the Federal Government and involves major changes to core Internal Revenue Service (IRS) tax processing systems. CADE 2 is a relational database[1] that contains data from the Individual Master File (IMF) and is intended to provide state-of-the-art individual taxpayer account processing as well as data-centric technologies to improve service to taxpayers. In order to limit risk and demonstrate incremental progress toward the target solution, the IRS created transition states. CADE 2 is currently progressing through the largest and most critical transition state: Transition State 2. The primary goal of Transition State 2 is to reengineer the IMF, written in an old programming language called Assembly Language Code (ALC), into a modern programming language (Java[2]). In April 2016, the IRS chartered the CADE 2 Individual Tax Processing Engine (ITPE) project to update the programming language to Java.

The ITPE project is critical to migrating IMF core programs to Java. It can take years for ALC developers to become proficient in understanding the business logic of the unique IMF core processing programs. In addition, ALC has limited capabilities as a programming language, and maintaining the IMF is difficult because of the decreasing number of ALC programmers available. The ITPE project will help address these issues by updating the code to Java. Comparatively, Java will provide a platform for future development and improved maintainability of the IMF code, and Java programmers are widely available. This is a modernization effort with the intent of maintaining all current functionality and capabilities.

At the outset of the CADE 2 Program, the IRS did not have an existing capability or commercially available product to convert ALC to Java. Converting ALC to Java is a highly complex process because there are fundamental differences between the two programming languages. ALC is a low-level programming language that is one step up from machine language (*e.g.*, a string of 0's and 1's that represent instructions understood by a computer). ALC contains few recognizable human words and uses mnemonic code[3] so that programmers do not have to memorize or look up instructions for every numerical string of code. In addition, low-level programming languages are used to write programs that relate to the specific architecture and hardware of a particular type of computer. In this case, the IMF uses a mainframe architecture that runs ALC programs in operation since Calendar Year 1963. Conversely, Java is a high-level programming language that is written with words and phrases which are close to human language. Programming languages are considered high-level because they are far removed from the machine code instructions understood by the computer. High-level programming languages create programs that are portable across platforms and are not tied to a particular computer or architecture. As a result, programmers using high-level programming languages like Java are not required to be knowledgeable of the hardware architecture.

In Calendar Year 2014, the IRS began work on an in-house automated method called the Auto-Translator Tool (ATT) to translate the IMF's ALC into Java. The ATT extracts ALC business and logical functions and data by scanning and parsing the source lines of code (LOC) and

[1] See Appendix VI for a glossary of terms.
[2] Java is an IRS-wide development preference.
[3] Widely used in computer programming to specify instructions.

identifying how the data are stored in physical memory. The tool also analyzes the structure of ALC to provide useful statistics, such as the well-formed subroutines and the number of self-modified code for certain patterns. The ATT also provides information to the Java Runtime Environment. The ATT converts ALC execution logic into Technical Rules Language by identifying patterns in the source code and converting these patterns into Java. The tool also uses a byte-for-byte comparison approach to verify that the Java code will produce the same outcome as the ALC code. The IRS planned to use the ATT as the method to convert ALC into Java for IMF Runs 12 and 15.[4] The new Java applications will take the same inputs and outputs as ALC IMF for these processing runs.

Beginning in February 2015, the Information Technology organization's Applications Development and Enterprise Services functions initiated a joint effort to explore the capabilities of the ATT and its methodology. They selected and translated a portion of the IMF ALC code using the ATT to determine the level of effort required for migrating to Java with this tool.

Between October 2015 and July 2017, the IRS performed three assessments (two external and one internal) of the ALC to Java conversion effort using the ATT. The combined results of these assessments led the IRS to the project strategy used today.

- In October 2015, the MITRE Corporation completed the first assessment. One of its three key findings was that the ATT will not achieve the performance requirements needed to meet the project schedule. The MITRE Corporation also stated that the ATT will replicate the legacy data structures and nonmodular program design of the ALC instead of using the full capabilities of Java, resulting in unmaintainable and unnecessarily complex Java. A third finding was that the converted Java code does not adequately capture business rules in the legacy code.
- In December 2016, the Applications Development function completed the second assessment of the ATT internally. The IRS reported two issues related to the ATT's converted code for IMF Run 12. First, the ATT-generated Java code was more complex than the original ALC or standard Java. In addition, continued use of the ATT would require developers to learn IBM mainframe architecture, thereby diminishing one of the main benefits of converting the legacy ALC code to a modern, high-level programming language.
- In July 2017, Deloitte completed the third assessment of the ATT. Deloitte recommended that the IRS change direction from implementing the translated Java code into production and instead use the translated code produced by the ATT as an input tool.

In September 2017, the IRS decided not to implement the ATT's translated Java code. Instead, the IRS implemented a scenario-based approach that is a five-step process for Java code development. The process begins with the identification of real-world IMF business scenarios (*e.g.,* tax assessment, payment, and adjustments). These scenarios are used to identify pieces of code functionality, or Building Blocks, to be incrementally designed, developed, and tested.

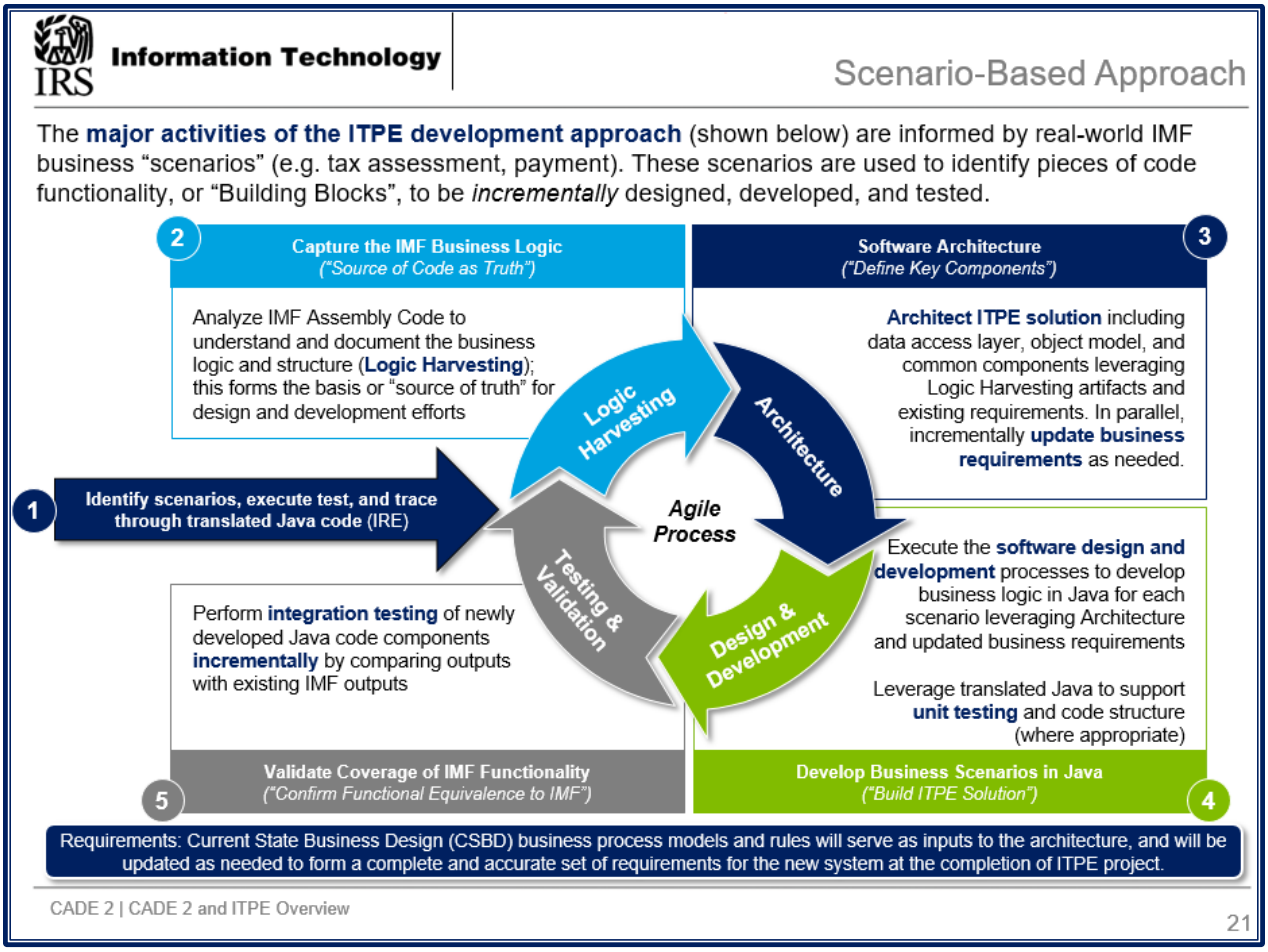
[4] These programs perform the core IMF business functions of Posting, Settlement, and Analysis, and are the most complex IMF programs.

