On-Orbit Software Analysis

ITC-RIF-04-0030

Final Report

July 2004

On-Orbit Software Analysis

Overview

PoC: Dr. Susanne I. Moran, Intrinsyx Technologies Corporation, NASA Ames Research Center

Introduction

The On-Orbit Software Analysis Research Infusion Project was done by Intrinsyx Technologies Corporation (Intrinsyx) at the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC). The Project was a joint collaborative effort between NASA Codes IC and SL, Kestrel Technology (Kestrel), and Intrinsyx.

Problem Statement

In December 2007, NASA intends to launch the next-generation host systems and payloads for deployment on-orbit. To mitigate technical and cost risks, and to ensure fidelity, reliability, and robustness of the on-orbit software, we applied the C Global Surveyor (CGS) Tool, developed by Kestrel for NASA ARC, to the Habitat Holding Rack (HHR) software. The Tool presented an excellent opportunity to evaluate the software, isolate defects prior to launch, and significantly mitigate potential risks.

The primary objectives of the Project were:

- Discovery and verification of software program properties and dependencies
- Detection and isolation of software defects across different versions of software
- Compilation of historical data and technical expertise for future applications

Application of the Technology to the Target Project

We applied the CGS Tool to the Biological Research Project Rack Interface Controller (B-RIC), the command and control component of the HHR. The B-RIC formats telemetry data received from payloads for downloading to the ground, and creates HHR and Payload Health and Status (H&S) data for on-orbit transmission. We chose three B-RIC versions, each containing five modules, for the Tool infusion, with expected results as follows: discovery and verification of software program properties and dependencies; detection and isolation of software defects over different source code versions; measurements of the efficacy of the CGS Tool; and historical data and technical expertise for future applications.

Data Collection and Analysis

We ran the CGS Tool on the B-RIC code on three separate baselined versions and on five logically distinct modules within each of those versions. After each run, we tabulated the data for error classification, tabulation of numbers of files, numbers of functions within files, and tabulation of numbers of actual error within the functions. We then did data interpretation, which consisted of data consolidation to single defect classes, error verification and isolation for severity against the B-RIC code, and cross-correlation between captured errors and established metrics. Finally, we evaluated the efficacy of the CGS Tool, which included correctness of error counts against known errors, length and duration of run-times, and ease of use.

We found and isolated two general types of errors: the first were caused by the test environment or the CGS Tool, which were not considered valid, and, hence, not serious; and the second, which were serious and may impact performance. We also evaluated the efficacy of the Tool.

The environmental and/or Tool configuration errors were:

- The B-RIC software was tested in a stand-alone mode, which caused a number of errors to appear because external hardware or interfaces were not connected.
- There were errors caused by the Tool's inability to "see" such areas as bit fields and un-initialized pointers.
- There were false indicators, caused by the structure of the code, which the Tool could not inventory.

The serious defects in the B-RIC code by order of severity were:

- There is a serious memory leak in one of the modules.
- There are un-initialized arrays of file pointers in the code.
- The run duration for one module indicates potential problems with execution. The mean run time for that module was 14 hours. The comparative run times for the other modules ranged from ½ hour to 2 hours.
- Three of the five modules contain dead code, which impacts code integrity and may limit code functionality and/or cause delays in execution because of potential overload of on-orbit platforms.

The CGS Tool was domain-specific and does not readily transform to various applications. It was also missing a Graphical User Interface (GUI) and had no capability for automatic error classification, which made data isolation and interpretation difficult. To make the Tool more extensible, we recommended modifications.

Summary and Lessons Learned

The primary objectives of the On-Orbit Software Analysis Research Infusion Project were met. The CGS Tool was successfully used to discover and verify program properties and dependencies, and to detect and isolate software defects. The Tool application to the B-RIC software resulted in data gathering that can be used for comparative evaluation of future B-RIC Version releases and other software projects. Application of the CGS Tool also resulted in successfully transferring expertise from the Technology Developers to the Applications Developers, critical for exportability and continuity of the technology to other programs and projects.

