EMBEDDED SYSTEMS CASE STUDY
Fortran – Raytheon – *JANUS* Studio®

Fortran to C++ Raytheon Japanese Patriot Missile System (PRED & BAS3)

CLIENT
Japanese Armed Forces

SOFTWARE
JANUS Studio®

LANGUAGE PAIRING
FORTRAN to C++

COMPLETION TIME
10 months

HISTORY

The Battalion Simulation Support System (BAS3) and its preprocessor (PRED) are simulation programs necessary for testing, integrating and validating the Patriot tactical software. The legacy Fortran code resided on a UNIYS 5600 which was unable to run simulations in real time, did not enable the use of modern development and debugging tools, required specialized hardware for communications, and was expensive to maintain. Raytheon's objective for this modernization was to port to a single board computer by translating and re-hosting the PRED and BAS3 Fortran to C++ code, so that PRED would compile in a Solaris environment and BAS3 would compile in a VxWorks environment. The modernized C++ would conform to Patriot C++ coding standards, standardizing languages with other groups and reducing training costs for new employees.

HIGHLIGHTS

- **Near 100% Automation**
- **Delivered on Schedule**
- **Nearly 50% Reduction in Technical Debt**
- **Military-Grade Secure Modernization Process**

CHALLENGE

Despite familiarity with Patriot C++ coding standards through previous engagements with Raytheon, several technical challenges made the eleven-week timeframe to deliver compiling code an aggressive target. Limiting object-oriented features in the modernized C++ BAS3 and PRED code, converting Fortran arrays into C++ structures, elimination of Fortran GOTO statements, and bit manipulation and packing to enable moving to 64-bit processors were among the Patriot C++ requirements.

RESULTS

TSRI transformed and delivered the C++ code on schedule, with PRED compiling and linking using the Solaris g++ version 3.4.4 compiler, and BAS3 utilizing Wind River g++ compiler along with complete documentation. Together, these modules consisted of nearly 26,000 lines of legacy Fortran source code. Automated refactoring was used to remove dead code and consolidate redundant code, reducing the line count nearly 50% to roughly 13,000 lines of C++ code. Automated modernization delivered an 80% reduction in cost and schedule versus manually re-writing the code, with defects in the transformation of .041%.

© TSRI. All Rights Reserved.