Applying a Service-Oriented Architecture to Operational Flight Program Development

Mitch Chan
309 Software Maintenance Group, Hill Air Force Base

This article describes how a Service-Oriented Architecture (SOA) was successfully applied to reuse data and applications previously deployed in single-user, single-computer configurations. The collection of data and applications was transformed into a unified, multiuser, client-server platform through the use of open source and standards-based technologies to minimize development time, maximize interoperability, and facilitate collaboration. The collaboration resulting from the multiuser network capability and shared resources enabled progress towards the Aerospace Basic Quality System Standard (AS9100) goals of lowering cost, meeting schedule, improving quality, and fostering a culture of continuous improvement.

This article describes the experiences of the Hill Air Force Base software engineers in the 309 Software Maintenance Group (SMXG) who develop the Operational Flight Program (OFP) for the Fire Control Computer (FCC) of the Block 30 F-16 aircraft. This article details how the engineers incorporated an SOA to automate its development and evaluation process of weapon coefficients for the delivery of air-to-ground munitions.

The use of open source and standards-based technologies were key success factors for a small team to accomplish the SOA implementation in a do-it-yourself fashion. Using publicly available open source software was instrumental in *minimizing* the implementation time and associated interruptions to the engineers' normal OFP candidate workflow. Choosing technology based on open standards ensured that they were *maximizing* the interoperability between systems.

The engineers were successful in *reusing* data and applications previously deployed in single-user, single-computer configuration and transforming them into a unified multiuser client-server platform that resulted in a building-wide network capability. As a result, non-collocated system engineers, developers, and testers had access to the design and evaluation tools.

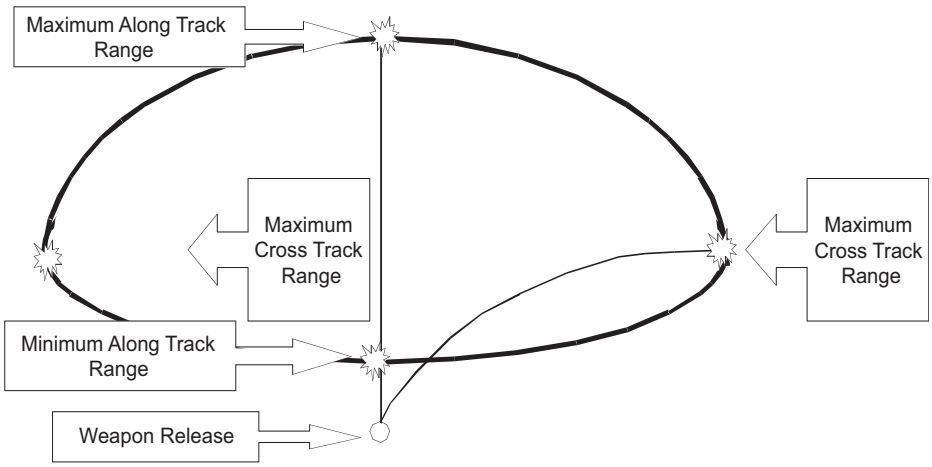
The improved collaboration resulting from the orchestration of network applications and shared resources enabled the engineers to achieve a return on investment (ROI) and progress in meeting its 309 Maintenance Wing's AS9100 objectives of lowering cost, meeting schedule, improving quality, and fostering a culture of continuous improvement.

This article proceeds by first providing a background section that introduces the key terminology. Next, the article describes the former development and evaluation process. Then, the article highlights former inefficiencies and provides the solution which incorporated an SOA as an integration tool to achieve better results.