The secondary objective, incremental validation of the CGS Tool, was also successfully met. The consequences of applying the Tool in a different-than-usual environment provided lessons learned for the Technology Developers about areas where the Tool may be modified and/or expanded to suit different applications. We continue to interface with the Technology Developers informally to derive Tool modifications, and to explore future uses of the Tool for other collaborative efforts.

Table of Contents

List of Illustrations

1.0` Introduction

The On-Orbit Software Analysis Research Infusion Project was done by Intrinsyx Technologies Corporation (Intrinsyx) at the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC). The Project was a joint collaborative effort between NASA Codes IC and SL, Kestrel Technology (Kestrel), and Intrinsyx.

The On-Orbit Software Analysis Final Report chronicles the Project and documents the findings.

1.1 Problem Statement

In December 2007, NASA intends to launch the next-generation host systems and payloads for deployment on-orbit. To mitigate technical and cost risks, and to ensure fidelity, reliability, and robustness of the on-orbit software, we applied the C Global Surveyor (CGS) Tool, developed by Kestrel for NASA ARC, to the Habitat Holding Rack (HHR) software. The Tool presented an excellent opportunity to evaluate the software, isolate defects prior to launch, and significantly mitigate potential risks.

The primary objectives of the Project were:

- Discovery and verification of software program properties and dependencies
- Detection and isolation of software defects across different versions of software
- Compilation of historical data and technical expertise for future applications

A secondary objective was:

- Incremental validation of the CGS Tool

1.2 System Background

The HHR is the central part of the biological and scientific experiments to be conducted on-orbit and on the ground for missions following the December 2007 launch.

The Biological Research Project Rack Interface Controller (B-RIC) and Centrifuge Interface Controller (C-RIC) are the command and control component of the HHR. The B-RIC formats telemetry data received from payloads for download to the ground, and creates HHR and Payload Health and Status (H&S) data for on-orbit transmission. The C-RIC is a derivative of the B-RIC with audio capability and additional payload channels. As noted below, only the B-RIC was analyzed for this Project.

The B-RIC and C-RIC contain circa 50,034 Source Lines of Code (SLOCs) each in the C programming language. The software is in flux with version releases done on an incremental basis. There are known defects in the software, documented in Software Problem Reports (SPRs) that are tracked for resolution.

1.3 Project Scope and Delimitation

The initial intent of the proposed Project was to apply the CGS Tool to the B-RIC and C-RIC software. However, the C-RIC software was not ready within the time frame of the Project.

As noted above, the C-RIC is derived from the B-RIC, and the differences between the C-RIC and the latest B-RIC Version are 20 SLOCs. The decision was made to use three Versions of the B-RIC source code in place of the C-RIC. Doing so provided an equal representation of the data and errors, and did not impact the goals of the Project.

The expected benefits include the following:

- Software program verification to isolate defect classes, such as array out-of-bounds, and un-initialized variables/pointers
- Software program analyses for precision and scalability, including false positives
- Data consolidation for comparative analyses
- Benchmarking of the CGS Tool for purposes of efficacy

1.4 Approach

The approach for conducting data gathering and analysis with the CGS Tool consisted of three phases, delineated below.

- Phase I – Pre Data Gathering. This Phase began with a formal Kick-off Meeting, followed by training from the Kestrel staff, the CGS Technology Developers to the Intrinsyx staff, the Applications Developers in the use of the Tool. The B-RIC source code was prepared for installation on a Laptop, and configured with Unix and the required CGS applications tools. Concurrently, the proposed approach and metrics were solidified. The Phase started on 4 March, and ended on 19 March.
- Phase II – Data Gathering. This Phase started with the Principal Investigator (PI) and the Application Developers configuring the required fields for data gathering and metrics on the Laptop that housed the Tool and the B-RIC code. Versions 5, 6, and 7 of that code were segmented into the five logical modules, and run with the CGS Tool. The raw data were gathered and consolidated for subsequent analysis. The Phase started on 22 March, and ended on 30 April.
- Phase III – Post Data Gathering Analysis. Phase III consisted of data consolidation and preliminary analyses of all code runs, preparation of a Draft Report, thorough analyses of the defects discovered with the CGS Tool, and preparation and delivery of the Final Report. The Phase started on 3 May, and ended on 1 July, slightly ahead of the project 12 July end of the Project.

