SOFTWARE SECURITY ASSURANCE SUMMIT

December 1, 2010 | Westin Tysons Corner | Falls Church, VA

T.E.N.

presented by

FORTIFY
An HP Company

The Not-So Secret Secrets of Selling AND Achieving Success with Software Security Assurance

John Keane

Software Security Assurance Summit
December 1, 2010 | Westin Tysons Corner | Falls Church, VA
TEN
presented by
Fortify An HP Company

The Not-So Secret Secrets of Selling AND Achieving Success with Software Security Assurance

John
703-38
ikeaneir@

Caveats

I am not speaking today as an official representative of the Military Health System. The views expressed are strictly my own but they do contain data and experience from my current and past jobs in both government and private industry.

I will be 64 in January. I wrote my first lines of code in 1964 and continued until 1968. By the time I returned to graduate school in 1972, the amount of bad code being written created the impetus to construct disciplines and structures to address the problems. I saw the same patterns in the early 1980's, the early 1990's and to a much greater extent in the last few years. In may ways we are combating a problem that seems to repeat itself every decade as lessons-learned, quality, experience and value are traded in for low cost.

GEN Patton on War (if he were around today)

"No bastard ever won a war by making viewgraph slides for his country."

"He won it by making the other poor dumb bastard make slides for his country."

George C. Scott as George S. Patton Jr.

Some Early Thinking on the Issues of Using Tools

"Everyone is looking for the Silver Bullet - a methodology, a tool, a 'killer' application, the latest technology - that will solve all of their current problems.
They forget that there is only one real Silver Bullet -- Continuous Hard Work"
Keane 1995

More Recent Thinking

"A Fool With A Tool is Still a Fool"

- *PMT256 - Program Management Tools Course*
- *TST203 - Intermediate Test and Evaluation*
- *Director, Federal Reserve Information Technology*

To achieve success you need a combination of :

- *Skilled People*
- *Disciplined Processes*
- *Enabling Tools and Technologies*

What We Are Doing

Terminology

Software Code Quality Checking (SCQC)

- A scan of the source code, executables, and related artifacts, e.g., documentation, to ensure that the System Under Review can continue with development, demonstration, and test; and can meet the stated performance, maintainability, and usability requirements within cost (program budget), schedule (program schedule), risk, and other system constraints.

- Encompasses the use of static code analysis, static security analysis, dynamic code analysis, dynamic security analysis and architectural analysis and is usually performed using automated tools.

Better Processes .. Better Products

More Terminology

Static Analysis *is the analysis of computer software and related documentation that is performed without actually executing programs built from the software.*

Static Security Analysis *is the analysis of computer software that is performed without actually executing programs to detect and report weaknesses that can lead to security vulnerabilities.*

Dynamic Program Analysis *is the analysis of computer software and related documentation that is performed by executing programs built from that software on a real or virtual processor.*

Dynamic Security Analysis *is the analysis of computer software that is performed by executing programs to detect and report weaknesses that can lead to security vulnerabilities.*

Architectural Analyses *may be supported by automated tools but are usually conducted by manual walk-through of documentation and visual inspection of the code.*

Better Processes .. Better Products

Aligned with National Strategies

- ***Software Assurance*** *is the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures to help achieve*
  - *Trustworthiness - No exploitable vulnerabilities exist, either of malicious or unintentional origin*
  - *Predictable Execution - Justifiable confidence that software, when executed, functions as intended*
  - *(http://samate.nist.gov/Main_Page.html)*

- ***Software Assurance*** *is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner. (CNSS Instruction No. 4009, 26 April 2010)*

- ***DOD Software Assurance*** *is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software*

- ***DHS Software Assurance*** *is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner to achieve*
  - *Trustworthiness - No exploitable vulnerabilities exist, either of malicious or unintentional origin*
  - *Predictable Execution - Justifiable confidence that software, when executed, functions as intended*
  - *Conformance – Planned and systematic set of multi-disciplinary activities that ensure software processes and products conform to requirements, standards and procedures.*

Better Processes .. Better Products

Why Do SCQC?

