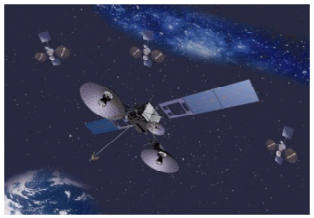
Technology Infusion of CodeSonar into the Space Network Ground Segment (Technical Briefing)

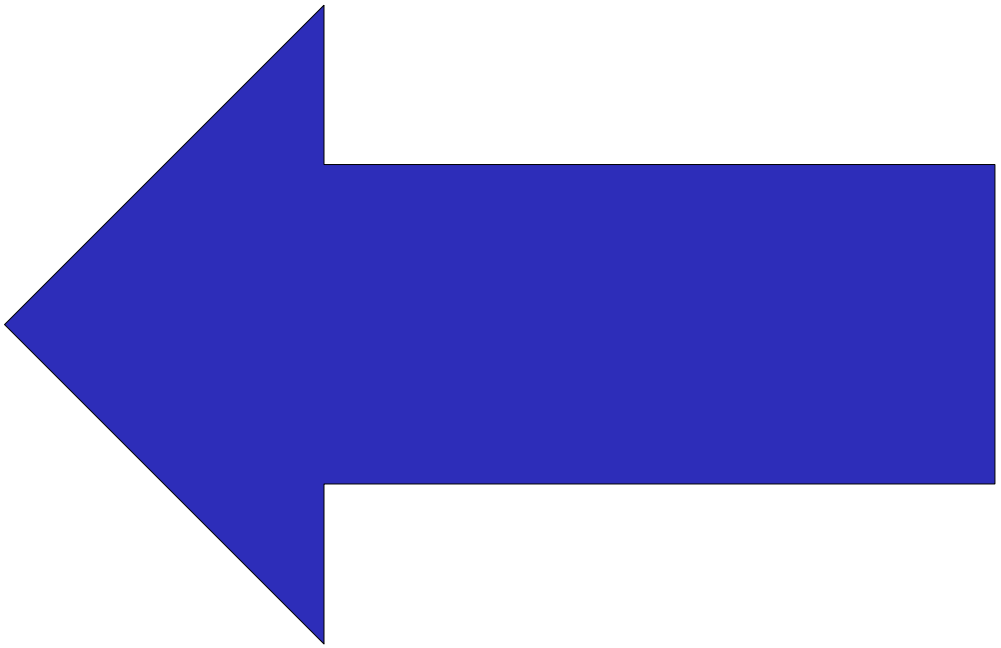
Markland J. Benson / NASA GSFC

2009 Software Assurance Symposium

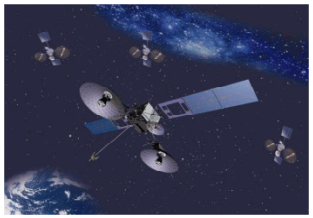
Agenda



Agenda
Preliminaries (2)
Environment (5)
Goals (2)
Approach (3)
Applicability (2)
Baseline (3)
Findings (7)
Conclusions (5)
Wrap-Up (2)



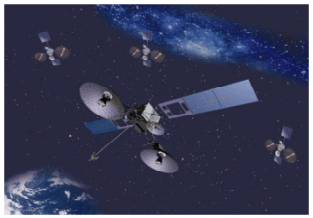
Environment-1



- Space network software must support 24x7x365 operations with a high level of integrity, confidentiality, and availability
- Current staff consists of 50 FTE to sustain and enhance software
- Approximately one-third of software effort goes into discrepancy work-off and two-thirds to enhanced capabilities

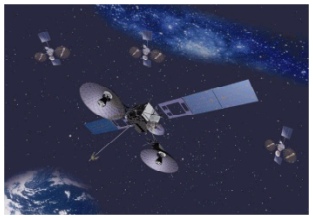
Programmer	30
Tester	5
CM	5
SysAdmin	8
Management	2

Environment-2



- Scope: ground software systems that control a satellite fleet and provide near earth communication services
- Prioritized Responsibilities
  - Investigate operational issues
  - Resolve urgent operational issues
  - Provide enhanced capabilities for customers and operations
  - Resolve routine operational issues