2.0 Methodology

The methodology for isolating defects entailed running the B-RIC code Versions 5, 6, and 7 with the CGS tool separately for each Version, and for each of the five modules in the Versions. Those modules are:

- Video Digitalization Compression Card (VDCC). The software module for the VDCC provides the main entry points for all modules running on the Card. The main module initializes message queues, spawns all tasks, and monitors H&S on the Card. It contains 4,027 SLOCs.
- High Rate Link Card (HRLC). The HRLC software module is the main entry point for all other modules running on the Card. Like the VDCC, it initializes message queues, spawns tasks, and monitors H&S. It contains 16,168 SLOCs.
- Serial Card 1553 (SC1553). The software module on this Card is the 1553 communications link to the HHR. It contains 9,006 SLOCs.
- Serial Card (SERC). The SERC software module is the serial communications link to the HHR. It contains 874 SLOCs.
- Main Controller Card (MCC). This software module is the main controller for the HHR. It contains 19,959 SLOCs.

Once the code runs were complete, the first steps consisted of data tabulation for each Version and module, in the following manner:

- Error classification
- Tabulation of number of files
- Tabulation of number of functions within files
- Tabulation of numbers of actual errors (Reds) within functions

The second steps in the methodology consisted of data interpretation, including:

- Data consolidation to single defect classes
- Error verification and isolation for severity against the B-RIC code
- Cross-correlation between captured errors and established metrics

The third and final step is an evaluation of the efficacy of the CGS Tool, including:

- Correctness of error count against known errors
- Length and duration of run-times
- Ease of use

3.0 Data Analyses and Findings

3.1 Data Tabulation

The CGS Tool provides a cumulative count by color code, as follows:

➢ G (Green):	No errors or defects detected
➢ O (Orange):	Potential errors, not enumerated
➢ R (Red):	Actual errors
➢ U (Unreachable)	Unreachable operations, dead code

Tables 3.1-1 through 3.1-3 depict the raw data count for each B-RIC Version.

Table 3.1-1 Number of Error Counts per Module for Version 5

Version 5	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
Green	1,267	1,009	1,498	329	1,273
Orange	4,895	1,896	1,969	1,073	2,528
Red	225	232	536	105	303
Unreachable	0	3	2	2	0

Table 3.1-2 Number of Error Counts per Module for Version 6

Version 6	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
Green	1,267	1,031	1,499	330	1,277
Orange	4,895	1,920	1,973	1,078	2,535
Red	225	233	536	105	304
Unreachable	0	4	3	3	0

Table 3.1-3 Number of Error Counts per Module for Version 7

Version 7	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
Green	1,619	1,252	1,755	333	1,311
Orange	4,553	1,929	1,907	1,097	2,600
Red	314	13	372	108	312
Unreachable	0	4	3	3	0

Table 3.1-4 depicts the number of files in each module by Version. There is no statistical significance to the number of files, but there are correlations between that number and the enumerated functions and errors. There may also be a correlation between file number and efficacy of the CGS Tool.

Table 3.1-4 Total Number of Files per Version/Module

Version	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
5.0	19	27	12	8	21
6.0	19	26	12	9	21
7.0	25	4	25	9	23

The CGS Tool detects errors within files for each function by line number. Because the functions may appear on a number of lines within the code, we counted the errors for all instances as one. For example, in the SC1553 module, the file BC.C contains a function Init-BC-Message-Link, which shows up on 45 different lines of code. The error is counted as one "class" with multiple instances. Table 3.1-5 contains the total function/error count.

Table 3.1-5 Total Number of Functions/Errors per Version/Module

Version	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
5.0	62	58	42	23	42
6.0	61	60	43	23	43
7.0	57	4	58	21	41

3.2 Data Interpretation

Data interpretation consisted of consolidation into error classes by function for each module, error verification and isolation, which entailed verifying defects discovered with the CGS Tool against known, documented defects, establishing severity of defects, and cross-correlation of defects to metrics.