- ***Software Defects***
  - *Too many, discovered too late in the system development/ testing process*
  - *Expensive to fix*
  - *Delayed delivery of products to customers*
- ***Improve Software Quality and Reduce Costs***
  - *Early detection and correction of defects*
  - *Achieve measurable return on investment through*
    - *Application of consistent standards by vendors and testers*
    - *Techniques*
    - *Tools*

Better Processes … Better Products
Independent Verification & Validation

GOAL: Implement Independent Verification & Validation best practices to improve the quality of MHS IT products and services

- ***Key Features***
  - *IV&V concepts developed by NASA in aftermath of Space Shuttle Challenger disaster*
  - ***Independent relates to using a disinterested third party***
  - ***Validation relates to "Are you building the right thing?"***
  - ***Verification relates to "Are you building the thing right?"***
  - ***Reinforce standardized, repeatable, and rigorous empirical procedures to reduce developmental risk***
  - *Provide ability to save and reuse multiple test cases and data sets for regression testing*
- ***Key Benefits***
  - *Ensure that functional requirements continue to be met as new products and services are introduced*
  - ***Improve product quality, decrease time to market, and reduce lifecycle costs***

Better Processes … Better Products
Developmental Test & Evaluation and Operational Test & Evaluation

Testing … necessary, but not sufficient

Focuses on: **Validation relates to "Are you building the right thing?"**

- ***Developmental Test & Evaluation (DT&E)**: designed by the Program Manager and normally executed by the prime contractor with government testers to observe, and by Government testers to verify requirements in the delivered product, and, when appropriate, facilitate early user involvement and contribution for the design and test processes*
- ***Operational Test & Evaluation (OT&E)**: conducted by independent agencies to evaluate system operational effectiveness, suitability, and survivability in support of the full-deployment decision review*

Verification: Software Code Quality Checking (SCQC)

Software Code Quality Checking (SCQC)

Focuses on: **Verification relates to "Are you building the thing right?"**

Not conducted by Software Developer or PM: **Independent**

- ***Purpose**: To ensure delivery of quality software code benchmarked against recognized standards (e.g., IEEE, CMMI, etc.)*
  - *Incorporates static, dynamic, static security, dynamic security and architectural analyses*
  - *Focuses on technical correctness of the code – except for performance, may not be obvious to the end user*
- ***Fact**: Poorly written code results in customer dissatisfaction and higher long-term maintenance cost*
- *Code Quality Checking supports MHS IM/IT Strategic Plan*
  - *Operational Excellence (optimal time to market);*
  - *IM/IT Effectiveness (high quality);*
  - *IM/IT Efficiency (cost-effectiveness/reduced lifecycle costs).*

Software Code Quality Checking (SCQC) – Evaluation Framework

Maintainability

- Lack of documentation
- Complex code logic
- Legacy Technology

Performance

- Inefficient use of threads
- Complex database queries
- Ad Hoc database architecture

Security

- Input data not validated
- Potential for malicious injection
- Possible hidden code

Software Code Quality Checking (SCQC) *Specific Corrections*

Recommendations
Refactor components to reduce dependencies and coupling
Reduce thread usage
Use stronger key generation algorithms
Remove unused code
Validate user input
Add buffer size checks
Optimize queries – leverage more stored procedures
Stop bypassing errors and use proper error-handling

TYPICAL APPROACH

Static Analysis Tools

- ***CAST Application Intelligence Platform (CAST)***
  - *Analyze source code (Static Analysis)*
  - *Supports multiple languages*
  - *Checks against multiple code quality metrics*
  - *Checks database schema*
- ***FindBugs***
  - *Analyze source code (Static Analysis)*
  - *Supports Java*
  - *Removes CAST Tool Bias*
- ***Fortify (Fortify Source Code Analyzer)***
  - *Analyzes source code security vulnerabilities*
  - *Supports multiple languages*
  - *Checks against multiple security standards*

Other Tools Used
Removing Any Biases

- *Past Experience*
  - *Purify Plus*
  - *NDepend*
  - *FxCop*
  - *Visual basic Project Analyzer*
  - *Ounce (IBM App Scan)*
  - *Microsoft Team System Code Analysis*
- *Under Consideration*
  - *Coverity*
  - *NIST Samate Recommendations*

Software Code Quality Checking (SCQC) *Common Issues*

- *Missing/defective Source Code*
- *Low code/comment ratio*
- *Dead Code*
- *Inconsistent logging*
- *Cross site scripting*
- *Unvalidated file-based operations*
- *SQL / Command injection*
- *Unvalidated input from an untrusted source*
- *LDAP Injection*
- *High Cyclomatic Complexity*
- *Weak Exception Handling*
- *Circular Dependencies*
- ***Weak re-usability***
- *Failure to recognize Common Weakness Enumeration (CWE) Software Weakness Types and apply recommended mitigation techniques*

Some Interesting Observations

- ***High Security Defect Density Is Closely Related to High False Positives***
- *Example 1*
  - *100,251 LOC*
  - *109,704 Security Violations (109.43%)*
  - *41,253 False Positives (99% Audit)*
  - *68,451 Residual Security Violations*
    - *6232.8 Hours to Remediate*
- *Example 2*
  - *67,395 LOC*
  - *2929 Security Violations (4.35%)*
  - *198 False Positives (6.76%)*
  - *2731 Residual Security Violations*
- *Example 3*
  - *117,000 LOC*
  - *31,314 Security Violations (26.77%)*
    - *16,083 Issues Audited (51%)*
    - *824 False Positives (5.13%)*
  - ***2947.6 Hours to Remediate***

Required Artifacts and Support
Asking for the Impossible?