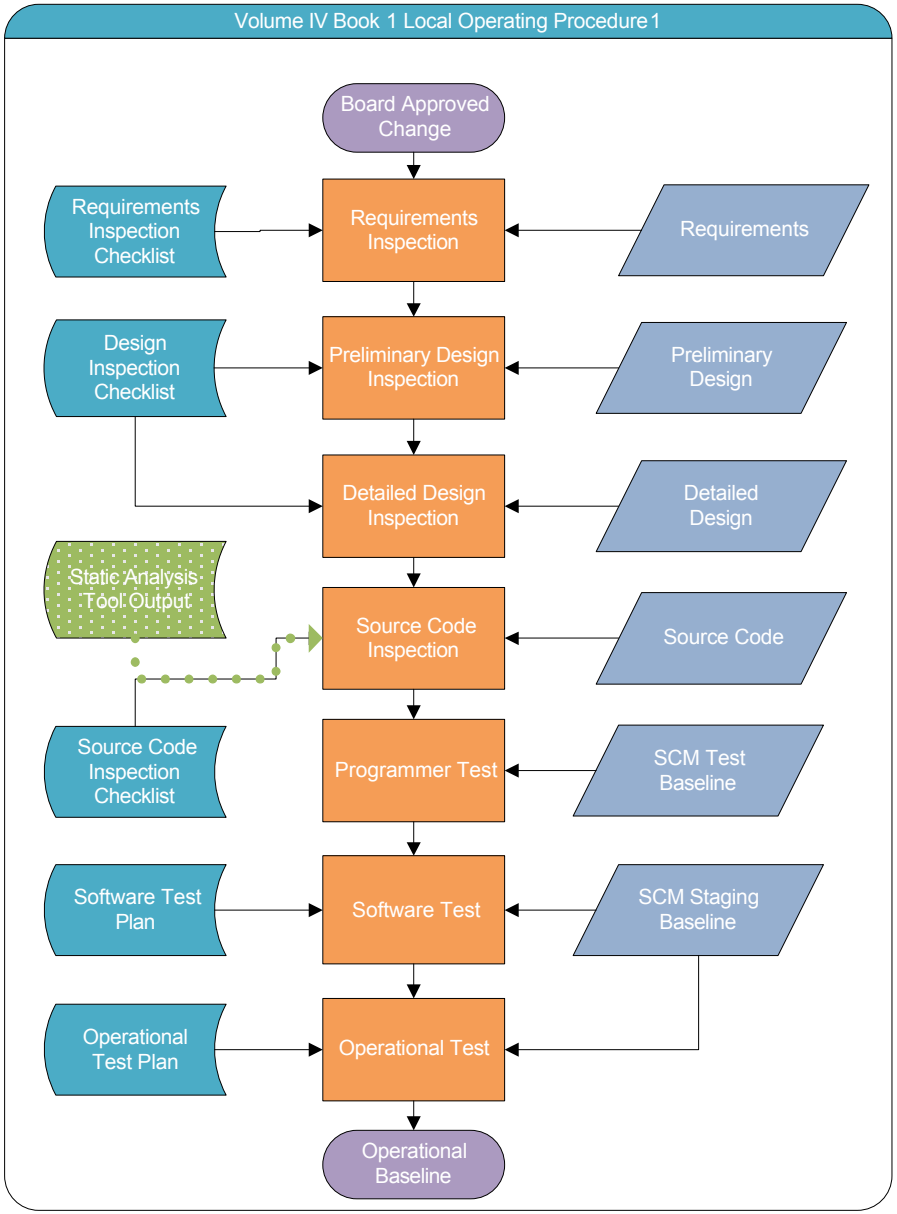
Environment-3

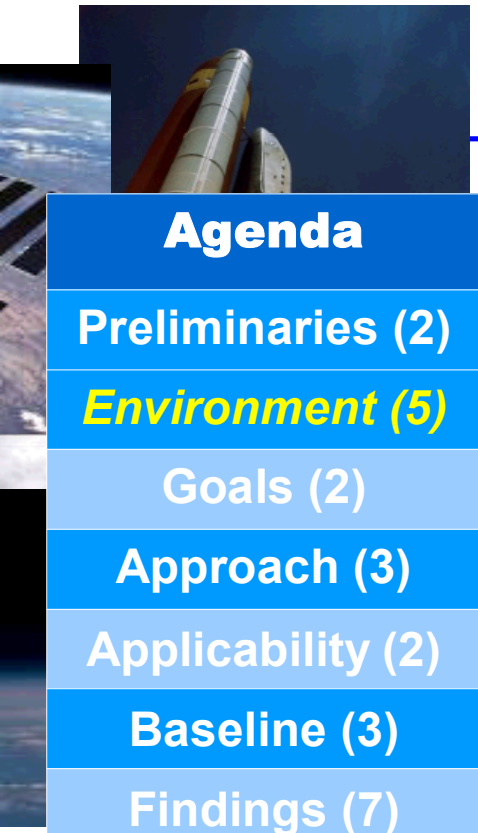


Agenda

- Preliminaries (2)
- *Environment (5)*
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- Findings (7)
- Conclusions (5)
- Wrap-Up (2)

Capability Maturity Model Integrated, Six Sigma, and NASA standards and requirements are applied in software sustaining engineering