Analyses of all defects show that error classes are consistent across the three B-RIC Versions; hence consolidation was done by isolating error commonality for all Versions by module.

3.2.1 Data Consolidation

After data consolidation and isolation of those errors that are caused by external variables, there are four error classes remaining that were the consequence of the test environment and/or the capability of the CGS Tool. They are listed below.

- Hardware Pointers, which are largely a consequence of the test environment. The isolation of the source code verification with the CGS Tool without attachments to external devices and/or interfaces shows up as repeated errors for all instances where there is no connectivity. For example, in the SERC Module, the Payload Manager Function calls the Payload Table. Because that Function depends on an external link for execution, the CGS Tool tagged all instances of calls as Red errors.
- Bit Fields, which is a consequence of the CGS Tool functionality. The Tool cannot "see" bit fields.
- Un-initialized Pointers, which the Tool cannot "see".
- False Indications, which are caused by the code inclusion of C files into another C file. The Tool does not keep a history of the inclusion.

Because of the nature of the errors in the classes, listed in paragraph 3.2.1, we would expect that there is no impact or severity to the integrity of the B-RIC code from those errors. The errors are caused by the Tool and/or the test environment.

3.2.2 Error Verification and Isolation

The code defects are listed below by order of severity.

- There is a serious memory leak in the HRLC module. We suspected a problem, which the CGS Tool did not register. We subsequently applied another tool to the module and found the leak.
- There are un-initialized arrays of File Pointers in the code.
- The run duration for the VDCC module indicates potential problems with execution. As illustrated in Table 3.3.2-1, the mean run time for that module was 14 hours. The comparative run times for the other modules ranged from ½ hour to 2 hours.
- As illustrated in Tables 3.1-1 through 3.1.3, three of the five modules contain dead code, which impacts code integrity. The HRLC, SC1553, and SERC have several instances of dead code, which may limit code functionality and/or cause delays in execution because of potential overload of on-orbit platforms.

3.2.3 Cross-correlation to Established Metrics

Cross-correlation to metrics consisted of the following:

- Dead code
- Misuse (arrays out-of-bounds)
- Initialization (no values, incorrect values)
- Null pointer de-referencing

Data analyses of the code runs show instances of dead code, as discussed above. The CGS Tool does not distinguish misuse, initialization, and null pointer de-referencing.

3.3 CGS Tool Evaluation

3.3.1 Error Correctness

In addition to total checking time, which is enumerated in paragraph 3.3.2 below, other metrics against which the CGS Tool was applied include:

- Numbers of false positives
- Percentage correct/incorrect

The numbers of false positives, indicated as Orange, for all modules and Versions were very high. Tables 3.1-1 through 3.1-3 illustrate.

The percentage correct/incorrect findings are derived with the following formula:

$$\frac{G+R+U}{G+R+O+U}$$

Raw tabulation across Versions and modules show mean correctness of 65%. When errors that are caused by the CGS Tool not "seeing" functions are removed from that equation, the mean is 90%.

3.3.2 Run-time Efficacy

Table 3.3.2-1 illustrates the mean duration of the runs for all the B-RIC modules.

Table 3.3.2-1 CGS Tool Run-time

Module	*VDCC*	*HRLC*	*SC1553*	*SERC*	*MCC*
Mean Time in Hours	14	1/2	1	1/2	2

As discussed above, the VDCC mean run-time of 14 hours is disproportionate to the other modules, including the MCC, which is much larger in SLOCs.

3.3.3 Usability

The CGS Tool has been used primarily on very large flight software applications at the Jet Propulsion Laboratory (JPL) and at MSFC. The application of the Tool at NASA ARC on the B-RIC code diverged from the norm because of the smaller scale and different findings. For the Technology Developers, the results of the B-RIC research infusion are lessons learned about areas that the Tool does not "see". Those lessons learned have resulted in evaluations of potential enhancements to the CGS Tool.

