NEAMS Draft SQA Plan

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Washington D.C. 10/19/2010

Gregory Pope, CSQE

Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, Ca 94551

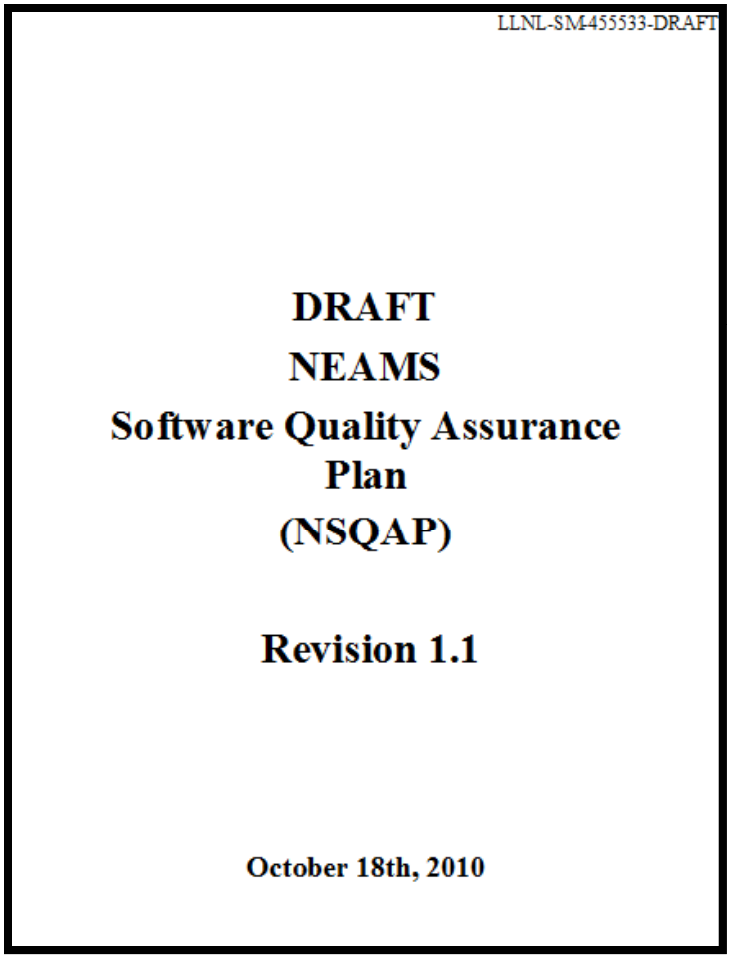
NEAMS SQAP

- Balance Agility and Discipline using Key Concepts:

  - ✓ Simplified Compliance Flow Down
  - ✓ Risk Assessment Tools
  - • Flow Down to Feeders and Libraries
  - • Required/Actual Gap Analysis
  - • Process Improvements to Close Gaps
  - • Embedded SQE's and Independent Reporting Organization
  - ✓ Automation and Recommended Tools Support
  - ✓ Shared Recommended Practices Approach
  - • Natural Metrics to Prioritize Improvements

NEAMS SQAP, The Document

- 29 Pages
- 11 Appendices
- 16 Tables
- 6 Figures
- Track Record



LLNL-SM455533-DRAFT

DRAFT

NEAMS

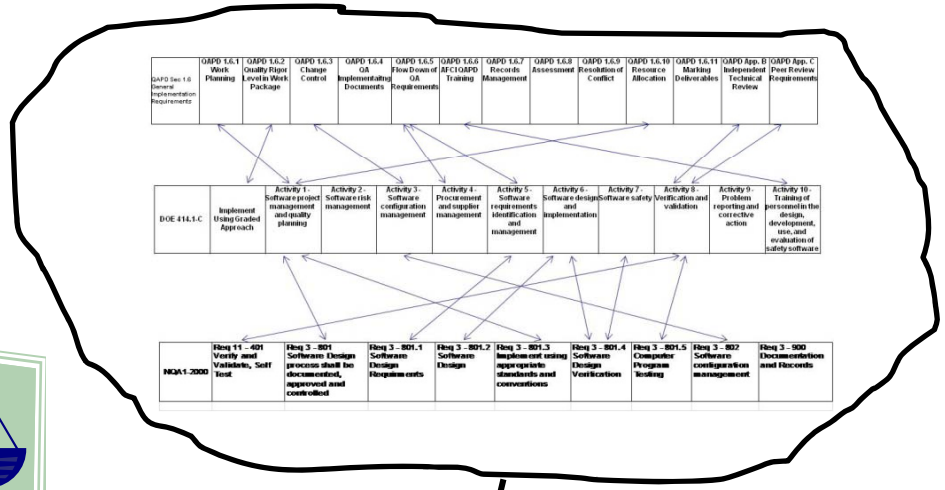
Software Quality Assurance Plan

(NSQAP)

