

# Independent Software Quality Assessment (ISQA)

**Coleen H. Coughlin**

**CERDEC SED**

**732-427-6023**

**Coleen.Coughlin@us.army.mil**

**Date: October 16, 2007**



# Outline

- **Independent Software Quality Assessment Defined**
- **System Profiles**
- **ISQA Deliverables**
- **The Value of ISQA**
- **Lessons Learned**

# The ISQA Fort Monmouth Initiative



Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance

**CERDEC**

**CECOM**

**PEO C3T**



**GARRISON**

**PEO EIS**

**PEO IEW&S**



# Independent SQA Defined

1. ISQA is a set of services (methods, tools, techniques) to assess the quality attributes of software products at various stages in their development.
2. Quality attribute examples include:
  - Adaptability
  - Performance
  - Safety
  - Functionality
  - Portability
  - Security
  - Maintainability
  - Reliability
  - Usability
  - Interoperability
  - Robustness
3. The purpose of ISQA is to provide an objective perspective on the “goodness” of the software, and to provide a confidence level for the software product.



# ISQA Service Capabilities

- ★ Static Analysis Service
- ★ Run-Time Analysis Service
- ★ Can Be Customized

## Quality Assessment & Audit ★★

This service represents a broad and general analysis of software quality indicators and attributes including but not limited to Architecture Review, Development Process Review, Configuration Management Maturity Review, Inspection Attributes, Structural Metrics, Code Completeness, Complexity, Portability & Security, Architecture & Design. In addition, a Statistical Defect Analysis of Instances Identified From Defect Categories is Performed.

### Error Detection ★★

**A 100% Defect Analysis** of instances identified from defect categories identifying failures and additional problems that have escaped the code inspection & testing processes. Issues are categorized by severity of impact.

### Memory Leak Detection ★

This service identifies overall machine memory use degradation, crashes and running out of resources caused by memory leaks and data corruption errors.

### 2<sup>nd</sup> Order Analysis ★★

Custom analysis techniques & methods to identify categories of software errors difficult, complex and high value defects to find which are outside of the scope that standard software quality automation products can identify.

### Performance Tuning ★

This service identifies modifications in the software application which improve the performance and response times. Recommendations are made based upon high value modifications with minimal architectural impacts.

### Software Threat Detection ★★

This process analyzes software source or binary code for vulnerabilities whether accidental or intentional and for potential interaction with other software and hardware products in the execution environment.

### Test Coverage Analysis ★

This service maps the customer's current inventory of tests to the percentage of the software system's source code covered identifying redundancy and insufficient testing.

### Unit Inline Static & RunTime ★★

These processes operate in shorter timeframes analyzing units of code (CSC) leveraging a combination of static and runtime technologies designed to impact engineer productivity and quality early in the development lifecycle

### Darwin Testing ★

Next generation testing engine technology that automatically extends test coverage reusing legacy tests and creating new test architectures for random execution strategies to model customer usage scenarios.

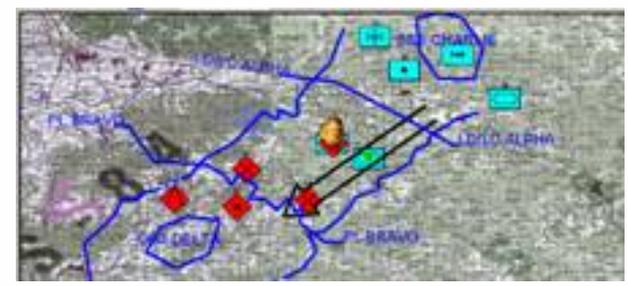
# ISQA Application Examples.....



1. Joint Tactical Terminal
2. Guardrail
3. Common Ground Station
4. Agile Commander
5. Maneuver Control System (MCS) Heavy
6. MCS Light
7. Electronic Key Management System
8. Soldier Radio Waveform
9. Global Command and Control Systems - Army
10. Advanced Threat Infrared Countermeasure/Common Missile Warning System
11. Joint Tactical Radio System
12. Distributed Common Ground Systems Army
13. Coalition Joint Spectrum Management and Planning Tool
14. Command & Control of Robotic Entities



