Tackling the Complexities of

Mainframe to Cloud Migrations

The Department of Housing and Urban Development (HUD)

Overview

Over the past few decades, a growing challenge within the government space has been the continued use of mainframes that ran decades-old applications, some of which had been written in customized languages that fit the agency's mission. One such federal agency that faced this challenge was The Department of Housing and Urban Development (HUD).

The Technology Modernization Fund (TMF), authorized in 2017 and disbursed by the General Services Administration (GSA), provided HUD with the opportunity and resources to modernize many of its critical applications that ran on a mainframe.

The Challenge

To get allocated funds from GSA's TMF, HUD needed a proven contracting team to deliver a feasible solution for modernizing its critical legacy applications. These applications, which ran on the same Unisys mainframe and were written in COBOL code, included:

- **The Computerized Home Underwriting Management System (CHUMS)** - Supports Home Ownership Centers (HOCs) staff in the processing of single-family mortgage insurance applications, from initial receipt through endorsement. This system conducts thousands of transactions with external resources daily.
- **Credit Alert Verification Reporting System (CAIVRS)** - Used by multiple federal agencies to determine if a loan applicant has outstanding federal debt that is in default or foreclosure status.
- **The Line of Credit Control System (LOCCS)** - A disbursement and cash management system that distributes over $28 billion annually for grants and subsidies.

GovCIO's Solution

Team GovCIO worked extensively to establish a holistic mainframe migration process for federal customers that improves organizational performance while minimizing risk.

After identifying HUD's need, GovCIO (formerly Salient CRGT) partnered with The Software Revolution, Inc. (TSRI), a leading provider of automated application modernization and refactoring, to proactively develop a working prototype for HUD that demonstrated the efficacy and viability of this process. HUD used this prototype to support its application for TMF funding. Once granted the funding, HUD awarded a sole-source contract to Team GovCIO for this effort.

Using our nine-step process, GovCIO and TSRI rapidly migrated HUD from their COBOL-based mainframes to a modern Azure-based cloud service, all while preventing negative impacts to their mission, day-to-day operations, and security.

9 Steps from COBOL to Cloud

1. Define Project Goals
2. Inventory System Components
3. Identify Unique Technical Challenges
4. Finalize Project Approach and Plan
5. Create Test Strategy
6. Migrate Data Stores
7. Execute Automated Code Conversion
8. Refactor Legacy Code
9. Build Externals

The following paragraphs elaborate how Team GovCIO specifically used our process to successfully migrate the applications to Azure.

Define Project Goals

HUD had five key goals:

- Shutdown the Unisys Mainframe.
- Complete in two years or less.
- Prevent disruption to critical business functions.
- Support ongoing updates and enhancements, including legislative and regulatory changes in parallel.
- Minimize the need for program area decisions.

System Component Inventory

GovCIO performed a comprehensive system inventory, including reviewing all intersecting platforms. Our team analyzed system logs and presented our findings to stakeholders to help them determine which system components could be retired prior to the conversion.

Identify Unique Technical Challenges

The inventory phase identified several technical challenges:

Challenge	Approach Selected
The predominant user interface (UI) resided within ColdFusion and the communication between the COBOL programs and the ColdFusion servers used an obsolete API.	We kept the existing UI and replaced the API with restful web service calls between the ColdFusion tier and the converted COBOL code.
The core system relied heavily on the Unisys hierarchical database (DMS).	We performed an automated conversion of the DMS data structures and associated data access code to relational tables and SQL code. Corresponding unload and load routines were created to move the data.
The use of a NETEX API for bidirectional transaction communication with systems running on the IBM mainframe.	We replaced this API with restful web service calls between the systems.
The web interface to the system is used by 25-30K external users daily. The response time of converted code, database, and platform had to deliver comparable response times.	We incorporated performance testing and measurement into our overall automated test approach from the outset of the project.

Create Testing Strategy

The project goals and approach made developing our testing strategy a straight-forward process. Our team had to make sure the converted system performed exactly as the existing system did, and it needed to do so in the same amount of time or faster. Testing was broken into two primary workstreams- online and batch.

Online Testing **-** Since the project team chose to maintain the existing user interface, we selected a Selenium-based test suite for testing. Prior to converting code, our testers began creating automated test scripts. As the automated refactoring solution iteratively converted the code, the testing team ran the tests against the legacy and target platforms. The test suite automatically compared results to identify any differences. The same test suite and test cases were used for performance testing to simulate production loads and gather performance metrics.

Batch Testing- Like with online testing, the batch programs ran on the legacy and target platforms. Our team compared outputs using binary compare utilities to ensure there were no unexplained differences.

Migrate Data Stores

The systems each had three primary data stores to migrate- a DMS database, an RDMS database, and 40 years' worth of archive data. All data stores were to be migrated to Microsoft SQL Server.

As the RDMS database was the most straight-forward data store, we migrated it to the test environment within the first two weeks of the project. In tandem with the COBOL conversion, our team converted DMS data. These data structures were converted from hierarchical to relational, extract and load programs were created, and then the data was loaded into MS SQL. Migrating the archive data was more tedious as the processing data collected over the past 40 years contained several data discrepancies.

Automated Code Conversion

GovCIO brought on our trusted partner, TSRI, to execute the code conversion. TSRI used its automation-based engine that converts 100% of code ingested. The conversion process produced several artifacts, in addition to the converted code, which would help our project team with troubleshooting and eventually maintaining the converted code. One of the most helpful artifacts TSRI helped us produce was an HTML based blueprint of the system. This blueprint allowed our developers to easily compare the original COBOL code and resultant Java code side-by-side.

Legacy Code Refactoring

Throughout the project, our team identified COBOL code most effective by first modifying in COBOL and then re-convert during the next conversion cycle. To identify this, we used a combination of AI conversion techniques in our testing efforts. In most cases, it was apparent that the COBOL compiler and runtime were much more forgiving than the Java code. The most prevalent, and easiest to illustrate example of this is array index checking. While Java code does strict index boundary checking, COBOL does not. Any code we updated to support the migration was released into production with maintenance releases.

Results

The project was completed in two phases. Phase one comprised the Housing systems, CHUMS and CAIVRS. Phase-two covered the CFO system, LOCCS.

While we planned the phase-one go-live for a three-day weekend to allow for the migration and unforeseen challenges, the system and data migration was complete in just 14 hours. The go-live decision was made on day two. The system went live on schedule with a normal production workload. The phase two migration, planned for a normal two-day weekend, was successfully completed in under 24 hours.

On day one of the system going live, CHUMS supported 25,356 users and 299,715 transactions with only three user problems reported. In the first 30 days after the LOCCS migration from the Unisys to Azure, the system disbursed just over 2.7 billion dollars in program funds without error.

The successful completion of the project solved one of the longest standing and most complex IT challenges facing HUD – moving these mission critical systems off the legacy mainframe platform.

As HUD was among the first to make use of the TMF funding for this type of initiative, HUD's success can also be viewed as a significant endorsement of what agencies can achieve with TMF funding. Team GovCIO proved a rapid, highly accurate migration of mainframe systems to Cloud services is possible, even for critical complex projects such as HUD's.