Ada to C++ – Boeing Geospatial Environmental Mapping Software (GEMS)

CLIENT	SOFTWARE	LANGUAGE PAIRING	COMPLETION TIME
Boeing	GEMS	Ada to C++	11 months

HISTORY

In addition to manufacturing military aircraft, Boeing's Space & Defense arm develops and maintains software systems used to operate those machines. The AvionX Geospatial Environmental Mapping Software (GEMS) system supports three aircraft— T-X Trainer, F-15 Eagle, and B-52 Stratofortress — providing them with mission-critical data about the terrain around them, in real time.

HIGHLIGHTS

- **99.5% Automation**
- **Low Technical Risk**
- **100% Unit Testing Success Rate**
- **Met Avionics Compliance Standards**

The GEMS' original codebase, written in Ada, was becoming obsolete and more difficult to maintain each year. In addition, each aircraft had a unique codebase, with nuances that needed to be reflected in the transformed code to capture and maintain all critical functionality. So Boeing worked with TSRI to transform each codebase in turn, each one roughly 350,000 lines of Ada code, to a more modern target language of C++, then merge them into one system to support all 3 aircraft.

TSRI completed the project successfully with their automated code-conversion process, and then automatically refactored the code for compatibility on a modern architecture to meet the stringent requirements set forth by the Department of Defense.

CHALLENGE

The aging compilers that ran Boeing GEMS were written mostly in Ada and were facing obsolescence, as Ada programmers are hard to find given universities no longer teach software development in the Ada language. Knowing that quality hardware costs billions of dollars to develop and build, Boeing wanted to implement a more common toolset throughout its embedded systems software. In this case, converting the entire codebase to C++ would give them a stable, extensible solution.

The certification process for military aircraft for the hardware and software that run GEMS is nothing short of exacting, requiring 80% compliance at minimum to move into production. To achieve this, Boeing invested in the well-known

The GEMS chose TSRI because an automated transformation would satisfy their goals of compliance-ready, speed, and no business disruption.

testing tool, LDRA, to run certification testing on the software. Unfortunately, the application codebase in Ada was not compatible with the LDRA tool.

Boeing knew a manual rewrite of the application would take years, so they explored other solutions and partners. TSRI was ultimately selected to undertake the GEMS modernization project because of their prior success completing modernizations in the military and defense space for Boeing, and they also offered the shortest timeline, lowest risk, and no code freeze. It was the most effective path for Boeing to meet their compliance requirements.

The original GEMS application had been in use for more than 25 years and the core Ada development team had long since retired. In order to understand the application completely, TSRI leveraged their *JANUS* Studio® toolset, generating an *Application Blueprint*® to document all three applications. This showed both teams the structure and functions of the existing code, including design metrics, navigation indices, control flow diagrams, structure charts, data element tables, state machine models, state transition tables, and cause-effect graphs.

Next, a *Transformation Blueprint*® of the target code provided Boeing an opportunity to see what the modernized C++ application would look like alongside the existing Ada applications. In order to achieve this, TSRI's engineers worked iteratively through the automated processing of the code, adjusting their solution model to achieve the required target state.

With these two blueprints in hand, the GEMS team completed a full analysis of the source and target codebases to identify where they could safely refactor the target code to eliminate dead or orphaned code and ensure existing mission-critical functionality would work as needed once the transformation was complete.

Once it was deemed that the old and new systems were functionally equivalent, TSRI's then provided the automated refactoring. Automated refactoring is an iterative process to identify and remove dead code and unreferenced definitions. This consolidates identical data structures and improves system maintainability and code quality without changing the application's core functionality. This refactoring enabled the new C++ code to achieve the necessary pass rate on the strict AvionX compliance testing.

Unit testing ensures that all components of a modernized application work as expected independently before deploying and testing them as a complete system. It was mandatory that each unit be 100% error-free to fulfill Boeing's compliance requirements.

The modernized and refactored GEMS application passed the stringent Avionics compliance testing.

The source application already had perfectly compliant unit-testing, so instead of rewriting these unit tests in C++, TSRI automatically transformed them from the Ada code to C++, and then ran the same tests in both the Ada application and modernized C++ application to prove the systems ran identically. This innovation in converting unit testing has been used in every TSRI project since.

Given that GEMS is a mission-critical system used on multiple Department of Defense aircraft, the compliance requirements for the GEMS system numbered in the hundreds. TSRI had to ensure the transformed code could adhere to strict AvionX standards with a pass rate of at least 80%.

At this point in time, the LDRA tool had never been used on a project or application at the scale of GEMS. TSRI's engineers worked alongside LDRA's engineers through the testing process so GEMS could meet the 80% pass rate required by Boeing and the AvionX Group. As a secondary benefit, the upgrades made by the combined LDRA and TSRI teams enabled Boeing to save time and money on future projects.

When *JANUS* Studio® processes the automated transformation, the goal is to produce like-for-like modernized code that maintains the source code's business logic and behavior. However, forcing C++ to behave like an embedded Ada system, rather than using objected-oriented design standards, does not make the best use of C++'s capabilities. Therefore, the transformation plan for the target application was expanded to take advantage of the modern language.

Another hurdle came from memory allocation: the memory on the source system had been apportioned based on the capability of the Ada code at the time. This was realized during the modernization process, so TSRI's engineers solved the problem by implementing dynamic memory allocation as a part of the system refactoring process.

RESULT

TSRI brought Boeing GEMS to the finish line with an improved system transformed to C++, able to pass the LDRA compliance testing and achieve 100% successful unit-testing, meeting all the government-mandated avionics standards.

TSRI transformed 3 codebases from Ada to C++ on a mission-critical system, delivering an improved and extensible fully compliant system in just under a year. In the end, TSRI was able to achieve the project's goals, and help evolve the LDRA tool for use on large-scale programs like GEMS.