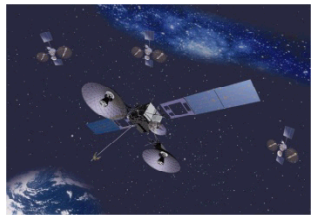




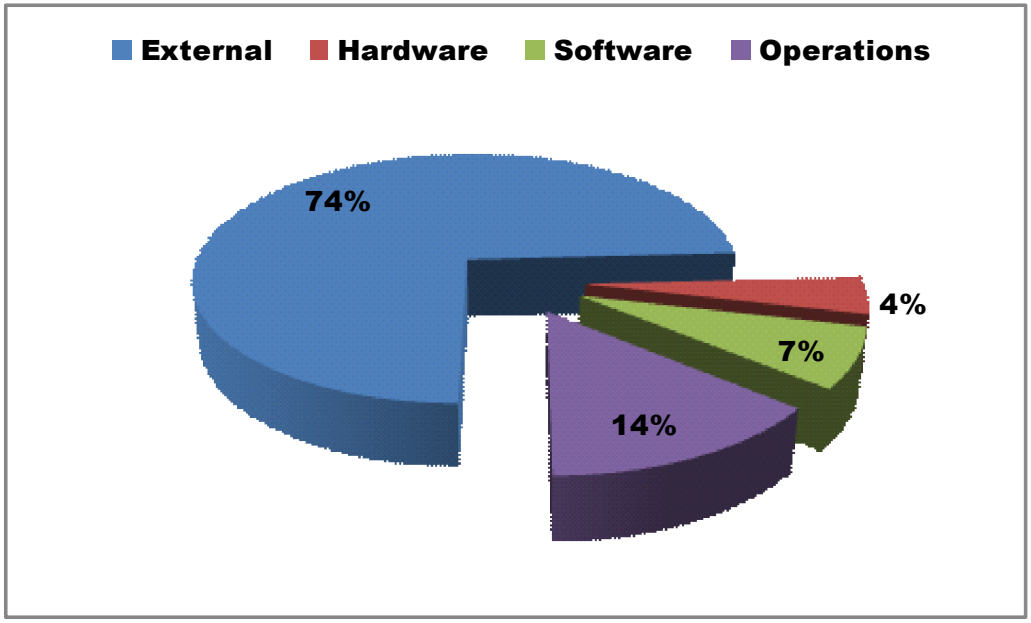
Agenda

- Preliminaries (2)
- *Environment (5)*
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- Findings (7)
- Conclusions (5)
- Wrap-Up (2)

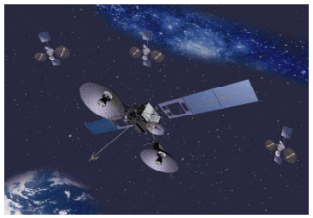
Environment-4



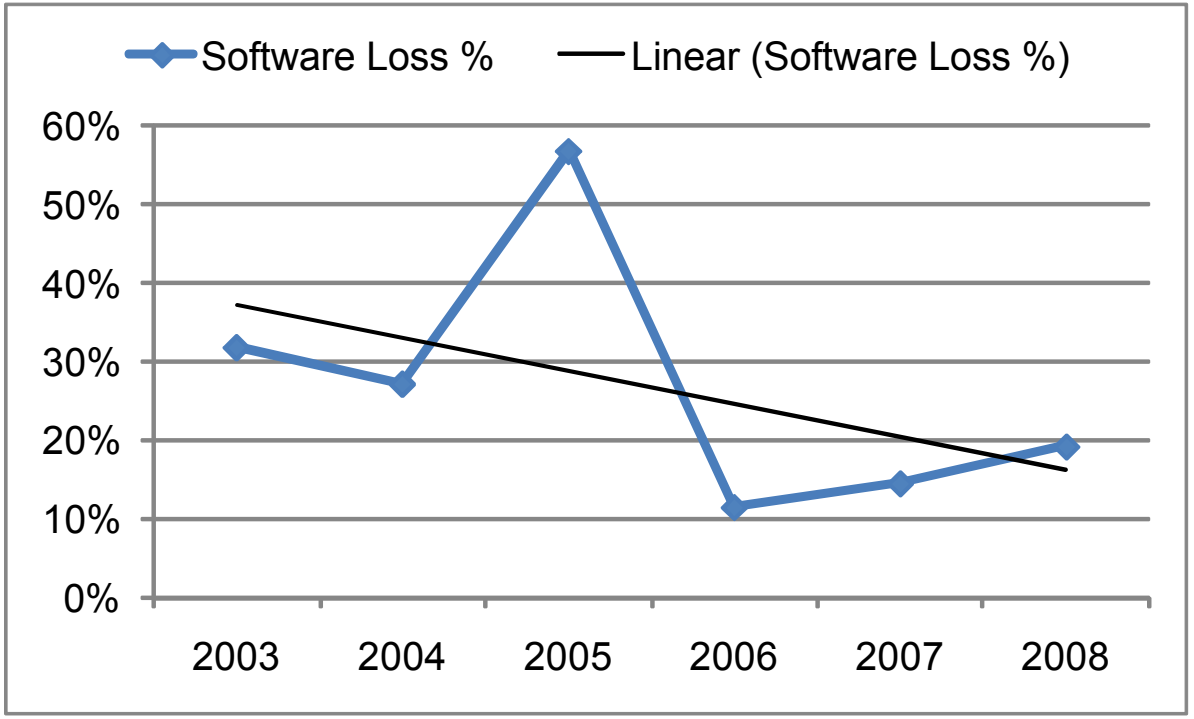
- Hours of loss due to software is 7% of overall
- The 7% slice of overall loss equates to 27% of the loss *internal* to the Space Network [that which we directly control]