*Past Effort Involving ISQA:*  
*"The error detection service was valuable and worth the cost to the MCS program. The service found about 20 errors that could have resulted in significant issues for the program ranging from software crashes to unexpected system behavior."*  
*-- Carol Wortman, PM MCS-Light*



# ISQA Customer Profiles

## Customer Profile

## Business Problems

## Solutions

1

“Code Red” Project



Customer / Congressional oversight / Crisis situation / No time to ramp / Need immediate solution

Code inspection services do not distract team / Services are turn-key & immediate / Provide management visibility into areas of risk

2

Rapid Prototyping Project: Team incentive is creative design & speed...not quality



Re-use prototype as foundation for real product / Prototype is in field use and has quality issues

Provide management visibility into areas of risk / Provide quality without sacrificing speed & innovation

3

Legacy Systems: Maintenance cost reduction, fresh coat of paint, re-use in new architecture



Need to cut funding from current system to fund new system / Extend life of system due to budgetary constraints / System’s tolerance unknown in new architecture

Reduce maintenance costs by identifying potential systemic quality problems driving costs / Visibility into “as-built” architecture / Darwin testing can mitigate risk of use in new architecture

4

Systems Integrator (SI) Syndrome



SI lacks visibility into subcontractor code quality / Presents significant risk to program

SI establishes external S/W quality perimeter to independently audit all subcontractor S/W / Customize service to SI’s coding standards

5

Software has to conform to specific industry and/or customer standards i.e. DO-178(b)



Project lacks domain expertise, time and resources to audit system for standards compliance

Services are provided incrementally and early in the lifecycle to reduce non-domain specific code inspection activities

6

Project Undergoing Schedule Compression



Need to increase engineering productivity to meet milestone & spiral deliveries

Services can be customized to audit source code for compliance to standards. In addition, as stated above, can also be leveraged on external code

# System Examples

# System 1 Profile

## WHAT WE DID:

- Performed independent source code quality assessment/audit (portability analysis, defensive programming analysis, defect analysis)
- Assessed five software components

## WHAT WE OBSERVED:

- Duplicate/Copied code
- Missing source code
- Undocumented code
- Programming style
- Some non-compliance with architecture standards
- Proprietary code
- Specialized processor & assembler code

## THE RESULT:

- Identified problems early prior to delivery to the Government
- Improved developer's coding practices/processes
- Increased quality of subsequent software versions
- Helped Government understand what was required for a clean build of software
- Established/captured consistent baseline measures across software systems
- Ensured programmatic standards (i.e. measured, tracked, compared across vendors & systems; trended and catalogued).
- Contributed to assessment of software-related Technology Readiness Levels



# System 2 Profile

## WHAT WE DID:

- Performed quality assessment/audit & error detection forensic services

## WHAT WE OBSERVED:

### Positives

- A number of categories of defects are defect free, due to:
  - Use of C++ vs. C
  - Use of some good habits for code quality.

### Improvement Opportunities

- Software code inspection process improvements needed especially for:
  - Exception handling
  - Null pointer checking
  - Dead / Dormant code
  - Metrics
  - Degree of code commenting
  - Flow of control.

## THE RESULT:

- Identified software quality risks early in the lifecycle to both Contractor & Government teams
- Improved developer's coding practices/processes
- Established/captured baseline measure for software system
- Software is going to be analyzed again and compared against baseline for measured risk mitigation improvement.

# System 3 Profile

## WHAT WE DID:

- Performed quality assessment/audit & error detection forensic services

## WHAT WE OBSERVED:

### Positives

- Code quality is higher than normally observed for similar size and language systems.

### Improvement Opportunities

- Minor issues identified included:
  - Synchronized Method Calling
  - SQL Command may permit undesired injection scripting
  - Equals() method does not use 'Instanceof' operator.
- Clean up of dead/dormant code.