Revision 1.1

October 18th, 2010

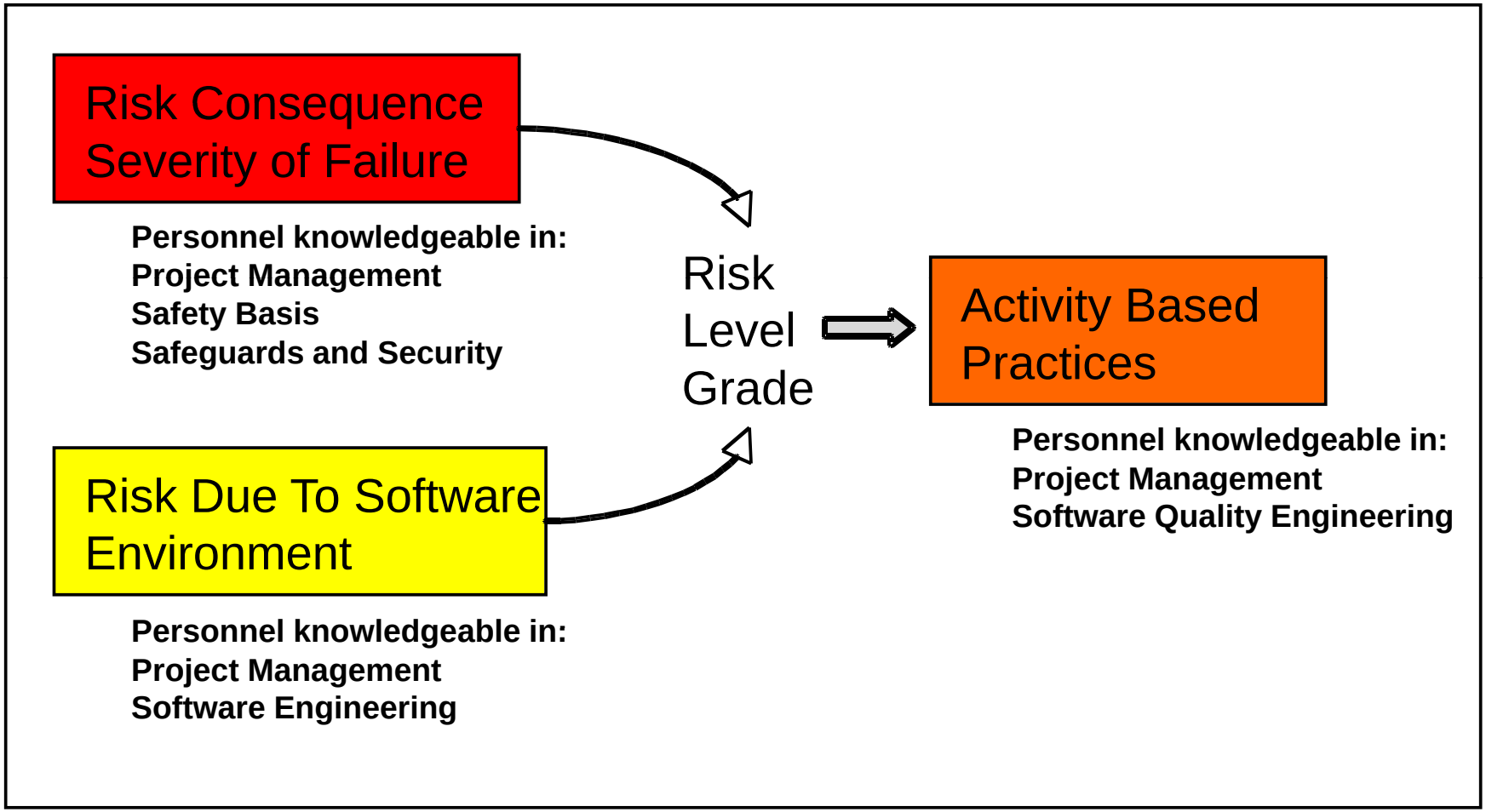
Key Concept 1:
Simplified Compliance and Risk Assessment tools