Environment-5



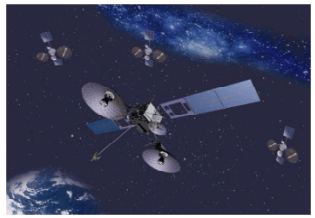
- Overall trend of software loss is down



Percent *internal loss* due to software

- Improvements in 2006 attributed to introduction of formal inspections

Goals-1



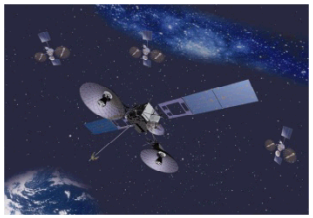
- High availability and proficiency of service requirements drive the need to reduce system defects

Goal: *Eliminate defects existing in the current baseline software*

- High demand for discrepancy resolution and new capabilities drive the need to produce software more quickly

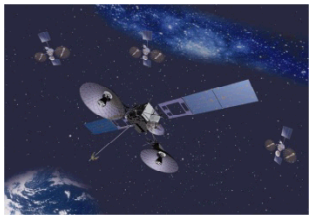
Goal: *Eliminate defects earlier in the software development lifecycle (reduce rework)*

Goals-2



- Infuse automated source code analysis technology
- Provide for a uniform analysis toolset across languages and platforms
- Apply technology to systems that have higher than average contribution to service loss
- Minimize engineer time required to apply technology

Approach-1

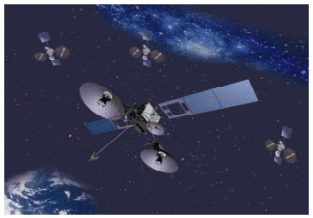


- GrammaTech's CodeSonar product provides mature analysis toolset for C and C++ (~50% of current code base) with new capability to cover Ada (~30% of current code base)

GRAMMATECH

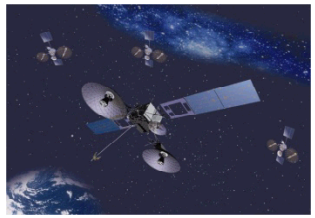
- Software engineers use CodeSonar results as an input to the existing source code inspection process

Approach-2



- Collect baseline information from the sustaining engineering processes
- Apply static analysis tool to a subset of the software baseline
- Review findings from the tool to eliminate false positives and estimate future review time to be added to inspection process
- Assess costs and benefits of larger deployment of analysis tools

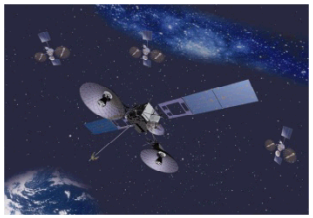
Approach-3



Apply study resources to Computer Software Components (CSCs) known to be more troublesome than others

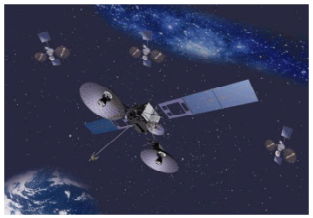
SW CSCI	Hours Lost	% Loss	DR count	% DRs	KSLOC	%KSLOC
CSCI A	6.126	37%	28	6%	200	2%
CSCI B	0.910	5%	33	7%	64	1%
Others	9.748	58%	398	87%	7865	97%
Total	16.784	100%	459	100%	8129	100%

Applicability-1



- The study is focused on large scale software developed using formal processes
- The systems studied are mission critical in nature but some use commodity computer systems
- Linux, Windows, and VxWorks operating systems are represented
- The application domain of the software is communications and spacecraft control systems

Applicabilty-2

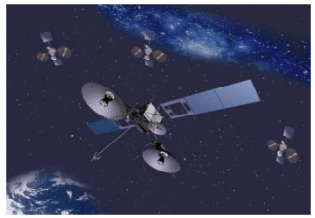


- If you have...
  - in-house maintained software...
  - using a general purpose language...
  - with formal development process...
  - where failures lead to injury or significant financial loss...
- ...this study has results that are directly meaningful to you.

(even if #4 is not true for you, lower cost analyzers likely would be of benefit)

Baseline -1

(What is SLOC anyway?)



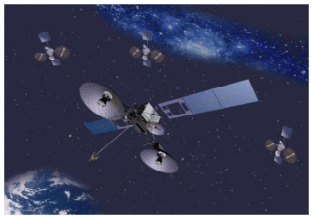
- SLOC definitions are inconsistent…
- David A. Wheeler's *SLOCCount* tool and definition is used here:

Physical SLOC is defined as follows: ``a physical source line of code (SLOC) is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) are considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) are not included.