## THE RESULT:

- Assessed tactical software quality risk - Low-Risk
- Established/captured baseline measure for software system for future analysis comparisons

# Independent SQA Deliverables

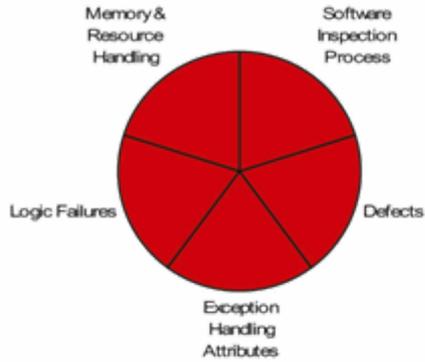
# ISQA Deliverables

- **Executive ScoreCard Summary –**
  - Designed for management for quick assimilation of data
  - Graphical charts and tables used for visual communication
  - Comparative statistical information
  - Comparative analysis
  - Highlights software manufacturing process improvements
- **Detailed Technical Report –**
  - Complete summation report of all findings and recommendations.
  - Contains a Source Code Analysis Review containing defect or issue analysis of individual instances and overall qualitative indicators per the scope of the service as identified in the software.
  - Provides recommendations for practical software development process improvements.
- **e-Defect Report(s) -**
  - Electronic reports containing the Raw Data from analysis
  - Reports are supplied in two (2) formats, a .CSV (comma separated value) and a .xls (Microsoft Excel Spreadsheet).
  - The .xls report will be delivered with “Macros” so program engineers can rapidly navigate from the defects to the actual source code using a code editor program.

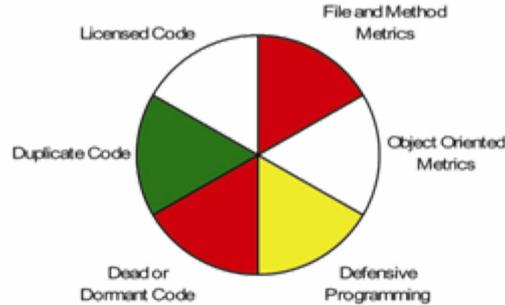
# ISQA Deliverables: Scorecard



Inspection Attributes

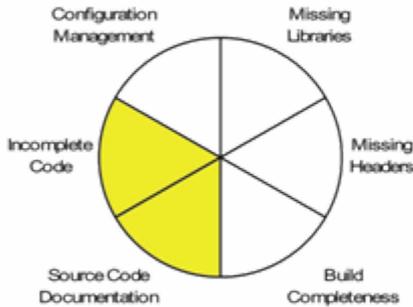


Structural Metrics



System Name	SYSTEM ABC
Assessed Lines of Code	303,397
Assessed Lines of Code Without Comments	262,535
File count	333
Class count	0
Method count	5,946

Code Completeness

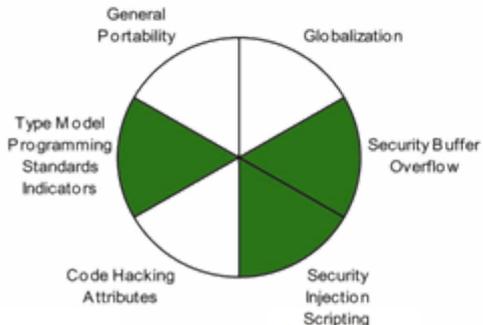


Complexity

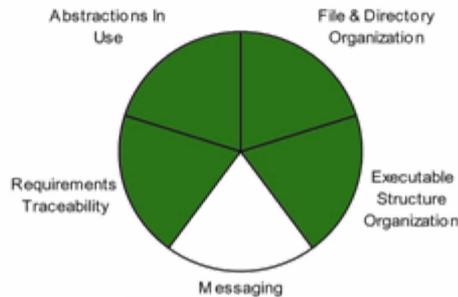


- Unrated
- Pending
- Exceptional
- Adequate
- Cautionary
- Concern

Portability & Security



Architecture & Design



## OVERALL CATEGORY RATING

Inspection Attributes	<input type="checkbox"/>
Structural Metrics	<input type="checkbox"/>
Code Completeness	<input type="checkbox"/>
Complexity	<input type="checkbox"/>
Portability & Security	<input type="checkbox"/>
Architecture & Design	<input type="checkbox"/>