Risk Assessment → NEAMS SQAP

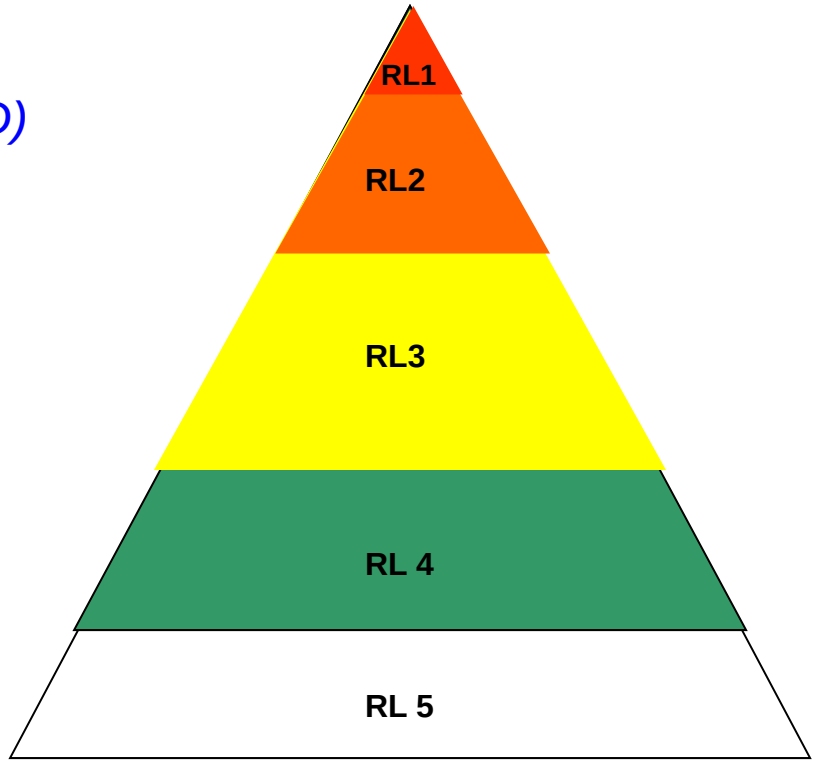
NEAMS SQAP → Reactor Codes | Fuel Codes | Waste Form Codes | Safesep Codes | Cross Cuts Codes

Key Concept 2:
Current Risk LLNL Assessment Tool



Key Concept 2:
Risk Assessment DoD 414.1-C Driven

- Activity Based
  - *From DOE Order 414.1-C (soon to be D)*
- Risk determination tool
  - *Automates calculations*
  - *Records risk assessment*
  - *Used in NEAMS SQAP*
  - *Used to determine GAPS*
- SQE Requirements
  - *Methodology independent*
- Tailorable
  - *To unique needs of each IPSCs and Cross Cut activities*
  - *Revised periodically*



Key Concept 2:
The Risk Assessment Establishes Risk Level Grade

- There are 5 levels of software risk level grades.

Risk Consequence Severity of Failure								
Tier 0	RL1			RL1			RL1	
Tier 1	RL3			RL2			RL2	
Tier 2	RL4			RL3			RL3	
Tier 3	RL5			RL4			RL3	
Tier 4	RL5			RL5			RL4	
		2			**8**			
	Risk Due To Software Environment							

NEAMS using Risk Assessment Tool

Three Levels of Activity Rigor

- Managed:
  - Highest level of rigor or formality. The project has processes and work products, which are reviewable in advance and well documented….
- Documented:
  - Middle level of rigor or formality. Artifacts are purposefully produced to communicate decisions during the course of the project…….
- Understood:
  - Lowest level of rigor or formality. These practices may be implemented either formally or informally…….

Full or Tailored Implementation

- Full implementation of the indicated rigor level is required.

- Tailored implementation of a practice allows for variation in the indicated rigor level.

Examples:

FM (Full Managed)

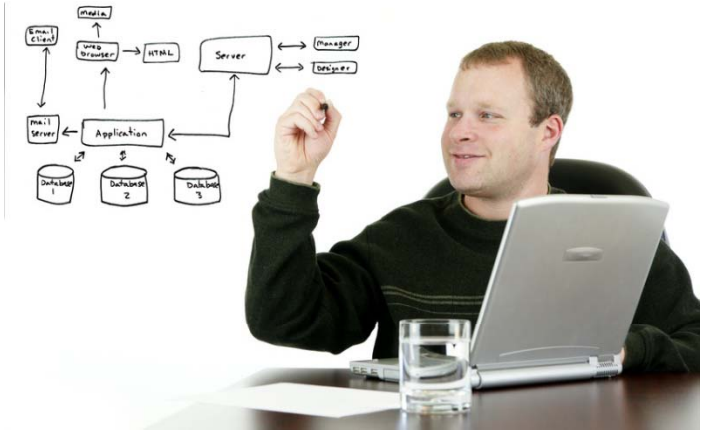
TD (Tailored Documented)

TU (Tailored Understood)

RL5 Optional

Ten Activities (DOE Order 414-1C)