Non-Comment Source Lines (NCSL) == Physical SLOC

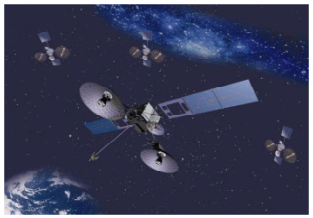
(Total) SLOC includes comments, blanks, and NCSL

Baseline-2



- Productivity of software engineers is 0.16 source lines of code (SLOC) per hour to perform requirements elicitation, design, coding, inspections, unit test, and Level 2 test
- Comparative industry productivity value is 0.6 SLOC per hour by the German Aerospace Center
- Difference can be attributed somewhat to evolving a product as opposed to new product development but productivity improvement is one of the goals

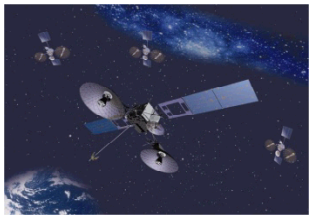
Baseline-3



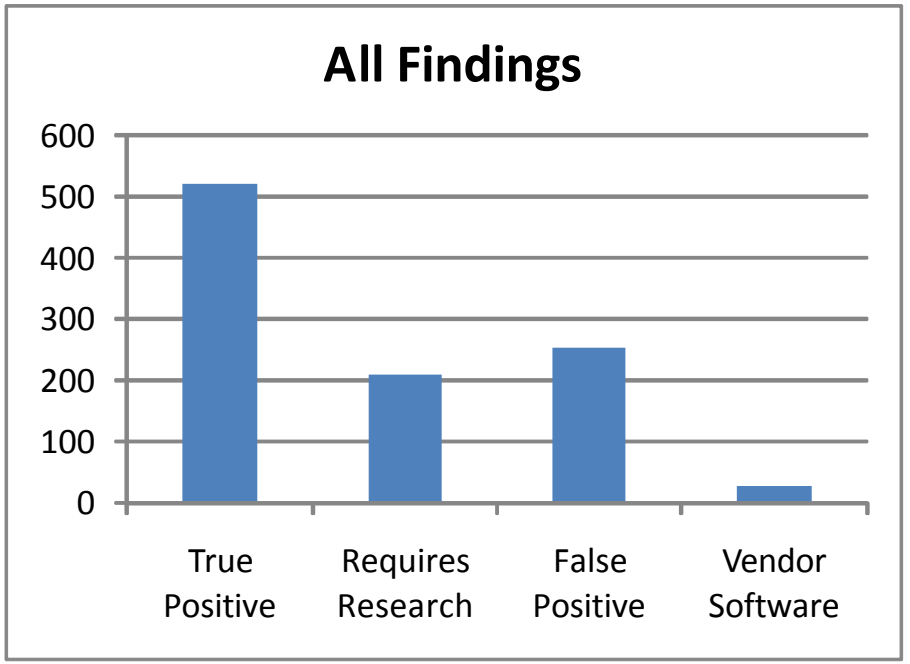
- In 2008, 204 software change requests (CRs) were completed
- Each CR produced an average of 209 new or modified lines of code

  (changes to database values not counted as SLOC modifications)
- Formal inspections caused the removal of 1,255 defects from the code; 374 of the defects were classified as major (~30%)

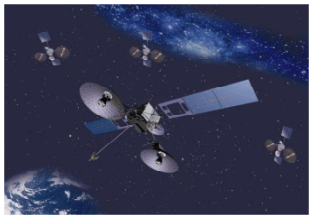
Findings-1



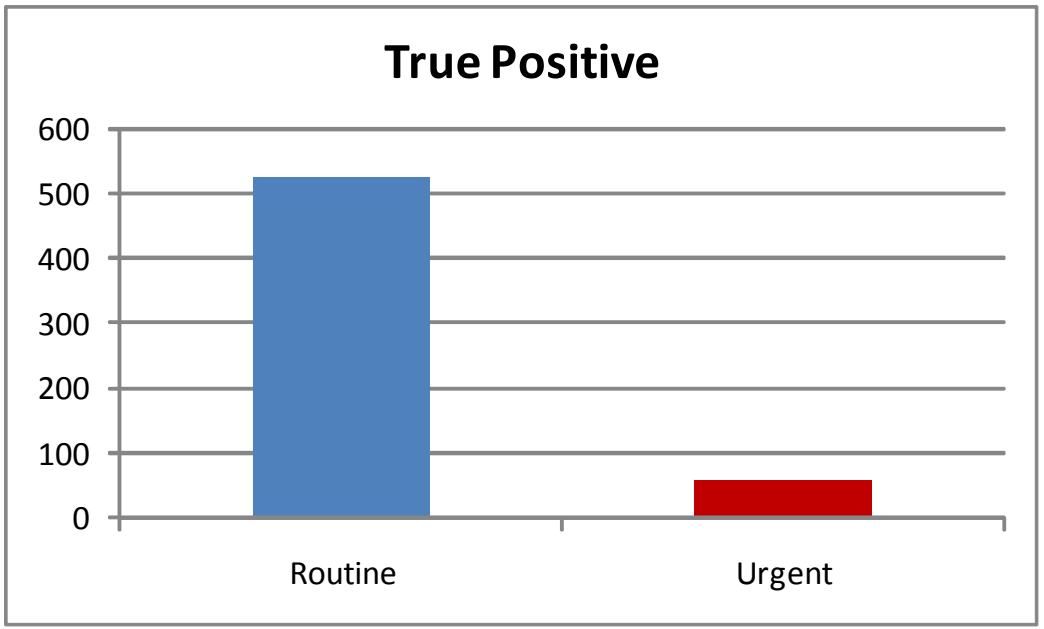
- For the 439 KSLOC (310 KNCSL) of C/C++ code analyzed in 1,245 files, 1,011 findings were produced
- All of 1,011 findings were reviewed with an average review time of ~7 minutes each