Results of Review

A Scenario-Based Approach Was Adopted

In September 2017, the IRS documented its scenario-based approach to convert legacy ALC to Java. This approach uses business scenarios based on IMF business transactions to incrementally implement IMF functionality into an end-to-end solution. This process will implement a common Data Access Layer and align to the target state architecture. The Java code produced by the ATT was repurposed as an input to the scenario-based approach. Figure 1 outlines the IRS's scenario-based approach for the ITPE project.

Figure 1: Overview of the Scenario-Based Approach



Source: CADE 2 and ITPE Overview, dated July 17, 2019. IRE = IMF Reverse Engineering.

In July 2019, we met with the ITPE project team for an overview of the project. IRS management stated that they were moving forward with a scenario-based approach for IMF Runs 12 and 15, which are the bulk of tax processing logic. The purpose of using the scenario-based approach is to facilitate the iterative delivery of the ITPE project and improve workflow. We also performed research to determine if other approaches for converting ALC to Java should be considered. We found one example of a private sector company successfully converting ALC to Java. However,

the scope of the conversion was 10,000 LOC total. By comparison, IMF Runs 12 and 15 alone have approximately 146,000 active LOC. Overall, the IMF has approximately 961,000 LOC total. We determined that the scenario-based approach is effective for the ITPE project given the size and complexity of the IMF. We also found this approach was consistent with the Internal Revenue Manual (IRM)[5] Agile path development guidance, which includes high-level feature definitions and allows for repetitive cycles of development and testing for a product or new solution.

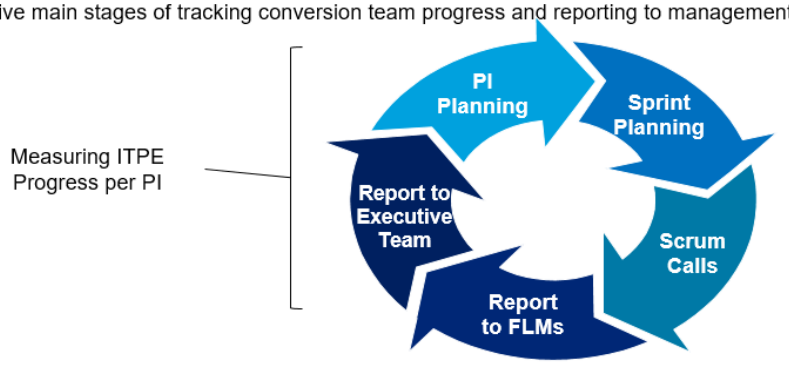
The Individual Tax Processing Engine Project Is Effectively Monitored

The ITPE project is broken down into 24 product increments. Each product increment is comprised of five two-week sprints, totaling 10 weeks in duration. The IRS held various meetings to plan and monitor the ITPE project during each product increment. Planning meetings occurred at the beginning of each product increment as well as at the beginning of each of the five sprints. Monitoring meetings occurred at various frequencies. Figure 2 provides an overview of the process for monitoring and measuring progress.

Figure 2: Overview of the Process for Monitoring ITPE Progress

IRS **Information Technology** | b) An overview of the process for monitoring and measuring ITPE progress

Based on the Targets and Projections calculated by the Trajectory Model, ITPE Progress is measured in several different stages throughout each 10-week Product Increment (PI), based on the targets and forecasts contained in the trajectory model. The five main stages of tracking conversion team progress and reporting to management are detailed below.



Meeting Type	Frequency	Description	Key Persons
Product Increment (PI) Planning	Every 10 weeks	Front Line Managers (FLM), Team Leads and Scrum Masters set Conversion Team objectives for the 10-week Program Increment.	Senior Managers, FLMs, Team Leads, Scrum Masters,
Sprint Planning	Every 2 weeks	Team Leads and Scrum Masters breakdown the PI objectives into 2-week sprint goals.	Team Leads, Scrum Masters
Conversion Team Scrum Calls	Daily	Scrum Masters discuss any team impediments and record all team progress on IBM Rational Team Concert (RTC). Team progress is compared against PI and sprint goals.	Conversion Team Scrum Masters
Report to FLM	Twice per week	Team Leads report progress to FLM and discuss any existing impediments for completing PI or sprint objectives.	Team Leads, FLMs
Report to Executive Team	Weekly	FLMs report conversion team status to the AD Executive Team. Discuss whether conversion teams are on track to complete PI and sprint-level objectives.	Senior Managers, FLMs, AD Executive Team

Source: Provided by ITPE Project Management on January 15, 2020. AD = Applications Development; IBM = International Business Machines.

[5] IRM 2.16.1, *Enterprise Life Cycle* (Nov. 26, 2019).

During our audit, we attended the Product Increment-11 planning meetings. The meetings started with a group session to discuss Product Increment-10 accomplishments and Product Increment-11 overall objectives. Next, the different functional teams (*e.g.*, testing, ALC conversion teams, architects) met separately to discuss their specific Product Increment-11 objectives. At the end of the individual functional team sessions, the planning outcomes were documented and included the objectives and risks identified by each team.