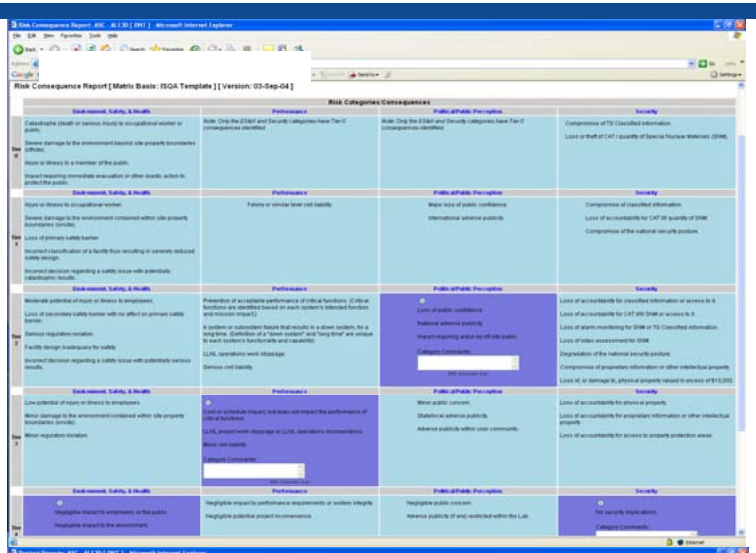
1. Software project management and quality planning;
2. Software risk management;
3. Software configuration management;
4. Procurement and supplier management;
5. Software requirements identification and management;
6. Software design and implementation;
7. Software safety;
8. Verification and validation;
9. Problem reporting and corrective action; and
10. Training of personnel in the design, development, use, and evaluation of safety software.

