MODERNSYSTEMS

CASE STUDY

AEROSOFT SYSTEMS

MOVING FROM ACUCOBOL/PERL/C-ISAM TO JAVA WITH MICROSOFT SQL SERVER, WINDOWS SERVER AND HYPER-V

Introduction

AeroSoft Systems, headquartered in Mississauga, Ontario, Canada since 1997, maintains a unique position in the aircraft maintenance management software industry. The company's flagship software, DigiDOC™ (CMS), WebPMI™ and DigiMAINT™, work in conjunction to ensure strict adherence to the OEM aircraft requirements for regulatory compliance and safe operations. A market leader, AeroSoft Systems delivers 7x24x365 support to over 900 aircraft operated by 30 airlines worldwide.

Project Summary

AeroSoft Systems was a unique case for us. They were not seeking to modernize in the traditional sense. They weren't looking to alter the user interface or customer experience. Their effort wasn't driven by saving mainframe MIPS. They wanted a change that would be transparent to their current installed base and enable operational and business goals behind the scenes.

AeroSoft Systems CEO Thanos Kaponeridis learned about Modern Systems via reference from another customer, SABRE, a global travel technology provider who successfully modernized their software. "Our solution is proven and mature, it's served our market for nearly 20 years," says Kaponeridis. "However, we're always looking for ways to streamline testing, development and support."

Initially, AeroSoft Systems evaluated the time and cost of rewriting its application from COBOL to Java as a project requiring over 10 man years. They also evaluated runtime converters and other conversion tools, but were not convinced of their effectiveness or fit with AeroSoft's long-term strategy.

"We have direct experience with different ways of modernizing, from tools to replatforming," says Kaponeridis. "The Proof of Concepts coming back from such 'auto-conversions' produced code that was unusable and still based in COBOL – it would require 5-7 man years in SQL optimization or learning proprietary OO languages. The Proof of Concept executed by Modern Systems produced maintainable Java, which met our requirements."

Modernizing with Modern Systems enabled AeroSoft Systems to:

- Keep the proven business logic and functionality of their application
- Get new features to the market faster
- Reduce maintenance cycles through better data normalization and uniting language and platforms with other Aerosoft Systems products
- Increase the addressable market for AeroSoft Systems by extending the technology to meet more customer requirements

Modern Systems' automated conversion service was used to modernize over 800 programs from AcuCOBOL to Java and migrate data from C-ISAM to a modern SQL Server environment. Additional source environment details include:

- Over 900k lines of AcuCOBOL code
- 13 PERL programs, over 2600 lines of code
- 228 Javascript, over 48,250 lines of code
- 131 C-ISAM files

Engagement Detail

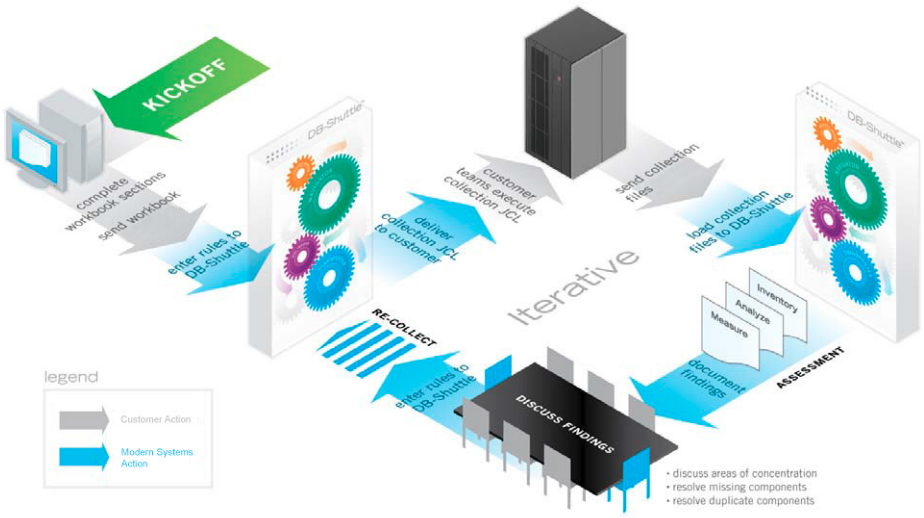
Assessment

All Modern Systems modernization engagements begin by using our DB-Shuttle® automation technology to perform a very detailed, targeted assessment of the entire mainframe application environment resulting in an average mainframe footprint reduction of 50%. It also provides a complete picture of the legacy system, application understanding to reduce maintenance complexity, and provides the ability to fully define a modernization project plan including assignments, responsibilities, time frames, and costs. Additionally, it identifies areas in code and processes that enable the customer to better prepare for regulatory compliance such as Sarbanes-Oxley.

The assessment begins with a question-and-answer session and ends with the presentation of findings and plans highlighting possible modernization options for some or all applications. Iterations of re-collection and re-assessment may be required and performed as additional components are identified and brought into the assessment scope. The DB-Shuttle assessment process is 100% automated, so it is fast, simple, and comprehensive. The process includes:

- Presentation of both technical and business findings
- Definition of the overall customer processing
- Details of missing and duplicate components
- Summarization of areas requiring special attention during conversion, migration, or other modernization
- Documentation of all areas of concern
- Organization of primary findings into an Executive Summary
- Disclosure of recommended actions and options

"This stage lays the foundation for the project's overall success," says Modern Systems Project Manager Pratik Dalal. "Any code or application scenarios that could cause problems in the translation are identified, reviewed and removed if they aren't critical to the application. Once the application is optimized to remove these factors, translation becomes more straightforward and can have a higher success rate."