In addition, ITPE Integrated Project Team meetings are held every two weeks. The meeting participants include a broader audience than those listed in Figure 2 (*e.g.*, the Cybersecurity function and the Wage and Investment Division's Modernization, Tools, and Technologies function). The purpose of the Integrated Project Team meetings is to discuss project status, hot topics, and next steps. Meeting minutes and a presentation deck are prepared for each Integrated Project Team meeting.

As a result of their ongoing monitoring efforts, IRS management identified the following challenges:

- **Insufficient Backlog of Building Blocks.** A bottleneck occurred due to an insufficiently established backlog of Building Blocks, *i.e.*, identified pieces of code with common functionality, for design and development. An insufficient backlog prevents the architects from providing complete designs to the development teams. This decreases development velocity and increases refactoring. In addition, the development end date did not initially include the work for the Technical Framework. Examples of items included in the Technical Framework are Technical Enablers and the Data Access Layer. The Data Access Layer is calculated based on input and output files. However, the exact percentage that the Data Access Layer makes up of the overall Technical Framework was unknown.
- **Insufficient Resources.** A limited knowledge of the project complexity led to a limited knowledge of the skillsets needed to complete the project. In addition to identifying the requisite skillsets, the ITPE project also needed additional human resources in order to increase project velocity. In December 2018, the project obtained approval to add two conversion teams.[6] However, the ITPE project received resources to add four new conversion teams. Currently, new Conversion teams 5, 6, 7, and 8 are staffed with 13, 12, 7, and 8 people, respectively. Team members are shifted as needed to fill specific needs, usually at the beginning of a product increment.
- **Implementing Tax Cuts and Jobs Act of 2017.**[7] This extended the schedule by two product increments (five months) because the Act included wider and deeper changes to the IMF code, resulting in a larger-than-normal 2019 Filing Season update to the IMF and, subsequently, the ITPE-related code.
- **Government Shutdown** (December 2018 through late January 2019). This delayed the onboarding of new personnel resources, which extended the schedule by two product increments (five months).

[6] Responsibilities include converting ALC to Java code.

[7] Pub. L. No. 115-97. Officially known as "An act to provide for reconciliation pursuant to titles II and V of the concurrent resolution on the budget for Fiscal Year 2018."

- **Coronavirus Aid, Relief, and Economic Security Act.**[8] The IRS identified a risk that if key resources are lost over an extended period of time, the velocity of ALC to Java conversion and testing will be affected and the ITPE project development will not be completed by September 2022. Eight resources were diverted to Act-related activities. The IRS stated that it made adjustments to the Trajectory Model to account for the 15 percent loss of productivity and that using the backlog of Building Blocks resulted in no impact to the development end date.

To address the ITPE project velocity challenges,[9] management implemented the following actions:

- Revised the scope of Product Increment-8 from the ALC conversion to focus on building a sufficient backlog of Building Blocks ready for development in Product Increment-9, developing Technical Enablers, and training new resources.
- Received approval for 40 additional resources (30 IRS employees and 10 contractors) in December 2018. The additional resources were used to staff the two new conversion teams and to fill skill gaps in the four existing conversion teams.
- Added four product increments in Fiscal Year 2019.
- Streamlined processes in order to reduce overhead (*e.g.*, reducing the number of meetings for team leads).
- Restructured the teams, thereby increasing collaboration between code development and logic harvesting.
- Held three CADE 2 ITPE Acceleration Summits. The purpose of the summits were to create a forum to build a higher level of trust across organizations and to identify and review updates on ITPE velocity bottlenecks and potential solutions to address them. These summits resulted in action-oriented plans to evaluate and implement solutions.

Identifying and mitigating risks and issues are part of managing and monitoring the project. The ITPE Project Management Plan states that project managers are responsible for tracking, monitoring, and reporting risks and issues. The ITPE project team documents risks and issues in the ITPE Integrated Project Team meeting documents and participates in the CADE 2 Program Risk Reviews, which also manages ITPE risks and issues. When warranted, risks and issues were also reported in the Item Tracking Reporting and Control system. For example, on May 22, 2018, Risk Number 29934 was submitted because the Applications Development function had not maintained a working pace that would ensure that it completed the ALC to Java conversion in time to perform parallel validation in Fiscal Year 2021. On November 6, 2018, it was elevated to an issue because the project continued to miss the velocity targets.

The IRM states that Agile development projects like the ITPE project should prepare reports at the end of each sprint to summarize the progress made, obtain feedback and approvals from stakeholders, and adjust planning for subsequent iterations. It also requires projects to establish reasonable plans for managing development projects and performing engineering tasks. In addition, the *ITPE Project Management Plan*[10] states that management is responsible

[8] Pub. L. No. 116-136, 134 Stat. 281.
[9] See Appendix II for the Evolution of ITPE Velocity over time that describes the challenges addressed to improve ITPE project velocity.
[10] IRS, *Project Management Plan for ITPE*, *Version 2* (Aug. 7, 2018).

for ensuring that all elements of the project are monitored and controlled and that information management requirements are satisfied. According to the plan, this will be accomplished by monitoring the actual performance and progress of the project against the planned baselines for scope, schedule, and cost. When performance deviates from the plan, management will take appropriate corrective actions, including escalation or revising the plan, estimates, and schedule, as defined by the approving authority. Management is also responsible for monitoring risks and issues according to the organizational risk management and contingency management procedures.

Generally, we found that the IRS is effectively monitoring the progress of the ITPE project. We reached this conclusion by attending project meetings, reviewing minutes and reports, and tracking the project's progress during our audit. We also assessed ITPE project monitoring against IRM and agency directives.[11]

An Updated Process Is Used to Measure Project Progress

The IRS has taken steps to improve the process for estimating the development time required to convert LOC from ALC to Java. At the outset of the ITPE project, the IRS identified the complexity of the IMF ALC containing several irregular coding conventions that do not exist in modern programming languages as a constraint. CADE 2 management stated that when they established initial development estimates in Fiscal Year 2017, the level of effort required to develop the ITPE scope was unknown.

The IRS chose LOC as the method to estimate the size of the ITPE development effort. There are 146,000 LOC to convert plus 68,000 LOC-equivalents for Technical Enablers, totaling 214,000 LOC for the entire ITPE project. To measure the progress throughout Fiscal Year 2019, the IRS identified four Confidence Milestones to measure overall project health. The IRS stated that velocity was the most significant of the Fiscal Year 2019 Confidence Milestones because it provided insight into the ALC to Java LOC conversion velocity, a major quantifiable metric. Due to the velocity challenges that the project faced, the ITPE project did not achieve the Velocity Confidence Milestones for Product Increments 2 through 8.[12] The Confidence Milestones were discontinued for Fiscal Year 2020. However, the IRS continued to monitor the ITPE velocity by comparing planned LOC work to actual work completed.

