FROM: **ROBERT ADAMS**
TITLE: **SOA WHITE PAPER**
EMAIL: **robert.adams@whatifwe.com**
WEB-SITE **www.whatifwe.com**
DATE: **JUNE 27, 2007**

1.0 INTRODUCTION

Robert Adams has had over 46 years of experience with military and exportable-military projects and organizations and has been a programmer/software engineer since 1966.

Over the last 15 years, he has been an independent software engineer dedicated to the economical development of high reliability software using methods supportive of ISO-9001. To this end, he has developed a technology based on a Programmable Code Generator and a Programmable Monte-Carlo Test System

2.0 PROGRAMMABLE CODE GENERATOR

The "Programmable Code Generator (PCG) enables the use of enforceable standard methods and processes for the development of a software product. Specifically, the PCG redefines the programming effort as a formal two-step process:

- The development of a library of "software parts".
- The use of this library in the development of product.

3.0 SERVICE-ORIENTED ARCHITECTURE

3.1 SIMULARITIES WITH THE PROGRAMMABLE CODE GENERATOR

An examination of the white paper "What Is Service-Oriented Architecture" found on www.xml.com has revealed a large similarity between the two methodologies. Enforceable standard methods and processes are involved in both methods. In SOA, the basic two steps are:

- The development of a set of basic software agents.
- The use of these software agents to supply a specific service.

Both methods are strongly supportive of ISO9001.

3.2 AN INTERESTING QUESTION.

At what point due we stop using the formal concepts of SOA and resort to standard programming methods? There will be a point at which we are no longer developing a service agent exclusive with more primitive service Agents. At this point, the Programmable Code Generator enables enforceable standard methods to continue to be used in the development process. There is no need to resort to more traditional coding methods.

4.0 QUALITY ASSURANCE CONSIDERATIONS

4.1 THE ERROR TO BE FEARED.

The error to be feared is the error that is not found in the development process. Such an error can create serious problems for the end-user. Programmers will not discover all of the errors that they make; and the more skilled the programmer, the longer it will take to find and correct the undiscovered error.

There are some programming methods and source code formats that tend to be more resistant to errors. Our standard methods and processes must include such methods; unfortunately, QA inspections cannot guarantee their enforcement.

4.2 "DIVIDE AND CONQUER" TEST PHILOSOPHY

An aggressive test plan is critical to the discovery and prevention of errors. This plan must include not only all basic service agents but also the standard methods and processes that are used in their design. This "divide and conquer" method of testing is not only more effective in discovery and prevention of errors; it is, in the long haul, more economical than conventional proper-operation testing and checkout methods.

4.3 PROGRAMMABLE MONTE-CARLO TEST SYSTEM

The Programmable Monte-Carlo Test System was designed specifically for this testing philosophy. It is not only able to test the functionality of a module or executable, but also the software part at the lowest possible design level. The use of a Monte-Carlo approach insures that the Software Unit Under Test is subjected to a wide range of conditions, not just the ones that make sense.

The Programmable Monte-Carlo Test System is also capable of measuring the strength of a methodology at resisting programming errors. To accomplish this, the PMTS will prepare, compile, link and then execute a test program for each test condition in the sequence. . If an error is caught during the compilation or preparation steps, it will be appropriately recorded and reported.

5.0 CONCLUDING REMARKS

The PCG and PMTS technology is supportive of the SOA architectural methodology. The PCG extends the benefits of enforceable standard methods and processes to all levels of the design. The PMTS maximizes the benefits of an aggressive "divide and conquer" test philosophy Furthermore, QA can be given a more proactive role by involving them in the development of the basic software parts and testing methods.