OSD Maintenance Challenge

Abstract Form	
Paper Title:	Dynamic Test Requirement-Code Architecture
Author:	Geoffrey Dolinger
Organization:	556 SMXS/MXDPB
Phone:	405-582-9387
Email:	geoffrey.dolinger@us.af.mil

Abstract: (300-500 words)

Test Program Sets (TPS) are the software, hardware and documentation used to test, diagnose, repair, and certify complex avionic systems. The development of Test Program Sets (TPS) has traditionally been divided into three main efforts: defining the test requirements, writing software, and designing hardware to implement the test requirements on specific Automatic Test Equipment (ATE). Historically, test requirements have been written as a static form of documentation, known as a Test Requirements Document (TRD). Unfortunately, a static TRD results in limitations and repeated efforts throughout the development life-cycle of a TPS. However, a dynamic TRD, using the Standard Generalized Markup Language (SGML) architecture, can improve development efficiency and end-product consistency, reduce limitations and cost, improve maintainability, and simplify creation of automation and reporting tools.

In avionics testing, a Unit Under Test (UUT) can be a single circuit card or an assembly of electronics. For each UUT, a TRD is developed to describe the stimuli, loads, and measurements required to verify UUT functionality and provide failure diagnostics to repair the UUT. The goal of a TRD is to describe the test requirements of a UUT without requiring specific test equipment. Once developed, an avionics TPS consists of software and hardware designed to implement the TRD on specific Automated Test Equipment (ATE). During the life-cycle of a UUT, TPSs are often converted (known as re-hosting) from one ATE to another due to maintainability of an aging ATE system. As a result, considerable time and cost is associated with translating the information in the TRD to the actual code in the TPS required for the specific ATE. Although it is necessary to design a TRD in an ATE-independent manner, there is no reason to completely disconnect the code from the TRD.

The Dynamic Test Requirement-Code Architecture allows a dynamic link between the TRD and the TPS software. A change in the TRD generates an update in the TPS code. The TRD content is created from a set of standardized functions that describe the stimuli, loads, and measurement requirements. These standardized TRD functions are actually software objects with properties that can be accessed by the code that implements the test requirements described by the functions. This dynamic system resulted in several benefits: efficient and accurate code generation, minimized code update time, dynamic warnings and error messages during instrument selection via instrument profiles and automated Test Accuracy Ratio (TAR) calculations, and simplified creation of reporting tools and converters for re-hosting TPSs onto another ATE system. Furthermore, an industry standard TRD can be generated from the dynamic TRD.

The Dynamic Test Requirement-Code Architecture has been in use for over a year and has resulted in an 18% reduction in TRD development time and a cost savings of 14%. More importantly there has been a 36% reduction in hardware design time and a cost savings of 42%. Additionally, the quality and consistency of the TRDs have improved due to the benefits of the new architecture.