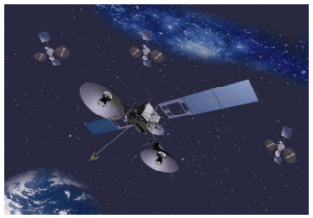
Findings-2



- Requires Research findings were classified as True or False Positive based on finding category
- Vendor Software findings were classified as False Positive

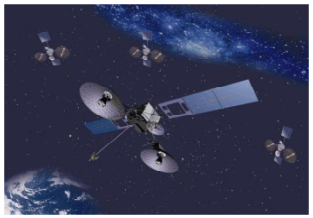


Findings-3



- Definition: An Urgent (a.k.a. major) defect is one assessed as directly impacting availability or proficiency
- Note: The large number of findings and corresponding review time is not expected to be repeated. Previously reviewed findings are filtered and only displayed if specifically requested

Findings-4

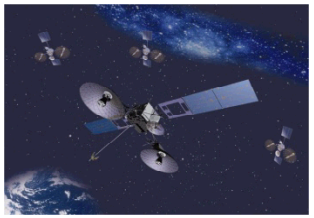


Findings were not uniformly distributed as a whole or proportionally among CSCs

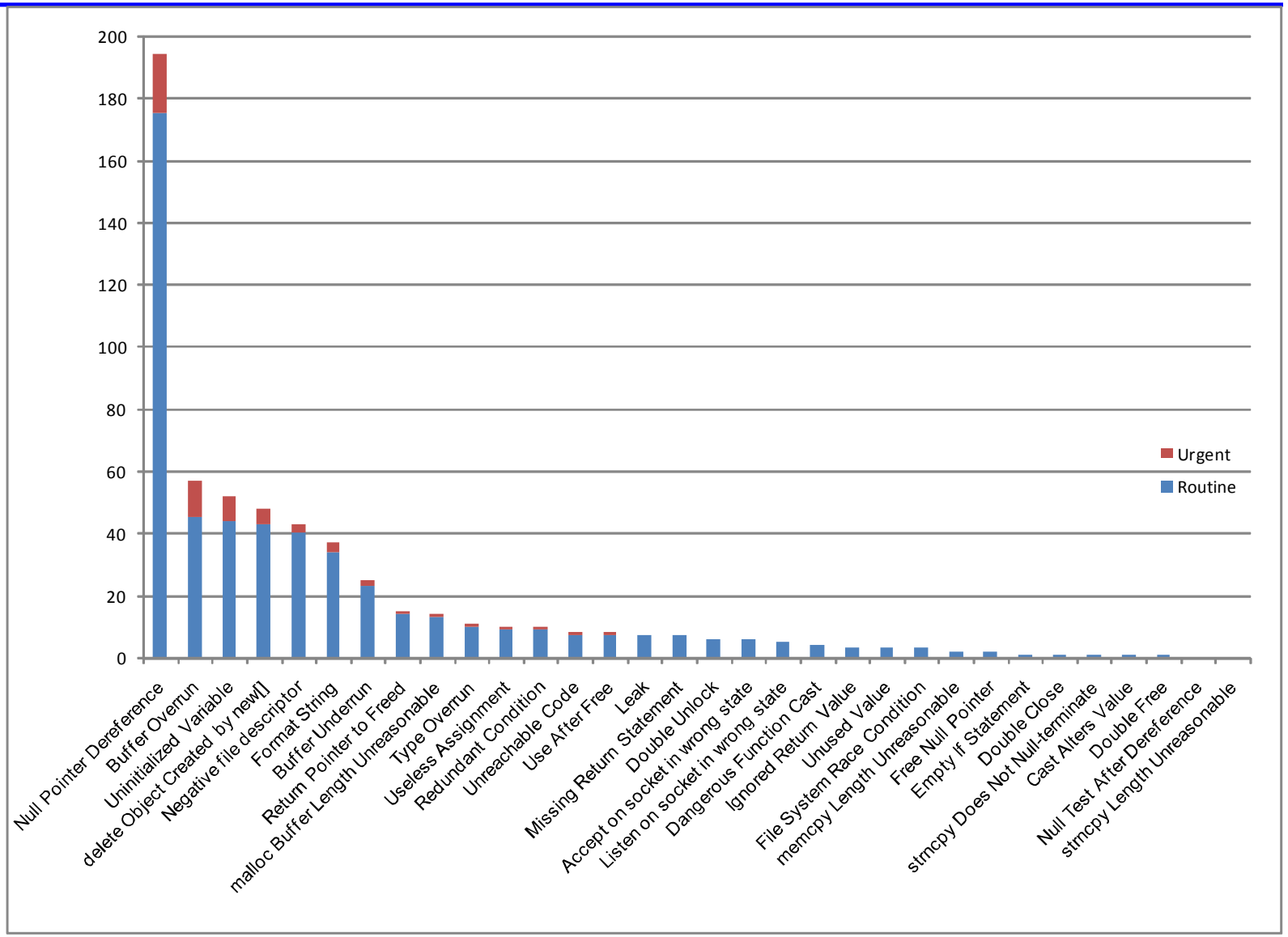
Component	K-NCSL	Defects / K-NCSL
CSCI B	121	1.8634
CSCI A - CSC B	139	1.1571
CSCI A - CSC C	22	3.7156
CSCI A - CSC D	29	4.1332