Application Code and Database Migration

Once the baseline inventory was established and agreed upon, we worked with AeroSoft Systems to prepare for the migration. The early goal for this stage is to understand the application and how it works in order to establish the proper migration path. Order of operations in a legacy migration scenario can be influenced by several elements, ranging from the presence of third-party tools to interfaces with non-mainframe internal or external systems. Once this application mapping process is complete, we analyze the data tier of the legacy environment to generate the data model and layout for the target SQL Server environment.

"With AeroSoft Systems, we focused heavily on performance optimization for the data tier," says Dalal. "Having a relational database put AeroSoft Systems and their customers in a position to generate better reports faster, so we wanted to make sure the system delivered. We worked together with AeroSoft's technical team to ensure program interactions were aligned to business logic and objectives."

Converting a procedural language like COBOL to Java (object oriented) presents several challenges. Our conversion process is designed to deliver on a set of particular standards:

- The converted application works exactly the same as the original application and produce exactly the same results
- The converted application is maintainable and follow the object oriented concepts and paradigms: encapsulation, abstraction, modularization, loose coupling, etc.
- The converted application performs the same or better than the original

To achieve this, we focus on three specific layers of the application:

Data Access Layer
Logic used to handle files and databases is decoupled from the main COBOL structure. To do this, the Data Access Objects (DAO)/Data Transfer Objects (DTO) design pattern is used. DAO ensures communication with external data (files & databases). DTO is used to transport data between main COBOL program and external data. Through the duration, DAO is using DTO and only DTO to communicate with the program.

Business Logic Layer
Each COBOL program is converted to a Java class. This class encapsulates all working storage fields and paragraphs, exposed as a service to the main entry point. Data structures are analyzed and remodeled to meet requirements for Encapsulation, Reuse, Readability and reduced memory footprint for better utilization. Program flow is normalized and GOTOs are removed. Program structure and comments are preserved during conversion, simplifying maintenance.

Presentation Layer
AeroSoft Systems had already invested heavily in the User Interface layer of their application. Therefore, alignment of the back end functionality was the focus of this engagement.

In a more typical scenario, BMS Maps are migrated to equivalent web pages. In all cases, the actual data fed into each new web interface is the same as the data fed into the legacy screen. When the application screen display logic is executed, the resulting HTML web page produced mimics the original 3270 screen as closely as possible. The actual screen layouts are maintained, so users of the application do not have to be retrained. The converted code preserves the look and feel of the original application as the generated JSF defaults to using fixed-width fonts and absolute positioning of fields and text labels. The web application will have the same navigational features as the original one, from PF Keys to Arrow Keys.

Converted Application Testing

Modern Systems works with each customer individually to establish the most efficient testing model. We can run all tests ourselves prior to code release, ship the code to the customer, or do a mix of both. In the case of AeroSoft Systems, we were given a series of baseline functionality tests to run. Passing these tests ensured the converted code worked- and was passed to the AeroSoft Systems team for further testing and refinement.

Modern Systems' standard Quality Assurance and Testing process runs over 20,000 automated tests We go beyond just running and passing batch processes. In the case of AeroSoft Systems, we worked together as a team to identify and document test cases to ensure optimal testing and usage of the target environment.

However, as with any modernization engagement, the AeroSoft Systems project hit a couple of bumps in the road during testing. "We found issues in the performance of the converted application, specifically with screen display of functional elements in a timely fashion," says Modern Systems VP of Delivery John Regan. The system was also taking a disproportionate amount of time to generate regular reports.

Regan and the Modern Systems team worked together with AeroSoft to identify the proper performance targets. "We enhanced the data access layer to take advantage of the relational capabilities in the target data model, defining relationships between tables and retrieving all relevant information in fewer operations," says Regan. "This refinement enabled the system to perform to specifications acceptable by the AeroSoft Systems team and the application was shipped out for further testing."

Converted Application Delivery

Upon resolution of these performance issues, we delivered the converted application code to the AeroSoft Systems team for final testing.

Their model included testing for performance on new hardware and networking platforms with a focus on batch processing and reporting. The AeroSoft Systems team also successfully completed integration tests, connecting the converted application to internal and external systems. Data transfer and synchronization with outside systems was also tested, with the converted application passing and moving to production.

Conclusion

The modernized applications enable AeroSoft Systems to market their software to customers that expect current tools and platforms like Java and SQL Server for their MRO environments. In addition, commonality with DigiDOC development was achieved, allowing better integration for project planning and reporting via Java modules.

As a result, Kaponeridis anticipates more efficient testing and support cycles. "We initially invested heavily in a user-friendly browser front end for our applications. Now the functional pieces of the application are aligned for a more direct data and integration model. COBOL maintenance resources are becoming increasingly hard to find, so this puts us in a position to work with a global community of young talented developers."

About Modern Systems

Modern Systems, Inc. is the leading provider of legacy language and database modernization. Leveraging over 30 years of best-practice domain expertise, Modern Systems works closely with its customers to minimize risk and provide a clear path from legacy platforms like COBOL, Natural/Adabas, CA Gen, and others to modern solutions like SQL Server, DB2, Oracle, Java and more. Modern Systems was chosen by Walmart to modernize the world's largest order system. We've also modernized the world's largest trading platform. Modern Systems has offices in the USA, UK, Italy, Romania, and Israel.

Visit http://modernsystems.com to learn more