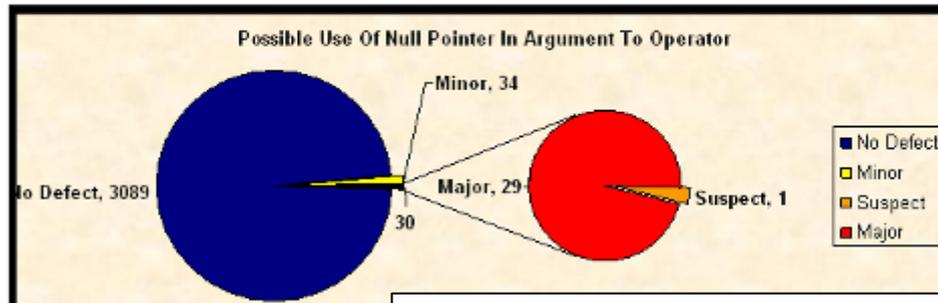
# ISQA Deliverables: Scorecard



CODE INSPECTION RESULTS												
		Instances	Major	Suspect	Minor	Bad	Style	No	Defect	Informational	% Assessed	
<b>INSPECTION ATTRIBUTES</b>												
Defects:	Possible Use Of Null Pointer In Argument To Operator	3153	29	1	34	0		3089		0	100	
Defects:	Potential Null Pointer, Dereferenced, Passed	2350	134	0	0	0	2130	150		21	100	
<b>STRUCTURAL METRICS</b>												
Defects:	<b>File and Method Metrics:</b> High Number Of Method Return Points		1019	0	0	0		1019		0	100	
Defects:	<b>File and Method Metrics:</b> Method Cyclomatic Complexity Exceeded		324	0	0	0		324		0	100	
<b>COMPLEXITY</b>												
Defects:	<b>File and Method Metrics:</b> Lines TSLOC Exceeded			302	0	0	0	0	0	192	110	100
Defects:	<b>Interface Complexity:</b> CORBA Or IDL Artifacts Detected			204	0	0	0	0	0	204	0	100
Defects:	<b>Interface Complexity:</b> Microsoft Distributed Processing Tools Detected			12	0	0	0	0	0	12	0	100
<b>PORTABILITY &amp; SECURITY</b>												
Defects:	<b>Security Buffer Overflow:</b> Apparent Data Overrun For Function			5	0	0	0	1	2	2	0	100
Defects:	<b>Security Buffer Overflow:</b> Context Requires A Scalar; Function; Array; Or Struct			19	0	0	0	0	0	19	0	100
Defects:	<b>Security Injection Scripting:</b> SQL Command May Permit Undesired Injection Scripting			138	0	0	0	0	1	137	0	100
Defects:	<b>Security Injection Scripting:</b> SQL Script Executed Within Application Code			93	0	0	0	0	0	0	93	100
Defects:	<b>Security Injection Scripting:</b> Operating System Command Or Shell Script File Executed Within Application Code			56	0	0	0	0	0	0	56	100
<b>CODE COMPLETION</b>												
Defects:	<b>Type Model Programming Standards Indicators:</b> Boolean Always Evaluates To [True/False]			337	0	0	0	0	314	22	0	100
Defects:	<b>Type Model Programming Standards Indicators:</b> Pointer Conversion Widens Size			113	0	0	0	0	58	55	0	100
<b>ARCHITECTURE &amp; DESIGN</b>												
Defects:	<b>Executable Structure Organization:</b> Main Procedure Found			80	0	0	0	0	0	12	68	100
Defects:	<b>Requirements Traceability:</b> Requirements Documentation Detected			2	0	0	0	0	0	0	2	100
Defects:	<b>Abstractions In Use:</b> Use Of C++ Standard Template Library Detected			106	0	0	0	0	0	60	46	100
<b>Summary of Issues Found</b>					165	8	210	11,478	15,275	2,352		
<b>KEY DEFECTS</b>					173							
<b>ALL DEFECTS</b>					383							

**Possible Use Of Null Pointer In Argument To Operator (3153 Instances, 3153 Reviewed)**  
 Use of null pointers found in such contexts include: Unary \*, pointer increment (++) or decrement (--), addition of pointer value to a numeric value, and subtraction of two pointers. In the case of binary operators, one of the words 'left' or 'right' is used to indicate the side of the expression that contains the null pointer. Dereferencing a null pointer may lead to an uncaught exception. Analysis of the errors generated from this category should be corrected as soon as possible; if not serious, the code should be corrected when convenient. This instance is listed as Minor when there is no potential for a null reference to cause a system crash. It is listed as Major when a system crash or serious software failure can be shown to result from the instance.

## Defect Categories Defined



### 8.5 Memory & Resource Handling

This scale identifies the presence of standards when managing resources including synchronization, and release. Mismanagement of system resources can slow, stop, or crash a system.