Background

1. **What is a weapon coefficient?**
   Weapon coefficients are parameters used to calculate weapon trajectories. Examples of weapon coefficients include weight, the cross-sectional area, and other aerodynamic performance factors.
2. **What is a footprint?**
   For every weapon release there is an associated area on the ground representing the possible impact points. This area is referred to as the footprint.
3. **What is a flyout?**
   In particular, there are four points on the boundary of the footprint that are of interest: the Maximum Along Track Range, the Minimum Along Track Range, The Maximum Right Cross Track Range, and the Maximum Left Cross Track Range. These points correspond to the longest forward distance, the shortest forward distance, the furthest right-forward distance, and the furthest left-forward distance, respectively, that the weapon can achieve. These points are referred to as flyouts.
4. **What is truth data?**
   Truth data is the real-world results used to compare calculated trajectories and impact points. Truth data is typically provided by the weapon manufacturer in the form of files or a computer application (weapon model) which produces an output file.

Figure 1: *JDAM Flyouts*



Former Weapon Coefficient Design Process

Figure 1 shows a top view of a Joint Direct Attack Munition (JDAM) release. In the FCC, the four JDAM flyouts are a function of the weapon release point and the weapon coefficients. The four flyouts define a quad-ellipse footprint.

Design Objective: Generate the optimal set of weapon coefficients to determine JDAM flyouts for all release points.

Figure 2 shows the former 3-stage weapon coefficient design process. In

stage 1, the system designer ran the weapon models to generate data for the purpose of comparing the results of the FCC calculations. In stage 2, the system designer generated a set of weapon coefficients intended to meet the design objectives stated above. In order to complete this task conveniently, the system designer used a simulation of the FCC OFP. Finally, in stage 3, the system designer handed off the weapon coefficients to the developer who incorporated the weapon coefficients into the OFP code. Both the developer and tester compared the test stand results against the simulator results used in stage 2.

Inefficiencies of the Former Weapon Coefficient Design Process

From a resource point-of-view, the former design process relied predominantly on the system designer, who performed stage 1 and stage 2. Because this process was *serial*, the developer and the tester were kept waiting until the end of design process. The system designer became the specialist and limited his *bandwidth* (available time and energy) to work on other projects.

The design process did not scale well. For practical purposes, the system designer addressed one weapon at a time, which caused a bottleneck if two or more weapons were involved.

Finally, and most importantly, the design process was not collaborative. The system designer essentially disappeared and the design of the weapon coefficients became a black art. The single-user, single-computer deployment did not encourage developers to invest time in the development tools.

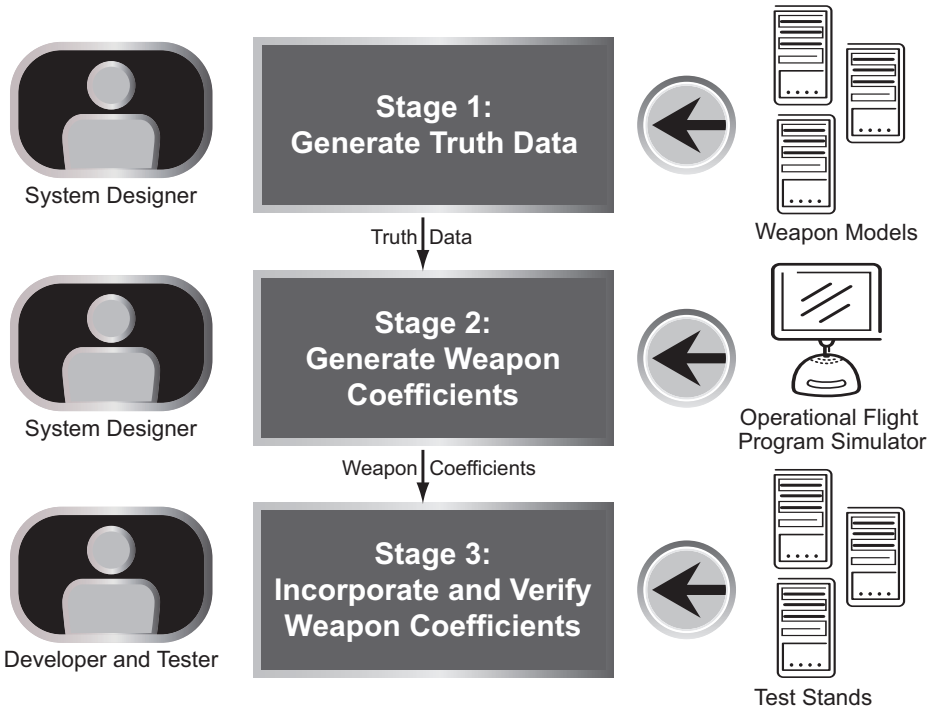


Figure 2: *Former Design Process*

Enter SOA

SOA Purpose

The SOA was designed, implemented, and launched with the intent of making the design tools and data accessible on UNIX desktops across the building network. Servers were a mixture of personal computer and UNIX workstations.