In April 2019, the IRS stated that it established an initial Trajectory Model to track and monitor velocity metrics, but it did not account for all work to be completed. In August 2019, the CADE 2 Program Management Office worked with a contractor to create the *CADE 2 Program Management Office Trajectory Model* (hereafter, references to the Trajectory Model refer to this one). In September 2019, the CADE 2 Program Management Office used data from Product Increments 6, 7, and 8 to update the Trajectory Model to project the ALC LOC conversion for each product increment, starting with Product Increment-9. Because of the extensive analysis to account for the ITPE project's complexity and capturing all work required, the updated Trajectory Model determined that the development end date for ITPE moved from August 2021 to September 2022.

[11] IRM 2.5.1, *Systems Development* (Sept. 1, 2006).

[12] See Appendix III for the Velocity Confidence Milestones and the actual LOC converted for Product Increment-1 through Product Increment-8.

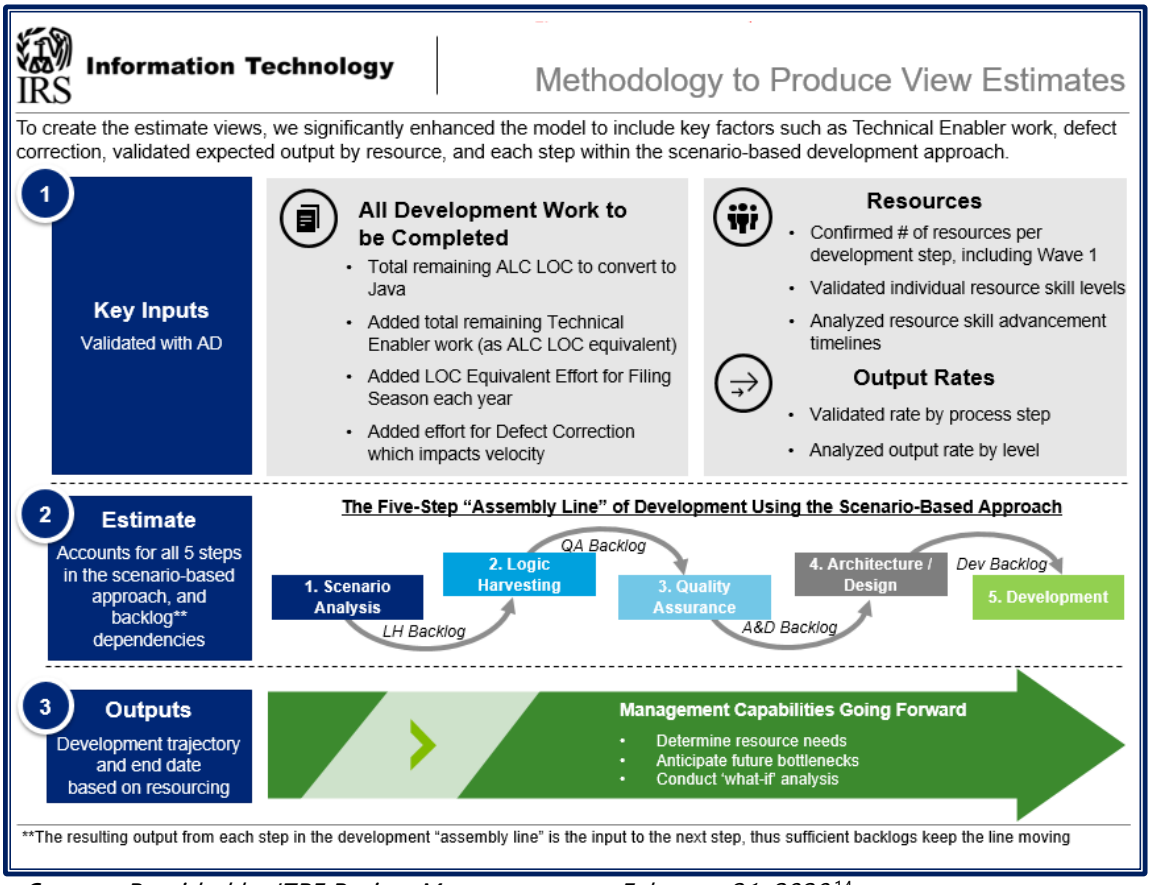
To ensure that the Trajectory Model is in sync with how development is operating and to give the most precise projections, the Trajectory Model is updated after the completion of every third product increment, a period of approximately seven and a half months.[13] Routine updates include:

- Validating the current resource list and skill level.
- Validating the assumptions outlined in the Trajectory Model based on data from the product increments.
- Including actual outcomes from each product increment.
- Identifying whether there are process changes that occurred that need to be incorporated into the Trajectory Model.

Other inputs to the Trajectory Model include total ALC LOC and LOC-equivalents for Technical Enablers. In addition, numerous assumptions are made to further estimate the project trajectory. Some examples of these assumptions are percent reduction in productivity for coaching, percent reduction in productivity for correcting defects, filing season LOC-equivalents, and vacations, leave, and holidays. Figure 3 provides more information about the methodology and data used to update the Trajectory Model.

[13] Calculated by dividing 30 weeks (three product increments times 10 weeks) by four weeks per month which equals approximately seven and a half months.