NCSL = non-comment source lines

Findings-5

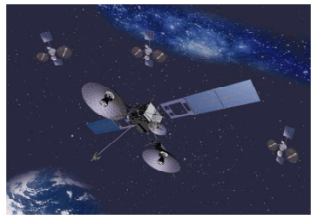


- Agenda
- Preliminaries (2)
- Environment (5)
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- *Findings (7)*
- Conclusions (5)
- Wrap-Up (2)

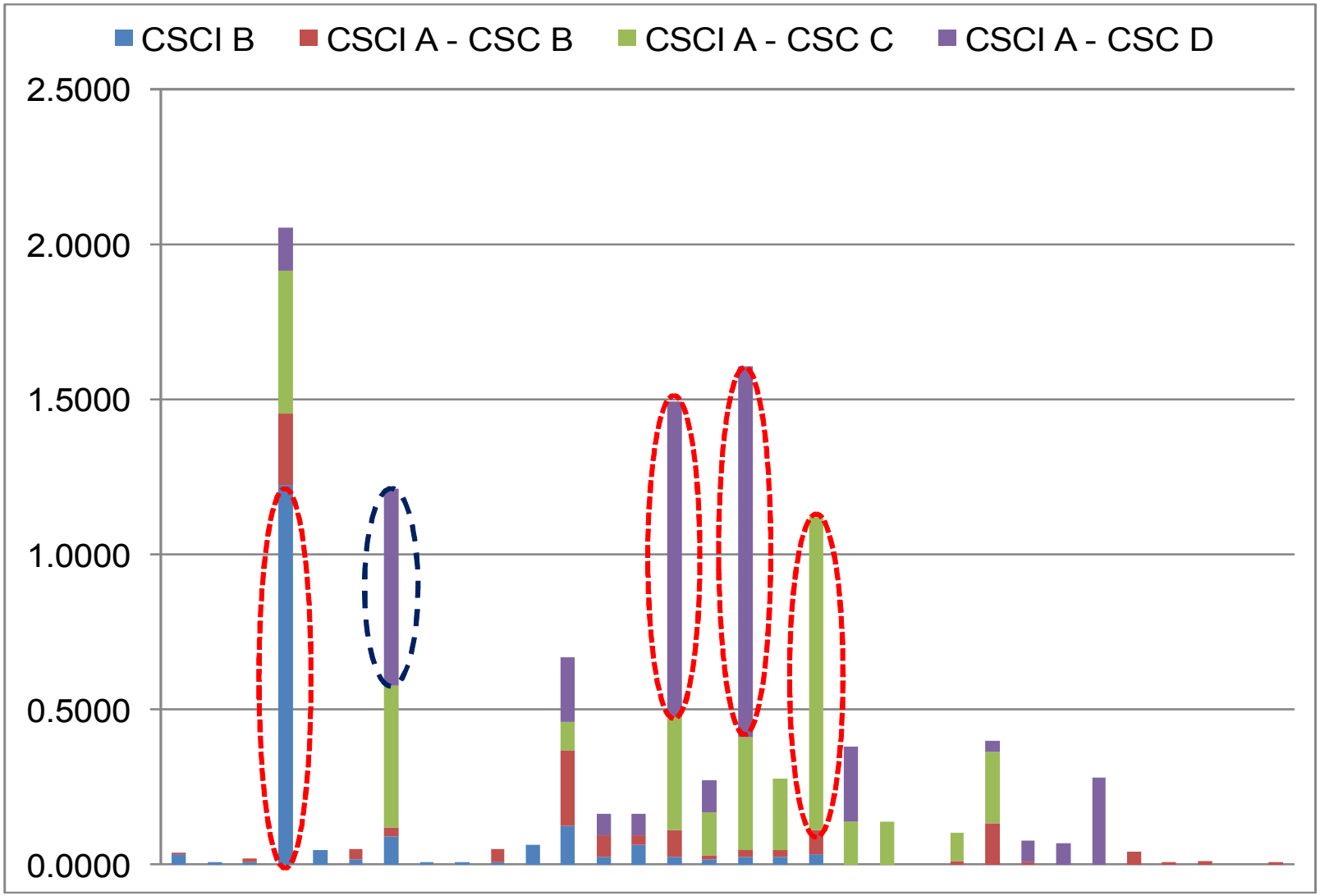


y-axis is the number of true positive findings; x-axis is finding categories

Findings-6

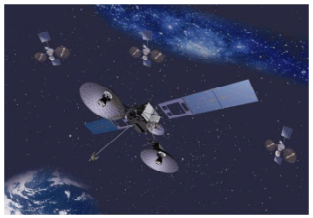


- Agenda
- Preliminaries (2)
- Environment (5)
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- *Findings (7)*
- Conclusions (5)
- Wrap-Up (2)