#### CONCERN

**SYSTEM\_ABC Results:**  
 Custodial Pointer Has Not Been Freed  
 Freeing the pointer before releasing the system memory allocation requirements should also be implemented.

```

Example:
File Name: SYSTEM_ABC_XYZ\infrastructure\general_utils.c
Line: 87
void *safe_memdup(void *ptr, size_t size) {
    [REDACTED]
    [REDACTED]
    [REDACTED]
    [REDACTED]
}
    
```

**Scorecard Charts Correlate back to Technical Report**

Defect Classification:	Major
File Path:	[REDACTED]
Line of Code:	912
Explanation:	Probable error in SYSTEM_ABC_XYZ\ces\framework_interfaces\msi_client.c. Processes only if the pointer is null.
Defect Classification:	Major
File Path:	[REDACTED]
Line of Code:	92
Explanation:	malloc returns a null pointer if it is unable to allocate the memory region requested. The code does not check the pointer return type before the subsequent assignments. Dereferencing the null pointer of an unsuccessful malloc will cause a runtime error.
Defect Classification:	Major
File Path:	[REDACTED]
Line of Code:	103
Explanation:	malloc returns a null pointer if it is unable to allocate the memory region requested. The code does not check the pointer return type before the subsequent assignments. Dereferencing the null pointer of an unsuccessful malloc will cause a runtime error.

**All Defects Are Identified to Their Location within the System Down to the Line of Code**



# iSQA Deliverables: Defects Report

## (Raw Data Files)



Defect Classification	Major Defect
File Path:	\\Company XYZ\woodstock\workflow\services\threddatahandler.java
Line of Code:	233
Explanation	The method endElement() is empty. Functionality is missing.

All Defects Are Delivered in “Raw-Data Files” as .csv and .xls file formats with macros enabling engineers to traverse easily & quickly back to the code.

B	C	D	E	F
FileName	Line	Category	Classification	Comments
ABC XYZ\cairn_v10.2_end\otk.c	10487	Expert Analysis	Major	See Defects section and appendix of the project report.
ABC XYZ\infra\structure\otk_lib\otk_lb.c	4682	Expert Analysis	Major	MEMORY LEAK: Pointer trap is malloc'd in line 4682 and then is
ABC XYZ\xyz\data_parea.c	1737	Expert Analysis	Major	NULL Reference Potential: See the Defects subsection under Inspe
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb.c	299	Expert Analysis	Major	NULL Reference Potential: Failure to check for null return after line 2
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb.c	101	Expert Analysis	Major	NULL DEREFERENCE: See line 93 where f could be null. Derefe
ABC XYZ\cairn_v10.2_end\window_atk.c	18481	Expert Analysis	Major	See Defects section and appendix of the project report.
ABC XYZ\cairn_v10.2_end\gui_atk_3.0.c	23430	Ignoring Return Value C/ Symbol	Major	The return value of Lock_file() should be examined and dealt with.
ABC XYZ\cairn_v10.2_end\file_xrower.c	824	Ignoring Return Value C/ Symbol	Major	The return value of Lock_file() should be examined and dealt with.
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	1002	Ignoring Return Value C/ Symbol	Major	The return value of Lock_file() should be examined and dealt with.
ABC XYZ\xyz\mal_client.c	104	Ignoring Return Value C/ Symbol	Major	The return value of net_connection_send() should be examined and :
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	138	Ignoring Return Value C/ Symbol	Major	The return value of net_connection_send() should be examined and :
ABC XYZ\cairn_v10.2_end\win\ams\afq.c	4818	Ignoring Return Value C/ Symbol	Major	The return value of ReadObject(Geomables0) should be examined as
ABC XYZ\cairn_v10.2_end\common_sub.c	3174	Possible Use Of Null Pointer In Argument To Operator	Major	The pointer may contain a null; causing a crash. The same problem
ABC XYZ\cairn_v10.2_end\radio_model_vmr.c	222	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb.c	838	Possible Use Of Null Pointer In Argument To Operator	Major	Dereferencing a null pointer may cause an abort. Similarly, in this pr
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	884	Possible Use Of Null Pointer In Argument To Operator	Major	Null pointer could cause problem. Similar anomalies are contained
ABC XYZ\cairn_v10.2_end\radio_model.c	218	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\radio_model_src.c	287	Possible Use Of Null Pointer In Argument To Operator	Major	calloc returns a null pointer if it is unable to allocate the memory re;
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	867	Possible Use Of Null Pointer In Argument To Operator	Major	realloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	47	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\common_sub.c	3021	Possible Use Of Null Pointer In Argument To Operator	Major	Could cause a major problem when not checked for a null value and
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	744	Possible Use Of Null Pointer In Argument To Operator	Major	c_new0 call a calloc() which returns a null pointer if it is unable to a
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	1217	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	238	Possible Use Of Null Pointer In Argument To Operator	Major	Kindor could be null at line 238. It is not checked for null before it i
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	112	Possible Use Of Null Pointer In Argument To Operator	Major	either will be NULL at the completion of the loop at line 104
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	628	Possible Use Of Null Pointer In Argument To Operator	Major	c_new0 call a calloc() which returns a null pointer if it is unable to a
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	348	Possible Use Of Null Pointer In Argument To Operator	Major	calloc returns a null pointer if it is unable to allocate the memory re;
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	691	Possible Use Of Null Pointer In Argument To Operator	Major	c_new0 call a calloc() which returns a null pointer if it is unable to a
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	398	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	718	Possible Use Of Null Pointer In Argument To Operator	Major	malloc returns a null pointer if it is unable to allocate the memory re
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	380	Possible Use Of Null Pointer In Argument To Operator	Major	The pointer may contain a null; causing a crash. The same problem
ABC XYZ\cairn_v10.2_end\otk_lib\otk_lb\otk_browser_atk.c	78	Possible Use Of Null Pointer In Argument To Operator	Major	either returns a value that could be null. These values are deref