Application of the CGS Tool to the HHR B-RIC code resulted in finding defects that were not previously demonstrated in test and verification of the B-RIC. The most significant of those is the memory leak in the HRLC module. Those findings warrant further investigation.

For the Technology Developers, the divergence from the norm of using the CGS Tool on the B-RIC has resulted in evaluating the Tool's efficacy across different applications, and locating potential areas where the Tool may warrant modification. According to the Technology Developers, the usability of the Tool would be improved if it computed information differently to make it more readable.

For the Application Developers, training in, and use of, the CGS Tool has resulted in knowledge that can be transferred to other mission-critical software development efforts for NASA.

4.0 Summary

The primary objectives of the On-Orbit Software Analysis Research Infusion Project were met. The CGS Tool was successfully used to discover and verify program properties and dependencies, and to detect and isolate software defects. The Tool application to the B-RIC software resulted in data gathering that can be used for comparative evaluation of future B-RIC Version releases and other software projects. Application of the CGS Tool also resulted in successfully transferring expertise from the Technology Developers to the Applications Developers, critical for exportability and continuity of the technology to other programs and projects.

The secondary objective, incremental validation of the CGS Tool, was also successfully met. The consequences of applying the Tool in a different-than-usual environment provided lessons learned for the Technology Developers about areas where the Tool may be modified and/or expanded to suit different applications.

The most significant findings are highlighted below.

- ***There are serious defects in the B-RIC code that may impact mission-critical performance and operations.***
- ***The B-RIC code structure is non-standard, which made reading outputs of the CGS Tool difficult.***
- ***Application of the CGS Tool to a non-flight application provided good insight for the Technology Developers into desired and/or required Tool modifications for different software applications.***
- ***Application of the CGS Tool provided critical transfer of technology application from the Technology Developers to the Applications Developers, establishing a good bridge for future technology infusion.***

5.0 Actions and Recommendations

5.1 Actions

Application of the CGS Tool to the B-RIC code yielded significant results by identifying defects, not previously documented. Although there are limitations to the Tool, we are confident that those limitations did not detract from true findings, primarily because we did a substantial amount of "manual" extraction from the data to verify results.

The defects are documented and will be tracked and monitored to closure via the Habitat Holding Rack Open Software Problem Report Database, and the Verification Closure Report (VCR) Database. There are planned future version releases, which can be re-tested using the same methodology as the one applied in this Project to verify that defects are corrected prior to launch. Re-test with the existing CGS Tool will yield correction/no-correction of the errors found in our analyses; if the CGS Tool is modified prior to re-test, there may be additional errors.

5.2 Recommendations

As noted above, application of the CGS Tool to the B-RIC code diverged from previous applications. The divergence showed limited extensibility of the Tool. The CGS Tool is domain-specific, using ISO C; the B-RIC uses GCC Extension. The B-RIC code is written in complex Bit-Structure, which the CGS Tool does not "see". There is currently no Graphical User Interface (GUI) and no capability for automatic error classification, which makes data isolation and interpretation difficult.

To maximize the efficacy of the CGS Tool for future on-orbit and/or ground systems' application, we recommend the following:

- ➢ Add a GUI to enhance user-friendliness and readability
- ➢ Modify the Tool to include specific error classes
- ➢ Include automated error classification
- ➢ Modify the Tool to be structurally independent
- ➢ Add automatic isolation of dead code

6.0 Acronyms

ARC	Ames Research Center
BRP	Biological Research Project
B-RIC	BRP-Rack Interface Controller
CGS	C Global Surveyor
C-RIC	Centrifuge-Rack Interface Controller
GUI	Graphical User Interface
HHR	Habitat Holding Rack
HRLC	High Rate Link Card
H&S	Health and Status
JPL	Jet Propulsion Laboratory
MCC	Main Controller Module
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
PI	Principal Investigator
RA	Research Assistant
SC1553	Serial Card 1553
SERC	Serial Card
SLOC	Source Line of Code
SPR	Software Problem Report
VCR	Verification Closure Report
VDCC	Video Digitalization Compression Card