x-axis are the findings categories; y-axis is findings per KNCSL

Findings-7



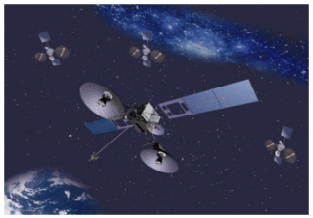
	CSCI B	CSCI A CSC B	CSCI A CSC C	CSCI A CSC D
Routine	200	130	78	118
Urgent	25	31	3	0
KSLOC	222	204	29	39
Defects/KSLOC	1.01	0.79	2.80	3.01
Urgent/KSLOC	0.11	0.15	0.10	0.00
Urgent Loss	8.75	10.85	1.05	0.00

Overall density and severity of findings vary significantly across CSCs

Urgent density is *somewhat* uniform

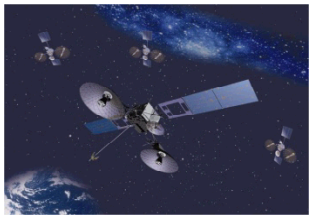
Urgent Loss in this table is a prediction of the loss expected from these defects based on the average loss per defect of 0.35

Conclusions-1



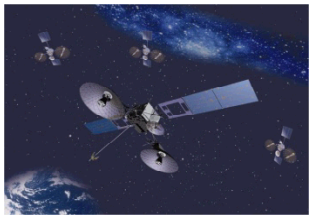
- For seven million non-comment, non-blank lines of code...
  - ...the initial cost for CodeSonar is equivalent to the fully loaded cost of a senior software engineer for one year
  - ...annual maintenance cost for CodeSonar is equivalent to about nine weeks of a senior software engineer's time

Conclusions-2



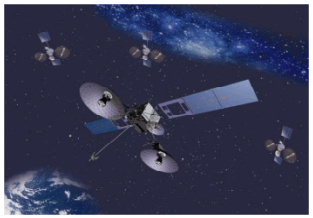
- Using baseline data combined with finding results and...
- ...very conservative cost numbers for staff time to do rework and...
- ...10% phase leakage from implementation to test and...
- ...10% phase leakage from test to operations and...
- ...and considering that one leaked defect triggers a non-trivial investigation...

Conclusions-3



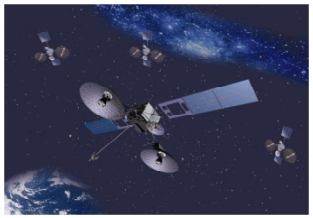
- ...then the amount saved in rework and investigation is slightly more than the annual maintenance cost of CodeSonar
- Changing only the assumption on hourly cost to a more nominal rate gives us a payback of less than five years for the license costs
- Being less conservative on other assumptions yields greater benefits

Conclusions-4



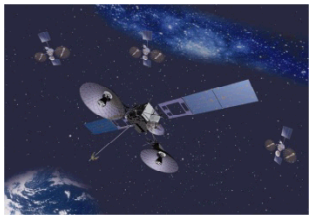
- Static source code analysis is mature, cost-effective technology
- Training and tuning of the software to the particular environment is important
- Beyond cost comparisons, CodeSonar provides a good value to the Space Network because a single critical error latent in operations puts at risk human life or once-in-a-lifetime scientific discovery

Conclusion-5



- Plan forward:
  - Incorporate automated static code analysis into sustaining engineering process for the Space Network
  - Include Ada, C, and C++ for the entire code base (one uniform tool and process)
  - Fix defects incrementally (by priority) as part of normal discrepancy work off process rather than as a large special project

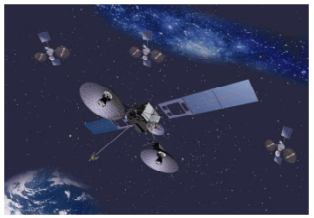
Contact Information



- Agenda
- Preliminaries (2)
- Environment (5)
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- Findings (7)
- Conclusions (5)
- *Wrap-Up (2)*

Markland J. Benson
Computer Systems Manager
White Sands Complex
PO Box 9000, Las Cruces, NM 88004
(575) 527-7034
Markland.J.Benson@nasa.gov

Questions?



- **Agenda**
- Preliminaries (2)
- Environment (5)
- Goals (2)
- Approach (3)
- Applicability (2)
- Baseline (3)
- Findings (7)
- Conclusions (5)
- *Wrap-Up (2)*

Technology Infusion of CodeSonar into the Space Network Ground Segment (Technical Briefing)

Markland J. Benson / NASA GSFC
2009 Software Assurance Symposium