# The Value of Independent Software Quality Assessment



# ISQA Cost Impacts: Return on Investment (ROI)

<b>DoD and SEI (CMU) Statistics</b>
5-15 Flaws in every 1,000 LOC
75 minutes/ flaw fix time

<b>Industry Accepted ROI Metrics</b>
\$10,000/bug to Fix Defects
80% of Cost = Finding Bugs

## Customer 1

335 Defects x \$8,000/Defect = \$2,680,000
Defect ROI
<b>*Estimated ROI \$2,680,000 - \$545,000 = \$2,135,000</b>

## Customer 2

219 Defects x \$8,000/Defect = \$1,608,000
Defect ROI
<b>*Estimated ROI \$1,608,000 - \$219,000 = \$1,389,000</b>

## Customer 3

1895 Defects x \$8,000/Defect = \$15,160,000
Defect ROI
<b>*Estimated ROI \$15,160,000 - \$1,214,000 = \$13,946,000</b>

## Customer 4

70 Defects x \$8,000/Defect = \$560,000
Defect ROI
<b>*Estimated ROI \$560,000 - \$140,000 = \$420,000</b>

**Overall ISQA Impact on these supported systems  
yields savings of \$17,890,000**

# Lessons Learned

# Lessons Learned

- Developed metrics, definitions and key thresholds
- Collected and analyzed raw data to provide clear and concise information to the customer
- Focused on both positive and negative results
- ISQA is an evolving process
- Other indirect benefits of ISQA:
  - Feedback to the developer resulted in process improvement
  - Enforce compliance to software coding standards
  - Documents code quality characteristics for software sustainment
  - Contributes to software-related Technology Readiness Level rating

***ISQA provides a cost-effective, schedule-efficient, performance-enhancing means to deliver/sustain software products***

# Other PM Concerns

- **Intellectual Property**
- **Schedule Impact**
- **Information Control**
- **Time Frame for Services**

***When should my program leverage ISQA?***

# Final Thoughts and Comments...



## Why Come to the C4ISR's ISQA Team?

- We are domain experts with an understanding of the problem space based on historical observations.
- Combined process, tooling and methodology solution available as a service

## The Road to Innovation:

- Open Systems vs. Closed Systems
- Object Oriented Languages
- Software Process Maturity Certification Requirements (SEI, etc...)
- Client / Server Architectures
- Service Oriented Architectures (SOA), ....etc.....
- Independent Software Quality Assessments



**When you need Quality Software – Think ISQA**