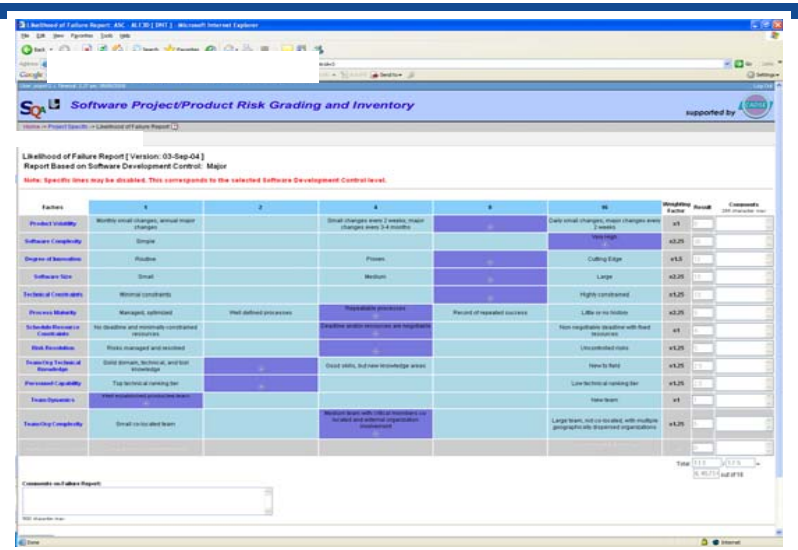


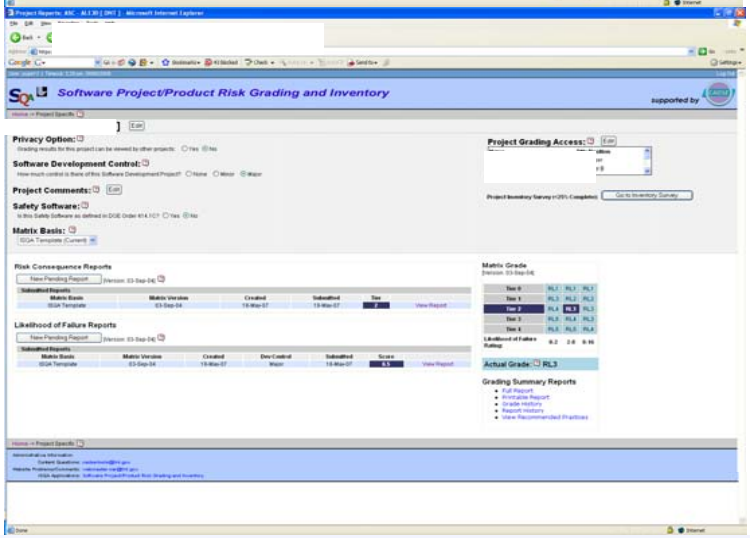
Example SQA Work Activity 1

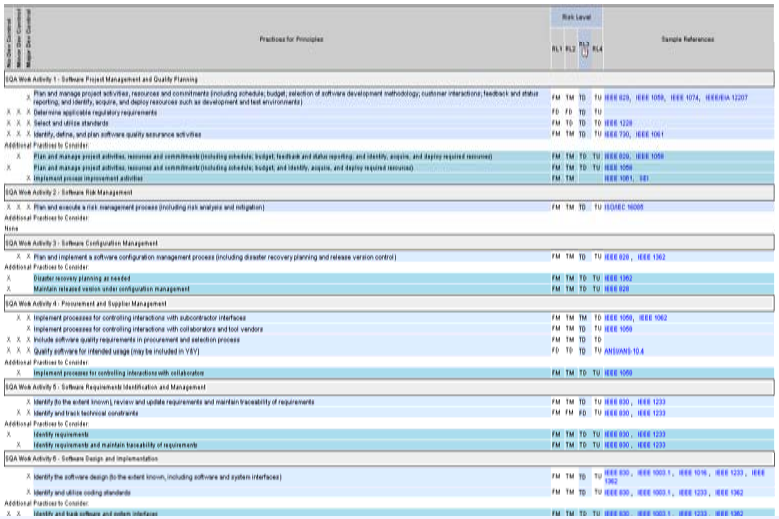
No Dev Control	Minor Dev Control	Major Dev Control	Practices for Work Activities	Risk Level				*Sample References*	*NEAMS Recommended Tools and* Practices
				RL1	RL2	RL3	RL4		See NEAMS website for additional data
			SQA Work Activity 1 - Software Project Management and Quality Planning						
		X	Plan and manage project activities, resources and commitments (including schedule; budget; selection of software development methodology; customer interactions; feedback and status reporting; and identify, acquire, and deploy resources such as development and test environments)	**FM**	**TM**	**TD**	**TU**	*IEEE 829, IEEE 1058, IEEE 1074, IEEE/EIA 12207*	*Team Forge Enterprise Edition, Microsoft Project, Excel, Word,* Agile, Scrum, XP, *IBM Rational Tools,* Unit Test Frameworks, System/Regression Test, Continuous Integration
	X		Plan and manage project activities, resources and commitments (including schedule; budget; feedback and status reporting; and identify, acquire, and deploy required resources)	**FM**	**TM**	**TD**	**TU**	*IEEE 829, IEEE 1058*	*Team Forge Enterprise Edition Microsoft Project, Excel, Word, System/Regression Test Frameworks*
X			Plan and manage project activities, resources and commitments (including schedule, budget, and identify, acquire, and deploy required resources)	**FM**	**TM**	**TD**	**TU**	*IEEE 1058*	Customer Surveys, Supplier Track Record, Acceptance Testing

Key Concept 2:
Risk Assessment Tool







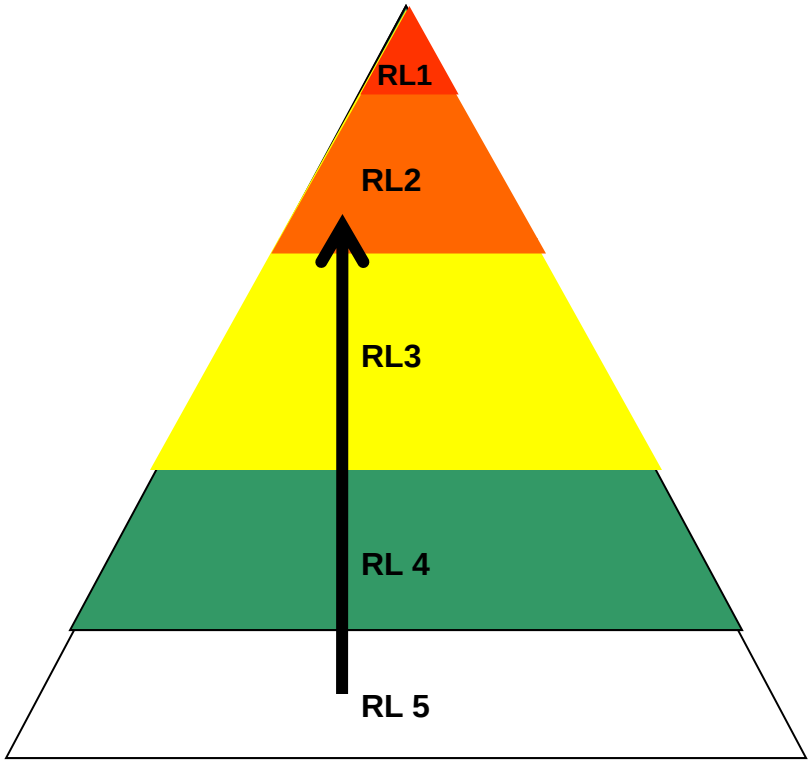


Preliminary Risk Grading of NEAMS Codes

Argonne National Labs NET5000	RL-5
Argonne National Labs Unic	RL-5
ORNL AMP Fuels Decision Aid	RL-2
ORNL AMP Research	RL-3
Safety and Separation Plant Simulation	RL-3
Safety and Separation R&D	RL-3
Waste Forms Disposal System Licensing Decisions	RL-2
Waste Forms Disposal System Decision Tool	RL-3
Waste Forms Disposal System Development	RL-4