- *Documentation*
  - *Application and Source Code*
  - *Integration Requirements*
  - *Knowledge Transfer*
  - *Environment Hardware Setup and/or Access*
  - *See Embedded Spreadsheet*
- *Designated HUMAN resources to address issues*
  - *Developer*
  - *SA*
  - *QA/Tester*


Microsoft Excel Worksheet

ANNUAL PROGRESS REPORT

STANDARD CONTRACTING LANGUAGE

Developed standard contracting language that ensures software code quality checking becomes "business as usual" for all our system acquisitions.

- **Key Features**
  - Provides clear and unambiguous guidance to developers as to the standards by which their products will be measured/assessed.
  - Transfers risk to the developer
- **Key Benefits**
  - Promotes common understanding with our developers as to the quality of the code we ask them to deliver.
  - Improves the likelihood of better code delivered for testing and deployment.
  - Reduces the high cost of fixing/re-testing/re-fixing the code.

"Getting things right the first time is what software code quality checking is all about"

Greg Guernsey – Test and Independent Verification and Validation

- **Ongoing activities**
  - Developed Training Program for Internal Staff
  - Conducted Initial training

Microsoft Word Document

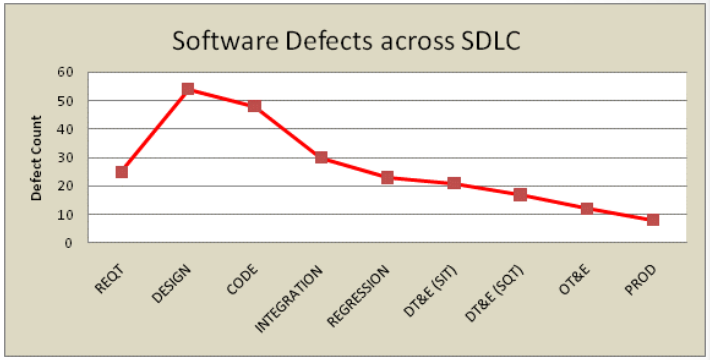
Vendor and Staff Training

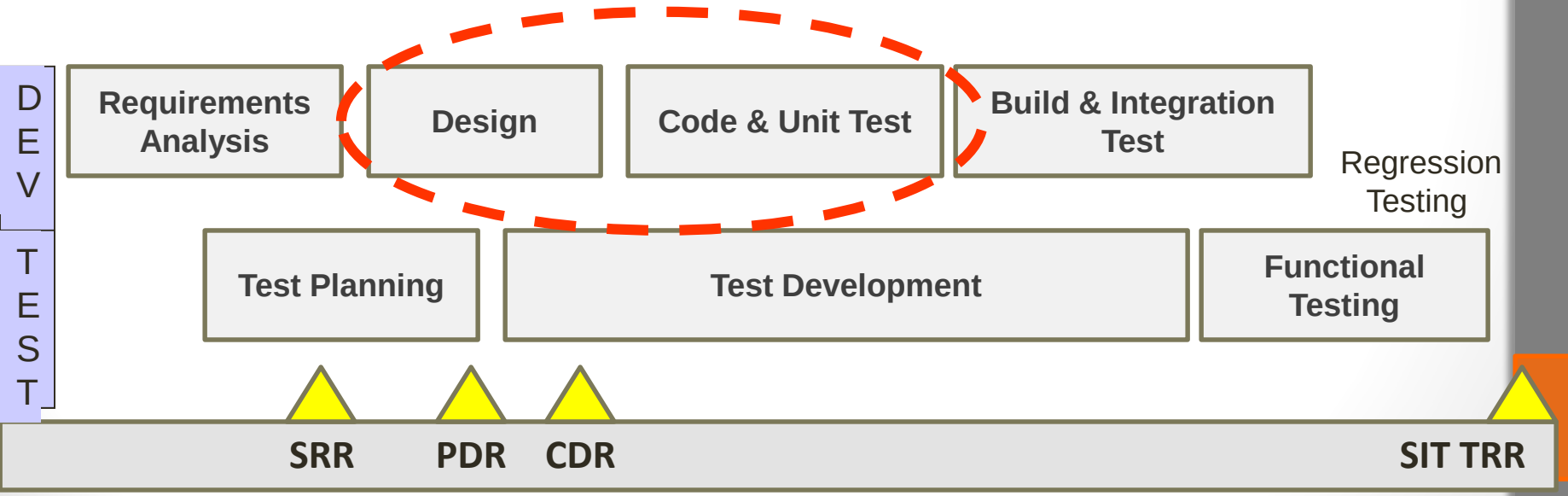
- ***DHIMS Quality Measures - Overview***
  - *DHIMS Quality Indicators*
  - *Defect Removal Efficiency (DRE) – Rayleigh Curve*
  - *Defect Removal Efficiency (DRE) – Phase Containment Matrix*