Figure 3: Key Data for Updating the Trajectory Model

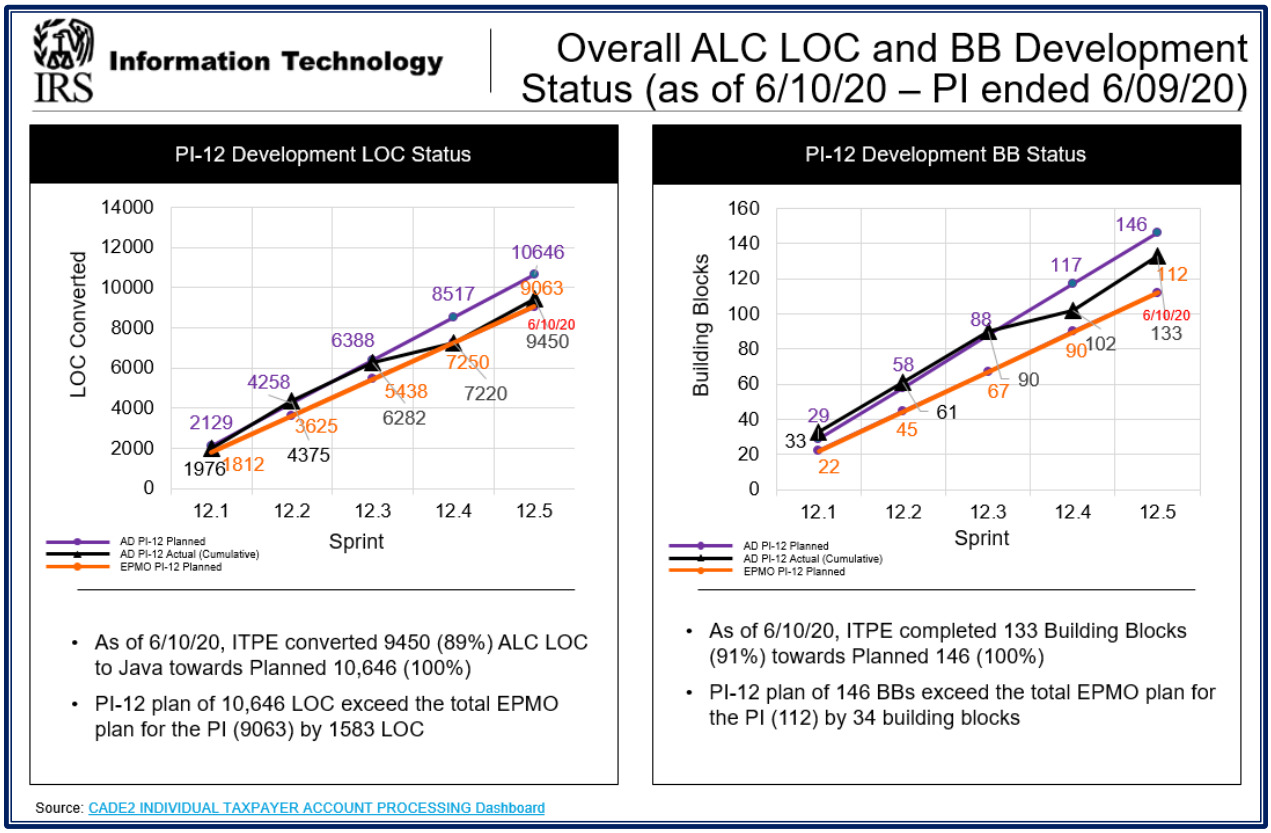


Source: Provided by ITPE Project Management on February 21, 2020.[14]

In weekly reports to IRS executives, the *CADE 2: ITPE Weekly Executive Update* compares the actual work completed to both the Enterprise Program Management Office and Applications Development function planned work estimates. IRS management explained that the Enterprise Program Management Office estimate is determined by the Trajectory Model and is the minimum work needed to meet the revised September 2022 development end date. The Applications Development sprint teams set goals for themselves during the product increment and sprint planning based on multiple factors. They are encouraged to set challenging goals, and those numbers become the Applications Development function's targets for the product increment. When reporting these metrics to the Chief Information Officer, the actual numbers are compared against the Enterprise Program Management Office estimates. Figure 4 provides an example of how this information is reported for the LOC conversion and Building Block development work performed during Product Increment-12.

[14] Additional information about updating the Trajectory Model is included in Appendix IV.

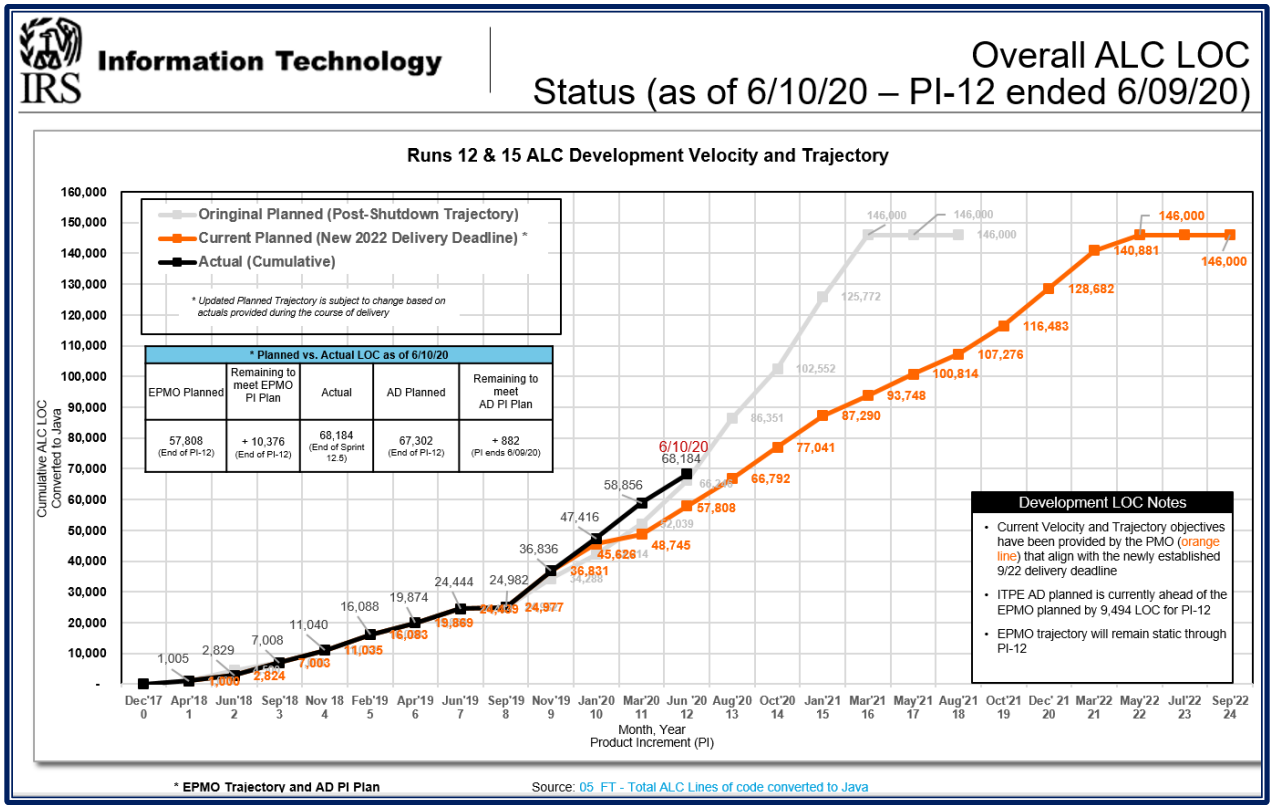
Figure 4: Comparison of Planned Estimates of LOC Conversion and Building Blocks to Actual Work Completed During Product Increment-12



Source: CADE 2: ITPE Weekly Executive Update dated June 11, 2020. AD = Applications Development; BB = Building Block; PI = Product Increment; EPMO = Enterprise Program Management Office.

By comparison, Figure 5 shows the overall cumulative planned and actual work for the ITPE LOC conversion development as of June 10, 2020. For tracking purposes, the Enterprise Program Management Office cumulative planned work of 57,808 LOC is included in the graph on the orange line. The Applications Development function cumulative planned work of 67,302 LOC appears in a box within the graph. As of June 10, 2020, the cumulative total actually completed is 68,184 LOC (47 percent of the overall ITPE project).

Figure 5: Cumulative ALC LOC Development Status for Runs 12 and 15



Source: Provided by the CADE 2 Program Management Office on August 5, 2020. BB = Building Block; EPMO = Enterprise Program Management Office; AD = Applications Development; PMO = Program Management Office.

We determined that the IRS's current estimation process incorporates Government Accountability Office best practices[15] to estimate the duration of the ITPE project and velocity rate. For example, the Government Accountability Office states that estimators should understand interdependencies that affect the schedule. Some examples of interdependencies are staff availability, effective work hours per shift, and downtime from meetings, travel, and sickness. The Trajectory Model accounts for these interdependencies and many more, such as staff skill level (*e.g.*, beginner, intermediate, advanced) and project role (*e.g.*, developer, design architect, test); percent reduction in productivity for coaching; percent reduction in productivity for correcting defects; and filing season LOC-equivalents. The Government Accountability Office also states that scheduling is complicated, and the more complex the software development effort is, the harder it will be to find the right staff for the job.