SOA Design

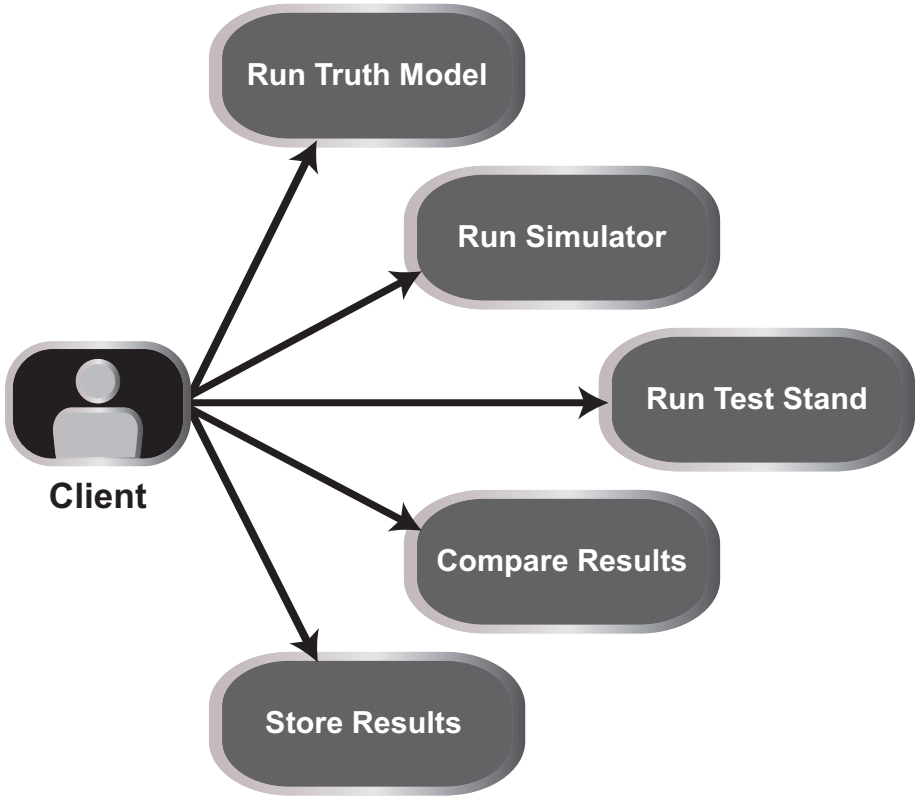
Figures 3 and 4 are the UML diagrams that describe the design of the SOA. Figure 3 shows the five scenarios or use cases. Figure 4 (see page 22) describes the orchestration sequence of the user's client application. Note that the fifth Use Case *Store Results* was added with the intent to replace data that was previously stored in personal directories with a central storage repository.

Figure 4 elaborates some details of the use cases discussed earlier. Note that results are stored in a database versus files.