- ✓ ***Quality Control Initiatives***
  - *Root Cause of Defects*
  - *Defect Removal Efficiency (DRE) – Improving DRE for Coding Software Coding Quality Checking (SCQC)*
  - *Defect Removal Efficiency (DRE) – Notional Impact of SCQC*
  - *Process Changes from CDR to TRR – Three Changes*
    - *Process Changes from CDR to TRR – Change 1: SCQC Drops*
    - *Process Changes from CDR to TRR – Change 2: DIT Inspection*
    - *Process Changes from CDR to TRR – Change 3: Smoke Test*
  - *DHIMS Quality Assurance Surveillance Plan (QASP)*

Root Cause of Defects

- ***Insufficient inspection (defect removal activities) in***
  - *Design Phase*
  - *Code and Unit Test Phase*

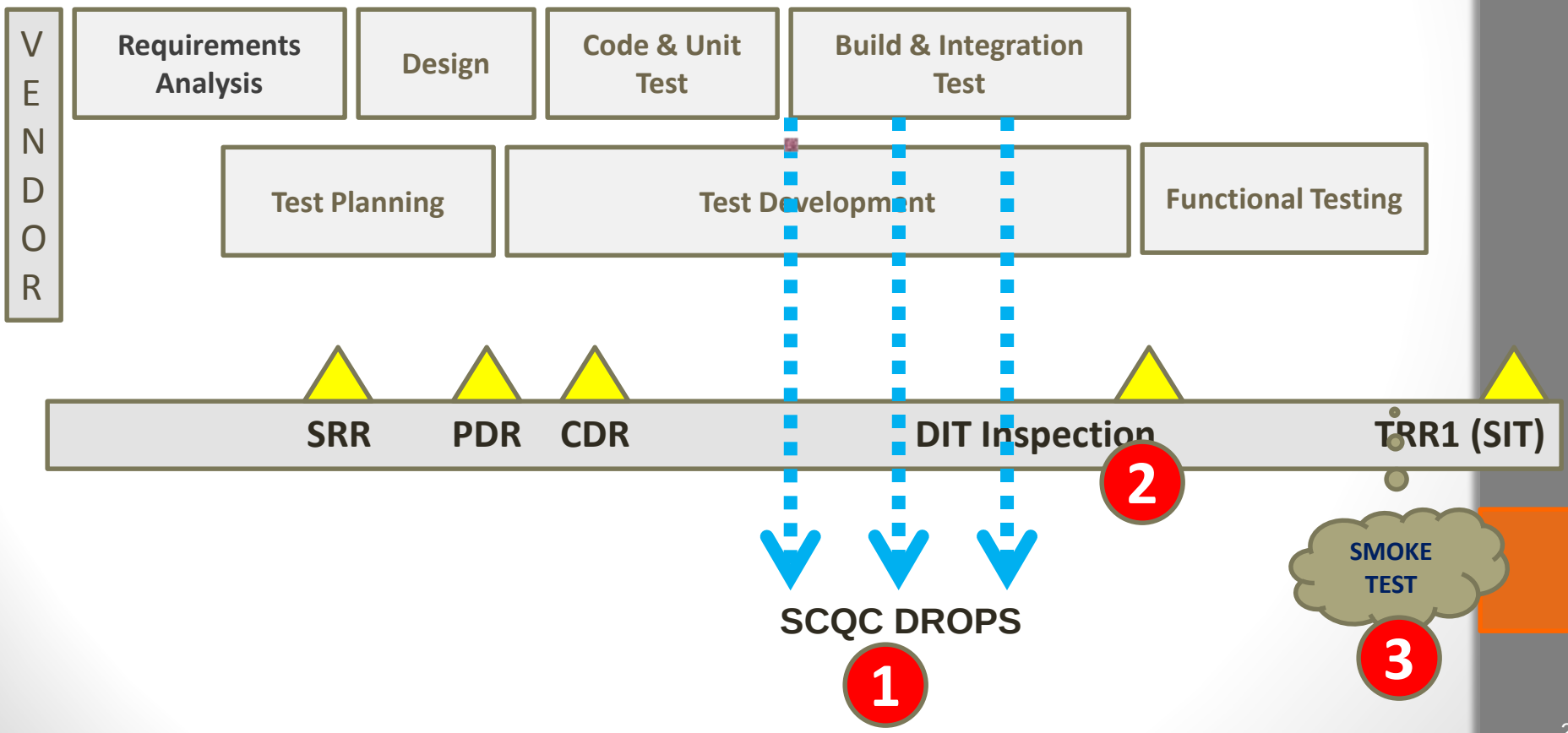




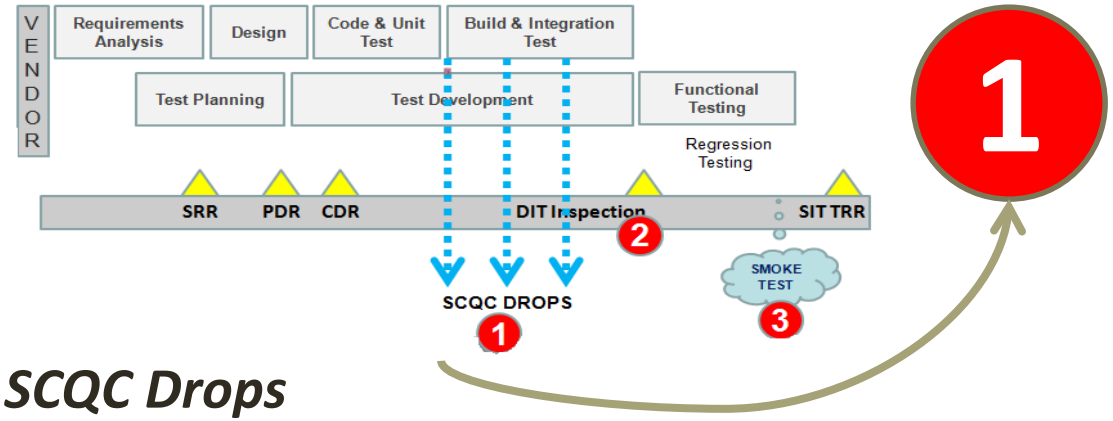
24

Process changes from CDR to TRR – Three changes

- *To support SCQC code drops to Government starts after CDR and before TRR1*
- *Government will conduct an inspection before the DIT testing phase at Vendor site*
- *Government will conduct Smoke Test prior to TRR1*