According to the *Agile Practice Guide,*[16] it can take four to eight iterations to achieve a stable and predictable project velocity. Due to the complexity of the ITPE project and the mix of experience levels, it may take three product increments for the teams to complete work at a stable velocity. The IRS can adjust the time period for updating the Trajectory Model once this stability has been achieved. The next update to the Trajectory Model was initially scheduled to

[15] Government Accountability Office, GAO-09-3SP, *Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs* (Mar. 2009).
[16] Project Management Institute, Inc., *Agile Practice Guide* (2017)

be completed by April 6, 2020, after the end of Product Increment-11.[17] However, the update was postponed and rescheduled for the first week in June 2020 because the IRS wanted to update the Trajectory Model with the final data from Produce Increment-12, which did not end until June 9, 2020. On June 11, 2020, the IRS met with us and provided the updated Trajectory Model. In our next review, we will fully analyze the effectiveness of the updated Trajectory Model.

The Government Accountability Office guidance states that estimating software size is not easy and depends on having a detailed knowledge about a program's functions in terms of scope, complexity, and interactions. We found that the IRS has documented the scope of the ITPE project's complexity and has taken interactions into account with its scenario-based approach. Frequent reporting based on LOC converted will inform management of any roadblocks and will better inform Trajectory Model updates when tracking project velocity.

Java Code Generally Aligns With Industry Best Practices

The IRS provided 235 Java class files so we could review the new ITPE code. We reviewed a judgmental sample[18] of 58 (25 percent) Java class files, totaling approximately 42,000 LOC, and found that 48 (83 percent) of the 58 files had lines in excess of 100 characters. We also found that five (9 percent) of the 58 files contained more than 2,000 lines. In addition, every file reviewed had incomplete or missing opening comments. A detailed comment review determined that all files did not consistently use the beginning comments section of the code as outlined by the IRM.[19] In our sample review of Java files, we identified 14 files (24 percent) that had a blank comment field with no information. One file (2 percent) had beginning comments with no date or class name listed. Two files (3 percent) had beginning comments that were lacking a class name, version, and date entry. Lastly, 58 files (100 percent) contained beginning comments that did not include a class name. All the files we reviewed did, however, comply with guidelines for Java declaration standards and Java statement standards. Figure 6 summarizes the results from our Java code analysis.

Figure 6: Assessment of Java Code Sample

Files reviewed	Files With Lines Longer Than 100 Characters	Files Longer Than 2,000 Lines	Files With Missing or Incomplete Comments	Declaration Controls Followed	Statement Controls Followed
58	48 (83%)	5 (9%)	58 (100%)	Yes	Yes

Source: Auditor assessment of Java files as part of the ITPE development process.

The IRM states that Java declaration standards are described for consistent code creation. The IRM also documents various Java statement standards to ensure consistent creation and logic flows. In addition, the IRM outlines standards for the format and use of Java exceptions, acceptable naming conventions, and best practices, all with the goal of creating uniform and

[17] Product Increment-11 began on January 22, 2020, and ended on March 31, 2020.
[18] A judgmental sample is a nonprobability sample, the results of which cannot be used to project to the population.
[19] IRM 2.5.3, *Systems Development, Programming and Source Code Standards* (Mar. 1, 2007).

readable Java code. We concluded that the ITPE Java code we reviewed with regards to declaration and statement controls aligns with the IRM and best practices.

The IRM and industry best practice guidelines[20] identify numerous practical techniques and quantifiable metrics to be followed. Elements described within the guidelines include but are not limited to programmers avoiding files longer than 2,000 lines as they are cumbersome for other programmers to follow. Programmers should break LOC at column 100 to maintain readability. Code within all source files are to begin with a C-style comment that lists the class name, version information, and date. Comments serve as a mechanism to provide an overview and additional information of Java code that is not readily available in the written code. According to the IRS, the existence of Java LOC in excess of 100 characters and files in excess of 2,000 LOC as well as the lack of opening comments do not affect the quality of the code or have any impact on the code at runtime, but these deviations from best practices could make future maintenance inefficient.

[20] Sun Microsystems, *Java Code Conventions* (Sept. 12, 1997).

Appendix I

Detailed Objective, Scope, and Methodology

Our overall objective was to determine whether the IRS is effectively and efficiently managing the CADE 2 program's ITPE project with a focus on velocity estimates and development. To accomplish our objective, we:

- Evaluated the challenges causing the ITPE project to not meet the planned velocity metrics by 1) comparing the IRMs and other guidance to the IRS's actual processes for estimating and managing Agile development projects and 2) determining whether these processes were effective.
- Evaluated the effectiveness of the ALC to Java conversion process by 1) reviewing the IRS's methodology for researching and selecting approaches for converting ALC to Java and 2) determining whether the completed Java source code followed best practices and procedures.

Performance of This Review

This review was performed with information obtained from the Information Technology organization at the New Carrollton Federal Building located in New Carrollton, Maryland, during the period November 2019 through June 2020. We conducted this performance audit in accordance with generally accepted government auditing standards. Those standards require that we plan and perform the audit to obtain sufficient, appropriate evidence to provide a reasonable basis for our findings and conclusions based on our audit objectives. We believe that the evidence obtained provides a reasonable basis for our findings and conclusions based on our audit objectives.

Major contributors to the report were Danny R. Verneuille, Assistant Inspector General for Audit (Security and Information Technology Services); Jena Whitley, Director; Michael Mohrman, Audit Manager; Tina Wong, Lead Auditor; and Nicholas Reyes, Senior Auditor.