Risk Grading Results - Trend

- Risk levels go up over time
- End game, licensing decisions, design trade offs, very disciplined.
- Mid-life crisis to inform decisions, more discipline
- Start as R&D, max agility



Making Codes Better – Risk Mitigation for R&D Codes

- Expert Review
- Static Analysis
- Hazard Analysis
- Code Refactoring
- Testing (positive and negative)
- Code Wrapping
- Recommended prevention techniques and tools
- Recommended detection techniques and tools

Risk Mitigation Using Expert Review

- Simulation results should not be used without expert review:
  - The inputs can be wrong
  - The model can be wrong
  - The users model of the model can be wrong

Risk Mitigation Using Static Analysis

Klocwork Insight review

user: pope12
project: mmeng

Project List | Reports | Issue Management | Source Cross-Reference | About | Help | Logout

< Previous | Return to List | Next >

'itemp' array elements might be used uninitialized in this function.

ID:	1 link to this issue
Issue Name:	Uninitialized Array - possible
Location:	/g/g23/pope12/mmeng/src/angles.c: 79
Severity:	Error (3)
State:	Existing
Status:	Analyze
Comments:	View History...
Last Update:	none

Save Changes

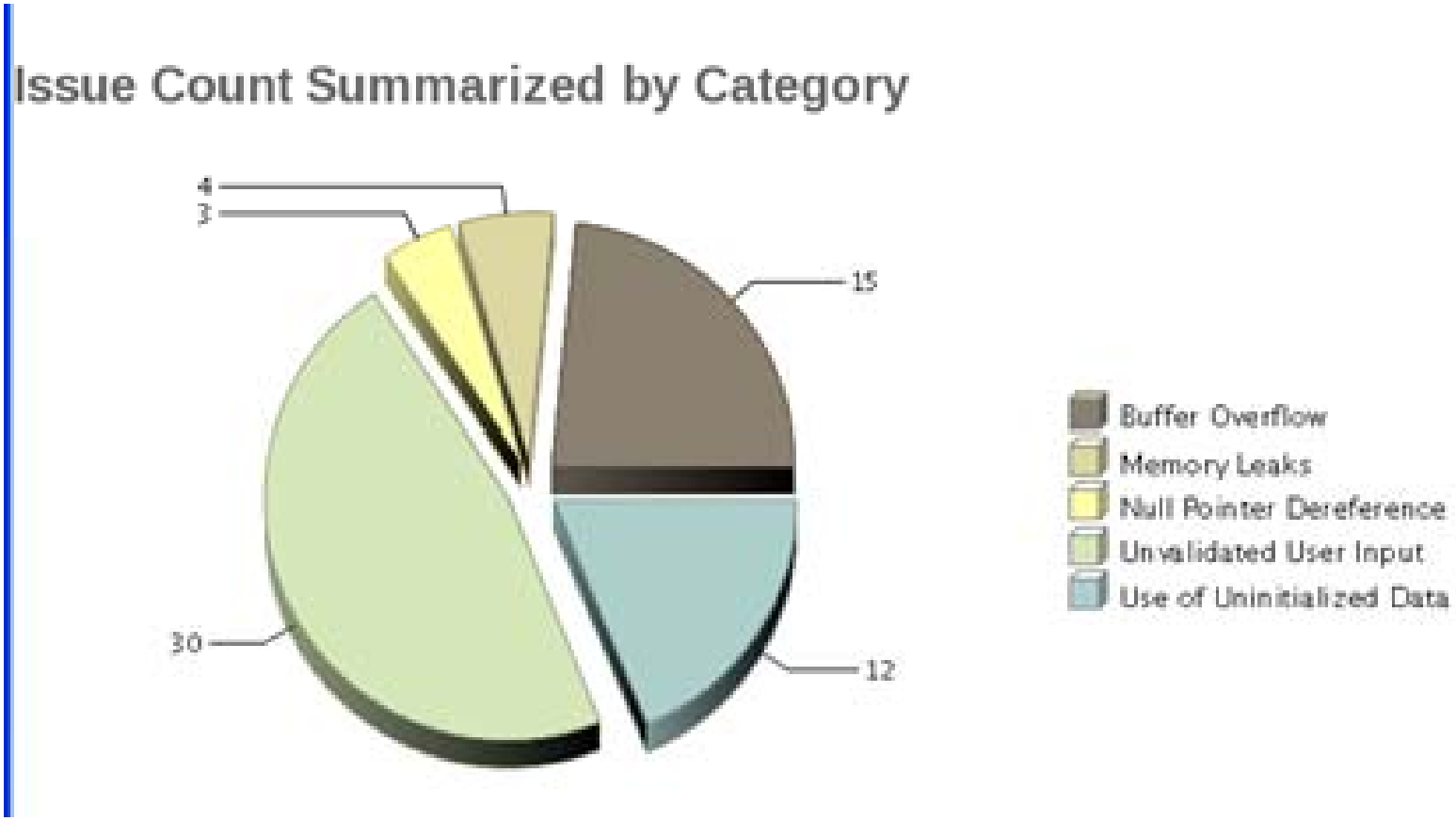
Traceback:

- angles.c:27: i<=natom is true
- angles.c:39: atom[i] .type<300 is false
- angles.c:39: jj==2&&atom[i] .type<300 is false
- angles.c:61: jj==3&&atom[i] .type<300 is true
- angles.c:63: icoord==0 is false
- angles.c:73: atom[i] .bo[j] !=9 is false
- angles.c:73: atom[i] .iat[j] !=0&&atom[i] .bo[j] !=9 is false
- angles.c:79: 'itemp' array elements are used uninitialized.

/g/g23/pope12/mmeng/src/angles.c

```
59          angles.i13[angles.nang][3] = 0;
60          angles.nang++;
61      } else if (jj == 3 && atom[i].type < 300)
62      {
63          if (icoord == 0)
64          {
65           ia = atom[i].iat[0];
66           ib = atom[i].iat[1];
67           ic = atom[i].iat[2];
68          } else
69          {
70            icoord = 0;
71            for(j=0; j < MAXIAT; j++)
72            {
73               if (atom[i].iat[j] !=0 && atom[i].bo[j] != 9)
74               {
75                   itemp[icoord] = atom[i].iat[j];
76                   icoord++;
77               }
78            }
79            ia = itemp[0];
80            ib = itemp[1];
81            ic = itemp[2];
82          }
83          angles.i13[angles.nang][0] = ia;
84          angles.i13[angles.nang][1] = i;
85          angles.i13[angles.nang][2] = ib;
86          angles.i13[angles.nang][3] = ic;
```

Risk Mitigation Using Static Analysis

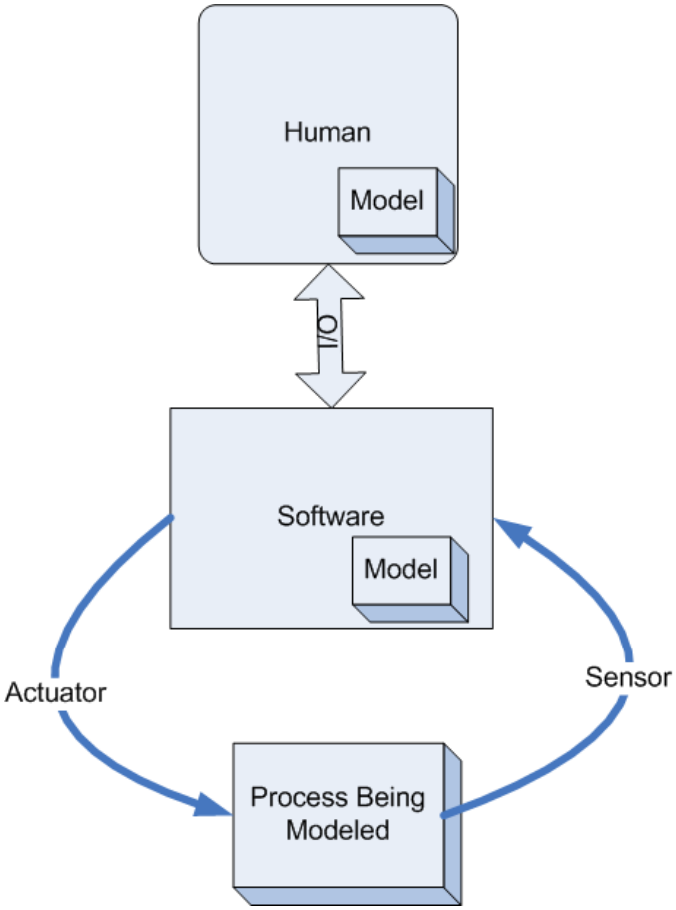


Risk Mitigation Using Hazard Analysis

- For RL 2 codes a Hazard Analysis should be performed on the software system.
- FMEA and Fault Trees developed for mechanical and electrical devices.
- Software fails differently, it does not wear out.
- For instance, software can fail because of bad data or a race condition
- Do an accident investigation before it happens.