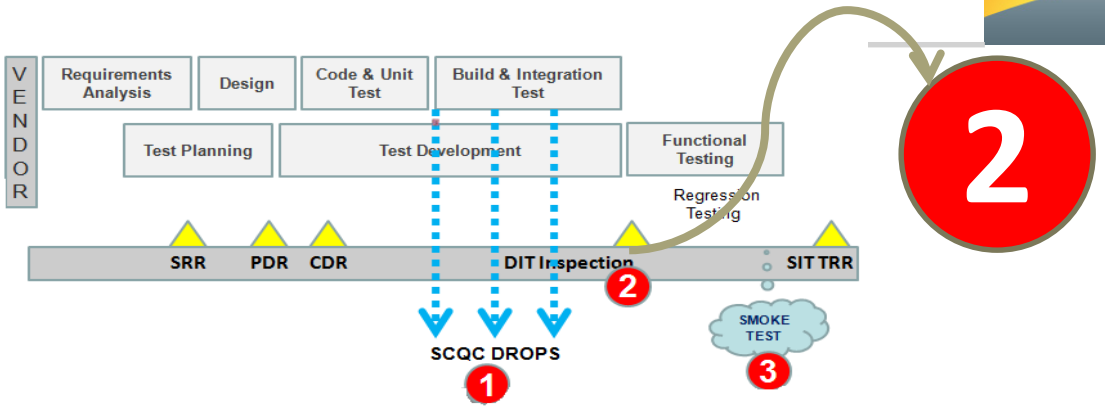


Process changes from CDR to TRR – Change 1 - SCQC Drops



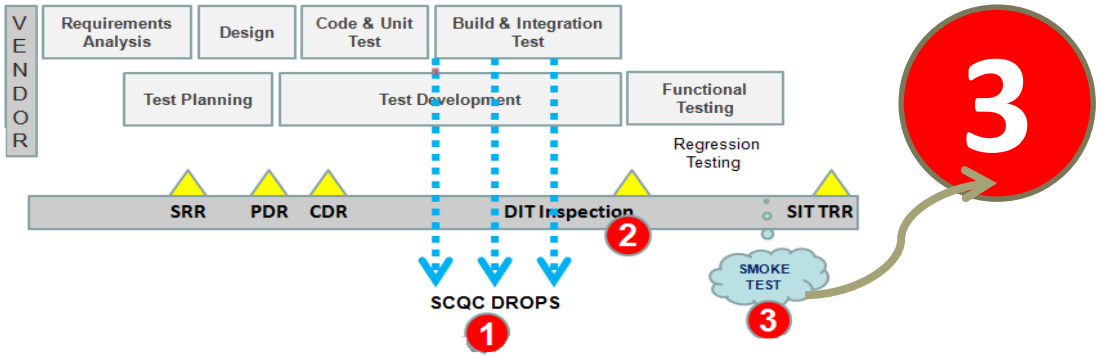
- ***SCQC Drops***
  - *Earlier the first formal code drop to DHIMS occurred after TRR. With SCQC IV&V effort, the first code drop will occur between CDR and TRR. The drop will include source code and libraries that will enable code compilation*
  - *To conduct SCQC, the Government IV&V team will work with the Vendor to determine and establish an environment for SCQC testing.*
  - *The defects found during the SCQC testing will be logged in the Phase Containment Matrix by the Vendor*
  - *The Vendor will follow the agreed upon Remediation Management Plan to address the issue or request a waiver for the same.*

Process changes from CDR to TRR – Change 2 - DIT Inspection



- ***DHIMS Inspection prior to DIT***
  - *Ensure unit tests for code are written, reviewed, executed and issues addressed*
  - *Ensure integration tests for code are written, reviewed, executed and issues addressed*