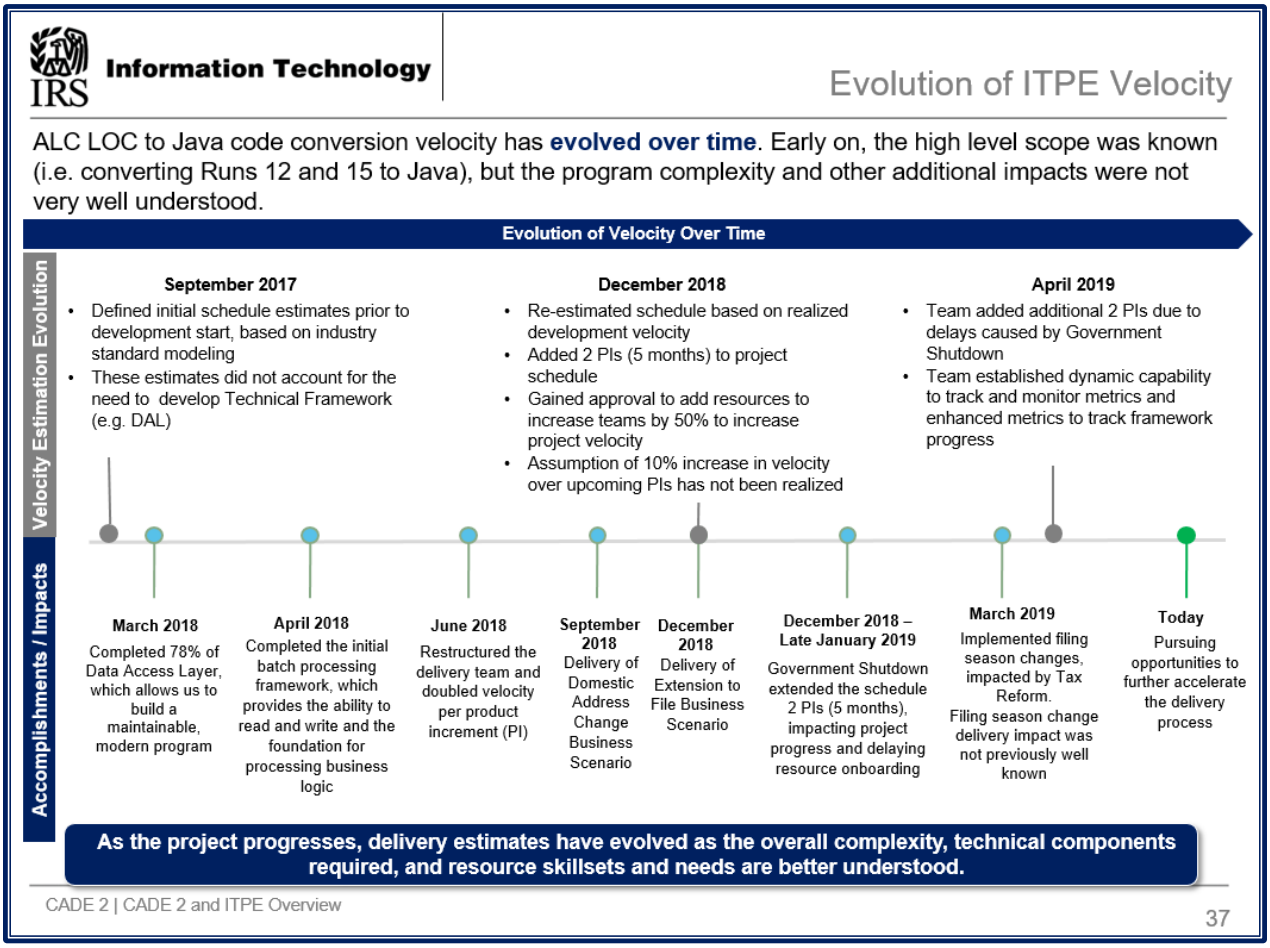
Internal Controls Methodology

Internal controls relate to management's plans, methods, and procedures used to meet their mission, goals, and objectives. Internal controls include the processes and procedures for planning, organizing, directing, and controlling program operations. They include the systems for measuring, reporting, and monitoring program performance. We determined that the following internal controls were relevant to our audit objective: policies, procedures, and best practices related to estimating software development time, monitoring information technology projects, developing Java code, and researching and selecting approaches for converting ALC to Java. We evaluated these controls by interviewing IRS employees and contractors, evaluating status reports, analyzing a judgmental sample[1] of Java files, and reviewing other relevant project documentation.

[1] A judgmental sample is a nonprobability sample, the results of which cannot be used to project to the population.

Appendix II

Evolution of Individual Tax Processing Engine Velocity



Source: CADE 2 and ITPE Overview dated July 17, 2019. DAL = Data Access Layer.

Appendix III

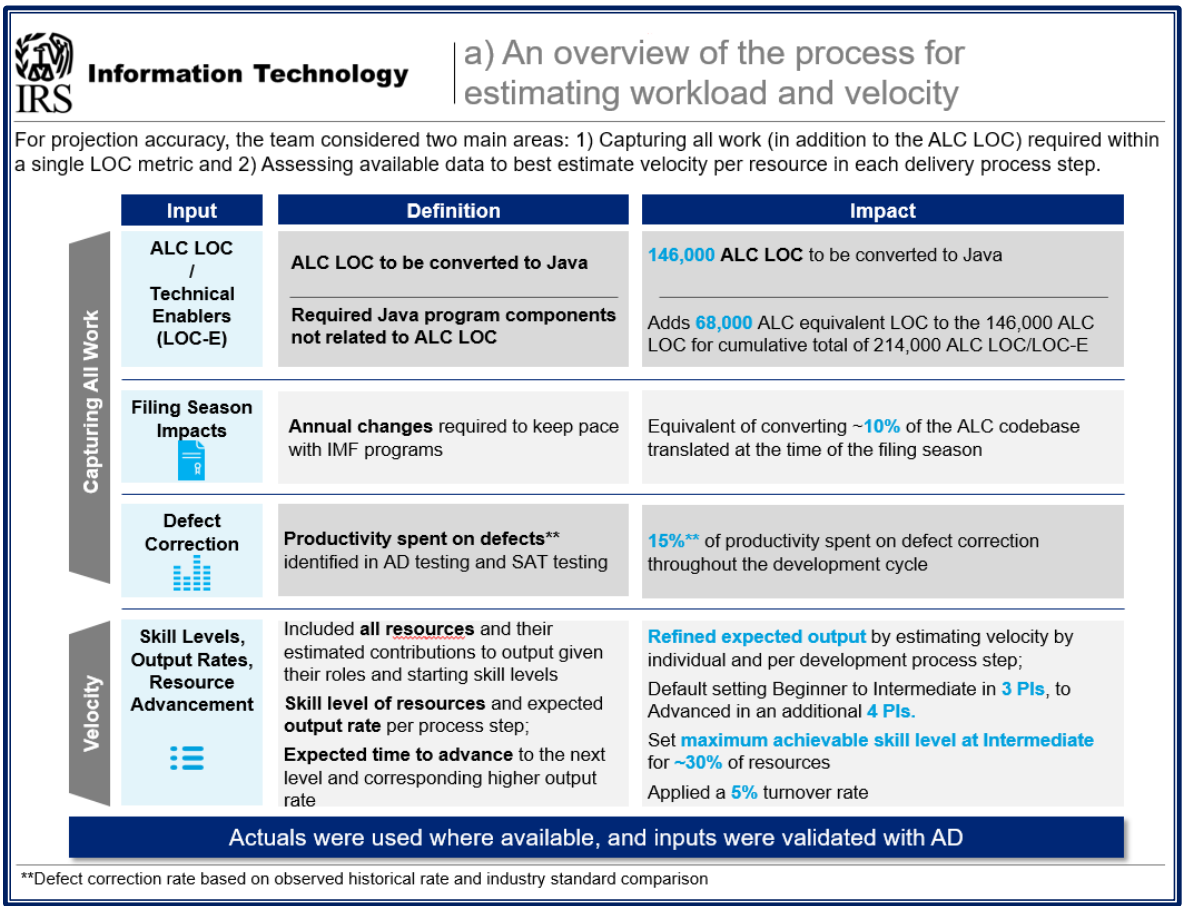
Velocity Confidence Milestones and Lines of Code Completed