SOA Implementation

Remote access to the database and the weapon models were provided using open source Web Services. The Web services were constructed using two Java 2 Enterprise Edition (J2EE) Web servers. The weapon models were accessed using a SUN Microsystems Java Web Services Development Pack server. Data Access Objects were exposed across the network using an Apache Tomcat Web server with Axis to provide the Web services connec-

Figure 3: *Use Case*



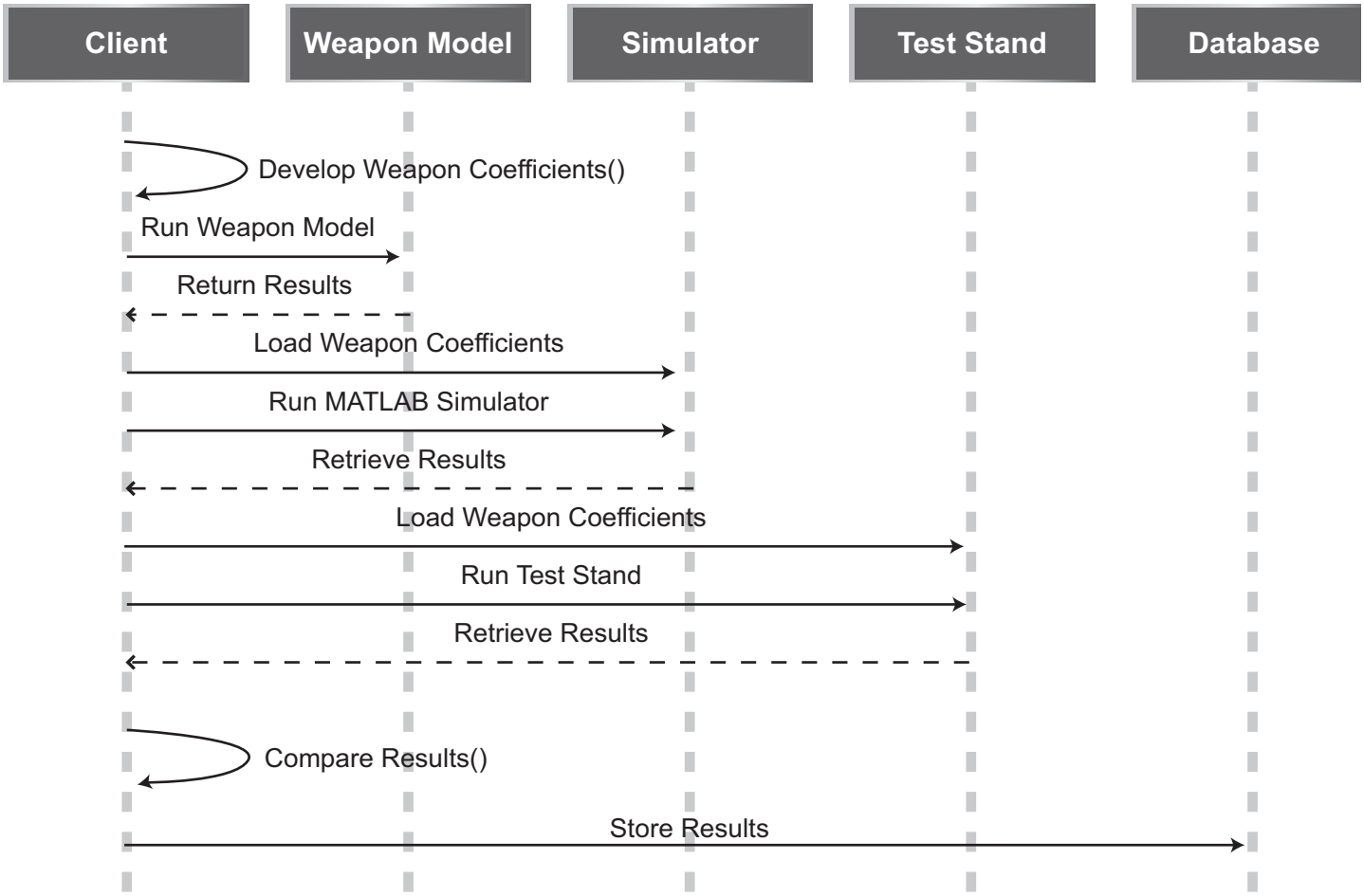
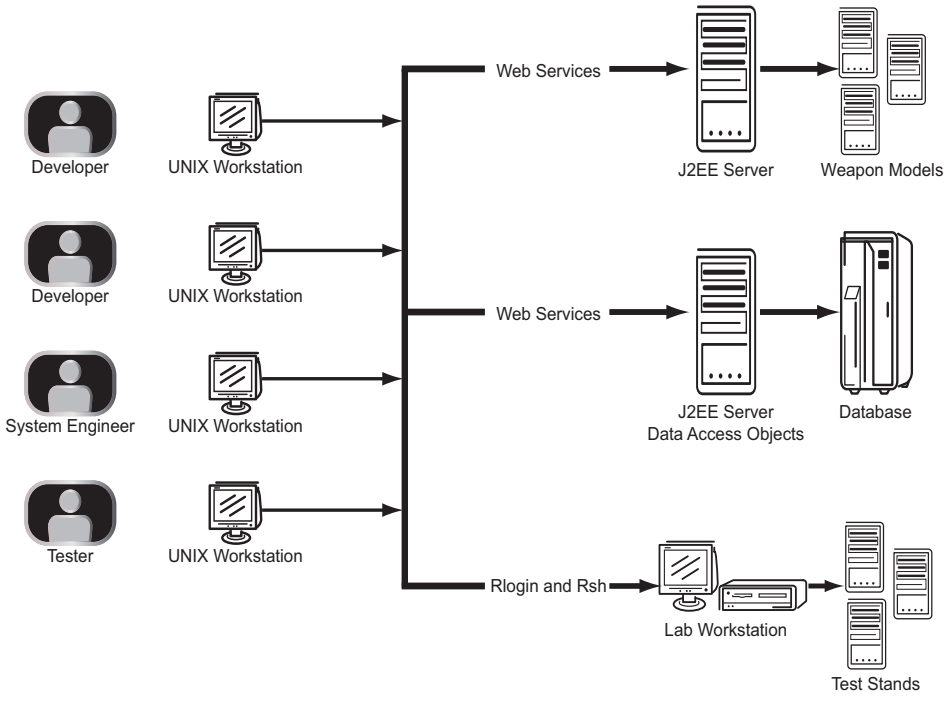