  - *Ensure regression/functional test cycles have acceptance criteria and these are met prior to regression test execution*
  - *Ensure regression/functional tests are written, reviewed, and ready for execution*
  - *Ensure code goes through Govt. SCQC process and issues found during SCQC are being addressed*
  - *Validate that the functional users are satisfied with the system usability*

Process changes from CDR to TRR – Change 3 - Smoke Test



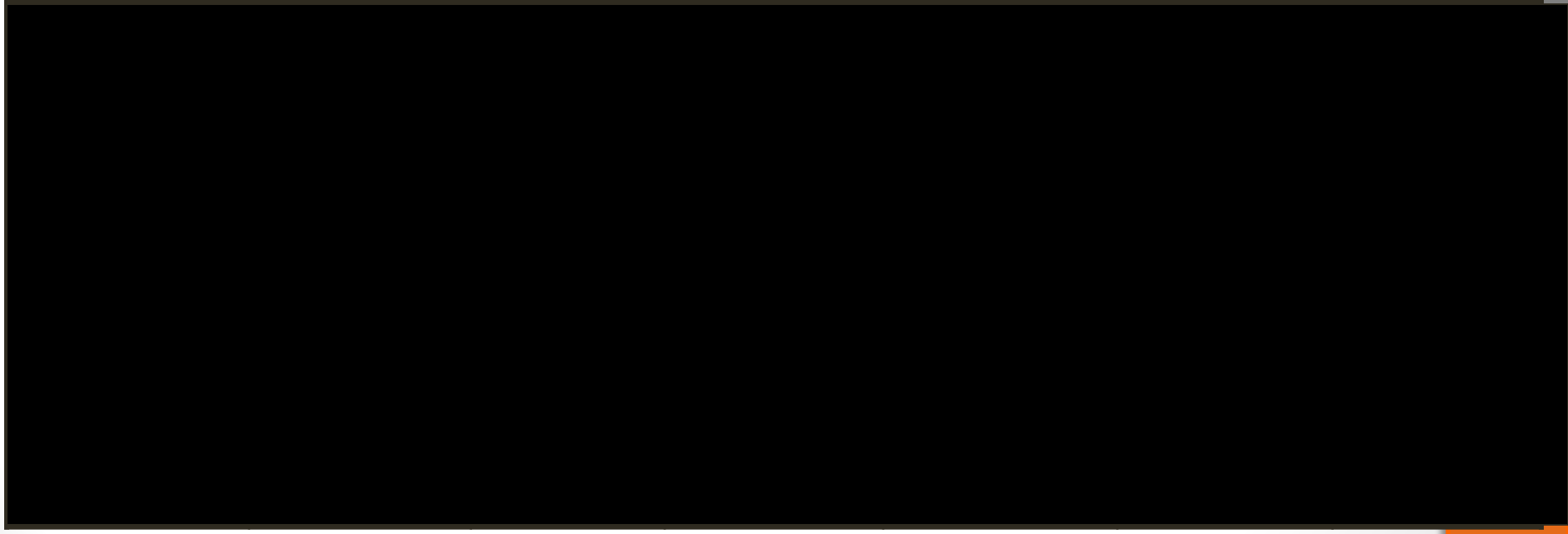
- ***Smoke Test***
  - *Contractor shall provide Smoke Test Plan*
  - *Government will conduct Smoke Test prior to TRR1. The smoke test will include the following:*
    - *Software Installation*
    - *Validation of Use Cases*
    - *CRUD of data management*
    - *Checks Interfaces*
    - *Roles & Privileges*
    - *Usability*
  - *The Smoke test is the entry criteria into TRR. Successful completion of the Smoke Test ensures that the application can proceed to TRR*