Product Increment	Product Increment Dates	Velocity Confidence Milestone	Actual ALC LOC Completed	Product Increment Goal Met?
1	01/29/2018 – 04/06/2018	1,000	1,005	Yes
2	04/12/2018 – 06/19/2018	3,500	1,824	No
3	06/20/2018 – 08/28/2018	7,000	4,179	No
4	08/29/2018 – 11/06/2018	4,000–5,000	4,032	Yes
5	11/07/2018 – 02/05/2019	5,000–6,500	5,048	Yes
6	02/06/2019 – 04/16/2019	6,000–7,500	3,786	No
7	04/17/2019 – 06/25/2019	7,500–11,000	4,570	No
8	06/26/2019 – 09/03/2019	8,500–12,500	538	No
	Totals	**42,500**	**24,982**	

Source: ITPE Status reports and the CADE 2 Program Management Office Trajectory Model dated November 14, 2019.

Appendix IV

Additional Information on Key Data for Updating the Trajectory Model



IRS **Information Technology**

a) An overview of the process for estimating workload and velocity

For projection accuracy, the team considered two main areas: 1) Capturing all work (in addition to the ALC LOC) required within a single LOC metric and 2) Assessing available data to best estimate velocity per resource in each delivery process step.

	Input	Definition	Impact
Capturing All Work	ALC LOC / Technical Enablers (LOC-E)	**ALC LOC to be converted to Java** **Required Java program components not related to ALC LOC**	**146,000 ALC LOC** to be converted to Java Adds **68,000** ALC equivalent LOC to the 146,000 ALC LOC for cumulative total of 214,000 ALC LOC/LOC-E
	Filing Season Impacts	**Annual changes** required to keep pace with IMF programs	Equivalent of converting ~**10%** of the ALC codebase translated at the time of the filing season
	Defect Correction	**Productivity spent on defects**** identified in AD testing and SAT testing	**15%**** of productivity spent on defect correction throughout the development cycle
Velocity	Skill Levels, Output Rates, Resource Advancement	Included **all resources** and their estimated contributions to output given their roles and starting skill levels **Skill level of resources** and expected **output rate** per process step; **Expected time to advance** to the next level and corresponding higher output rate	**Refined expected output** by estimating velocity by individual and per development process step; Default setting Beginner to Intermediate in **3 PIs**, to Advanced in an additional **4 PIs.** Set **maximum achievable skill level at Intermediate** for **~30%** of resources Applied a **5%** turnover rate

Actuals were used where available, and inputs were validated with AD

**Defect correction rate based on observed historical rate and industry standard comparison

Source: Provided by ITPE Project Management on January 15, 2020. AD = Applications Development; PI = Product Increment; SAT = Systems Acceptance Testing.

Appendix V

Management's Response to the Draft Report

DEPARTMENT OF THE TREASURY
INTERNAL REVENUE SERVICE
WASHINGTON, DC 20224

CHIEF INFORMATION OFFICER

September 1, 2020

MEMORANDUM FOR DEPUTY INSPECTOR GENERAL FOR AUDIT

FROM: Nancy A. Sieger
Acting, Chief Information Officer

Nancy A. Sieger
Digitally signed by Nancy A. Sieger
Date: 2020.08.31 07:29:28 -04'00'

SUBJECT: Draft Audit Report – Customer Account Data Engine 2 Program's Individual Tax Processing Engine Project (Audit # 202020015)

Thank you for the opportunity to review your draft audit report and the opportunities to discuss this Project with TIGTA's Audit Team.

The Customer Account Data Engine 2 (CADE 2) Program's Individual Tax Processing Engine Project (ITPE) is a key initiative within the IRS's Modernization efforts. We appreciate the time TIGTA's Audit Team took to independently evaluate and validate the IRS's Scenario Based approach to the ITPE project.

Further, as the report points out, we remain committed to self-identifying and mitigating potential issues as we continue progressing through this project. In spite of the unprecedented challenges this year, the ITPE project has continued to meet its objectives based in large part to a solid approach and the ability to adapt to the resource constraints as a result of the pandemic.

Finally, we want to acknowledge the TIGTA Audit Team's support of the project in identifying some inconsistencies between coding format and our Internal Revenue Manual (IRM) processes. While we concur that these discrepancies don't present major issues, it is our intent to bring the coding in line with the IRM recommendations.

The IRS remains committed to continuing to build on the progress made to date in order to complete the modernization of the core components of the of the Individual Master File system. The continued support, assistance, and guidance your team provides is very valuable to us in this regard. If you have any questions, please contact me at (202) 317-5000 or a member of your staff may contact Darrell White at (512) 358-6671.

Appendix VI

Glossary of Terms

Term	Definition
Building Block	A grouping of ALC LOC with common functionality.
Data Access Layer	Code or part of a software application that connects directly to a database. It bridges the gap between the application and the database.
Data-Centric	Refers to a focus on the specific data relevant to a given task.
Fiscal Year	Any yearly accounting period, regardless of its relationship to a calendar year. The Federal Government's fiscal year begins on October 1 and ends on September 30.
Individual Master File	The IRS database that maintains transactions or records of individual tax accounts.
Java Runtime Environment	A software package that contains what is required to run a Java program.
Legacy	In the context of computing, it refers to outdated computer systems, programming languages, or application software that are used instead of more modern alternatives.
Logic Harvesting	Analyzing IMF Assembly Code to understand and document the business logic and structure.
Refactoring	The process of clarifying and simplifying the design of existing code, without changing its behavior.
Relational Database	A collection of data items organized as a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
Scenario	Defines an end to end set of building blocks that implement a business result.
Self-Modified Code	Code that alters its own instructions while it is executing.
Target State Architecture	Provides capabilities that will allow direct visibility and access to taxpayer account detail on a near–real-time basis and furthers the overarching effort to retire the IMF.
Technical Enabler	Required Java program components not related to ALC LOC.
Technical Framework	Implements a modern layered architecture which provides the foundational software, data access, tools and common code needed to implement features from scenarios.
Technical Rules Language	A script/procedure language specifically designed to capture ALC constructs and provide a separation between ALC data and program flows, and to provide limited Java functions and class definitions to facilitate translation.

Term	Definition
Trajectory Model	Captures the progress on the LOC and framework that need to be completed and projects future conversion velocity based on factors that affect development.
Transition State	An intermediary state for the CADE 2 system, delivering a set of functionality.
Velocity	Measurement of how much work can be completed in each product increment iteration.
Velocity Confidence Milestone	A goal established for converting ALC LOC to Java during a product increment.

Appendix VII

Abbreviations

ALC	Assembly Language Code
ATT	Auto-Translator Tool
CADE	Customer Account Data Engine
IMF	Individual Master File
IRM	Internal Revenue Manual
IRS	Internal Revenue Service
ITPE	Individual Tax Processing Engine
LOC	Lines of Code