Figure 4: *Sequence Diagram*

tion, and Spring/Hibernate to provide the data access objects, persistence, object-relational mapping, and database connection.

The choice to use open source software greatly accelerated the implementation since a major coding effort was avoided. The majority of the effort was tweaking pre-existing Java source code and editing of eXtensible Markup Language configuration files. The use of standards-based technology on the server-side such as Web services ensured maximum interoperability and tools to create client applications[1].

Figure 5: *Enterprise Diagram*



The lab workstation running the test stands was accessed via UNIX Remote Login and Remote Shell Programming since the client and server workstations both ran UNIX.

Orchestration of the weapon models, the database, and the test stands was performed using Matrix Laboratory (MATLAB) in concert with Perl to create a rich client interface. MATLAB and Perl were chosen as client orchestration tools since they were native to the client Sun workstations.

MATLAB was chosen primarily because of its visualization tools, graphical user interface interfacing tools, math library, and toolboxes. Most of the engineers, especially the recent hires, had plenty of hands on experience with MATLAB. Perl was used for its regular expression capability to handle text input and output. Perl also had the ability with its Simple Object Access Protocol-Lite

(SOAP-Lite) package to create a Web services client simply by setting its service pointer to the URL of the Web Services Definition Language file.

Figure 5 shows the final Enterprise Diagram for the SOA. Note that the simulator does not appear since it runs locally on the client UNIX workstation.

SOA Innovations

20 Minutes to 20 Seconds

One of the first initiatives was to reduce the role of the simulator and develop directly on the test stand. The major hurdle was the length of time required to check whether the coefficient changes were right or wrong. Coefficient changes had to be incorporated in the OFP code. Then the OFP code had to be re-compiled which took more than 20 minutes.

The contractor responsible for the test stands recommended and implemented a workaround to speed up the process. The contractor added a capability to upload weapon coefficient files which took only 20 seconds. The new process of incorporating coefficient changes in a file and uploading the file was simple and fast. With the new process, it became practical to develop directly on the test stand.