The Value Proposition

Defect Density

Conservative Model

	Requirements Analysis/Design	Code/Unit Testing	Government Testing	Production/ Deployment	Total Cost/ Investment	Return on Investment
Error Distribution	10%	20%	55%	15%		
Hours to Correct		50	120	380		
Cost per Hour		$100	$100	$100		
Cost to Fix 1000 Errors		$1,000,000	$6,600,000	$5,700,000	$13,300,000	



***Stewart-Priven Group, 2009 Presentation to PMI-MHS "Software Inspection Success"**

Return on Investment (ROI)

*Conservative Cost Model**

	Requirements Analysis/Design	Code/Unit Testing	Government Testing	Production/ Deployment	Total Cost/ Investment	Return on Investment
Error Distribution	10%	20%	55%	15%		
Hours to Correct		50	120	380		
Cost per Hour		$100	$100	$100		
Cost to Fix 1000 Errors		$1,000,000	$6,600,000	$5,700,000	$13,300,000	
SCQC Applied						
Error Distribution	10%	40%	45%	5%		
Hours to Correct		50	120	380		
Cost per Hour		$100	$100	$100		
Cost to Fix 1000 Errors		$2,013,518	$5,400,000	$1,800,000	$9,213,158	
Cost Avoidance		$1,013,518	$1,200,000	$3,900,000	$4,086,842	
SCQC Investment					$1,868,230	
ROI						**118.75%**

***Stewart-Priven Group, 2009 Presentation to PMI-MHS "Software Inspection Success"**

Return on Investment (ROI)

*Optimistic Cost Model**

	Requirements Analysis/Design	Code/Unit Testing	Government Testing	Production/ Deployment	Total Cost/ Investment	Return on Investment
Error Distribution	10%	20%	55%	15%		
Hours to Correct		50	120	380		
Cost per Hour		$100	$100	$100		
Cost to Fix 1000 Errors		$1,000,000	$6,600,000	$5,700,000	$13,300,000	
SCQC Applied						
Error Distribution	10%	76%	11%	3%		
Hours to Correct		50	120	380		
Cost per Hour		$100	$100	$100		
Cost to Fix 1000 Errors		$2,812,000	$976,800	$843,600	$4,632,400	
Cost Avoidance		$1,812,000	$5,632,200	$4,856,400	$8,667,600	
SCQC Investment					$1,868,230	
ROI						**363.95%**

***Stewart-Priven Group, 2009 Presentation to PMI-MHS "Software Inspection Success"**

What Happens When ROI Doesn't Sell?

The Graphical User Interface (GUI) Example from 1994

Why a GUI?

Increased Productivity
When Compared to
Character-Based Environment!!!

30% Productivity Improvements -- Gartner Group

35-40% Improvements -- DMR Group

WHAT DOES THAT MEAN IN THE REAL WORLD?