Multiple Sessions on the Desktop

Using an older version of MATLAB (Vers. 5 versus Vers. 14) enabled multiple MATLAB client sessions to run on the UNIX desktops. This enabled multiple projects to be open at one time. Perl, in concert with its SOAP-Lite package, facilitated the use of an older version of MATLAB (Vers. 5) which did not have the Web services remote access capability but was less of a central processing unit and memory hog.

ROI and Progressing Towards the AS9100 Objectives

The AS9100 organizational objectives are the following:

1. Decrease cost.
2. Meet or exceed schedule.
3. Improve quality.
4. Develop a culture of continual improvement.

Decreasing Cost

Cost saving was realized both short term and long term. In the short term, the SOA was applied to the current release of the FCC OFP. Expensive end-of-cycle rework costs were avoided by getting the design right the first time. In the long term, cost saving was realized by reduced man-hours (50 percent reduction) resulting from the improved automation and efficiency. A fellow engineer working on a follow-on project noted the following: *I love the tools. I can run the application, check back later, and find all the graphs and results that I need.*

Meeting and Exceeding Schedule Requirements

Removing the keystrokes and mouse clicks reduced the probability of operator error. Comparison between different weapon coefficients sets and improved collaboration lowered the *variability* between developers in designing weapon coefficients. These factors helped to achieve better *predictability* in the execution of the design process.

Improving Quality

The quality of weapon coefficients could be *scored* by *coverage* and *false positives*. *Coverage* is the percentage of the area of truth model footprint covered. *False positives* are the number of impact points lying outside the truth model footprint. Also, the number of release points on the test stand was increased by more than 1,000 percent. The previous method was too time consuming to allow scoring of more than 40 release points per set of weapon coefficients (over 400 coefficients tested at a time). The efficiency of the new method now allows us to score up to 1,100 release points per set of coefficients.

Developing a Culture of Continual Improvement

The SOA enabled a more data-driven design process. As alluded to earlier, the scores *coverage* and *false positives* could be measured from the test run data recorded in the database. Adding more release points over a full range of release conditions increased the statistical significance of these scores. Using the current scores as a feedback mechanism, the developer could further refine the weapon coefficients to produce better scores for the next design iteration.

Lessons Learned

The engineers considered a base-wide network version of the SOA. However, servers and applications on the base-wide network were subject to quarterly time compliance network orders and vulnerable to any collateral damage resulting from patch pushes to update the Microsoft Operating System and Standard Desktop Configuration. In the end, the engineers decided to stay off the base-wide network to avoid the extra computer administration and maintenance.

Conclusion

This article presented SOA from the front line and trenches view of software OFP development at Hill Air Force Base. An SOA applied to automate software development process was introduced. An overview of the inception, elaboration, construction, and transition was covered. Finally, no discussion of an SOA would be complete without evaluating the ROI. ROI was presented in the framework of meeting the 309 Maintenance Wing AS9100 core objectives.◆

Note

1. Deliverables from the Basic Profile Working Group. 2007. WS-I Web Services Interoperability Organization. 6 June 2007 <www.ws-i.org / deliverables/workinggroup.aspx?aspx?wg=basicprofile>.

About the Author

Mitch Chan has served four years at Hill Air Force Base as a Principle Engineer and an Embedded Systems Software Developer for the 309th developing the OFP for the FCC of the Block 30 F-16 aircraft. He specializes in the delivery of air-to-ground weapons and was the primary FCC engineer responsible for adding the capability to deliver the 500-pound JDAM (GBU-38), currently employed by the F16C+ aircraft deployed in Iraq. Chan has a bachelor's degree in mathematics from the University of California at Berkeley, a bachelor's degree in electrical engineering from California State University at Sacramento, a master's degree in electrical engineering from Santa Clara University, and a masters of business administration from the University of California at Davis.

309 SMXG
6137 Wardleigh RD
BLDG 1515 RM 248
Hill AFB, UT 84056
Phone: (801) 586-6756
E-mail: mitch.chan@hill.af.mil