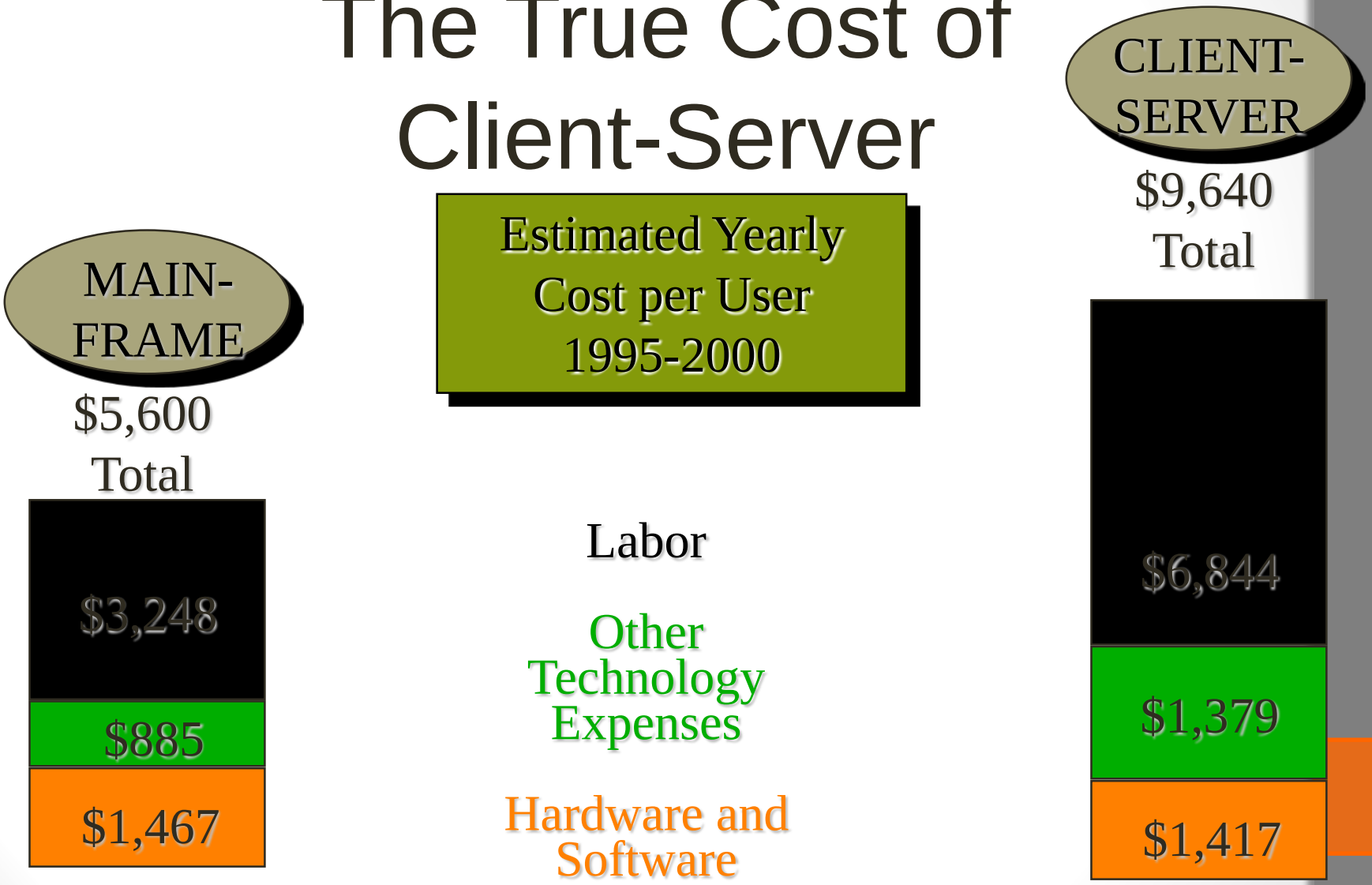
One Finance Clerk
Processing **100** Transactions per Day
Can Now Process
130 Instead
or
100 Clerks Processing **100** Transactions per Day
Can be Replaced by
77 Clerks Processing **130** Transaction per Day

100 x 100 = 10000 Transaction Per Day

77 x 130 = 10010 Transactions Per Day

The True Cost of Client-Server

Estimated Yearly Cost per User 1995-2000



Source: Fortune Magazine, April 17, 1995: Page 19

THE BUSINESS CASE FOR A GUI

(OPTION #1 - Reduce Staff)

	Text-Based Entry	Graphical User Interface	Net Change
People	100	77	23
Salaries	$7,000,000*	$5,390,000*	($1,610,000)
Terminals	$560,000	$742,280	+$182,280***
Total Cost	$7,560,000	$6,132,280	($1,427,720**)

* $70,000 Fully-Loaded Cost Per Civil Servant

** Does NOT Include Application Development Costs

*** All Increased Costs are Due to Increased IT Staff Costs

THE BUSINESS CASE FOR A GUI

(OPTION #2 - Same Staff Size but More Work)

	Text-Based Entry	Graphical User Interface	Net Change
People	130	100	-30
Salaries	$9,100,000*	$7,000,000*	($2,100,000)
Terminals	$728,000	$964,000	+$236,000
Total Cost	$9,828,000	$7,964,000	($1,864,000**)

* $70,000 Fully-Loaded Cost Per Civil Servant

** Does NOT Include Application Development Costs

I'm Ready For Any Questions