Risk Mitigation Using Hazard Analysis

- System Theoretic Process Assessment



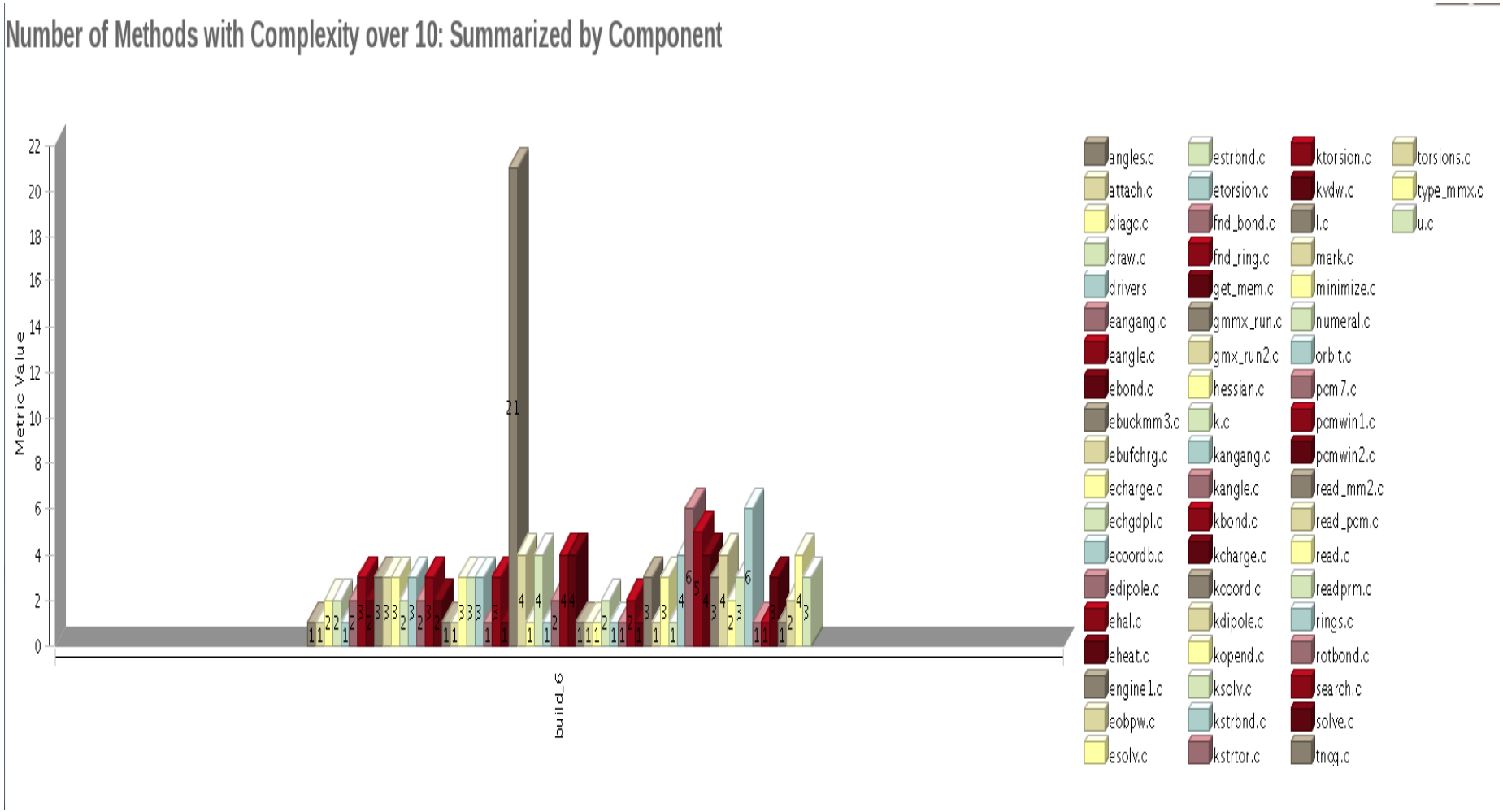
For Each Path Ask What If:
Message not sent
Message incomplete
Message noisy
Message stops prematurely

Nancy Leveson, MIT

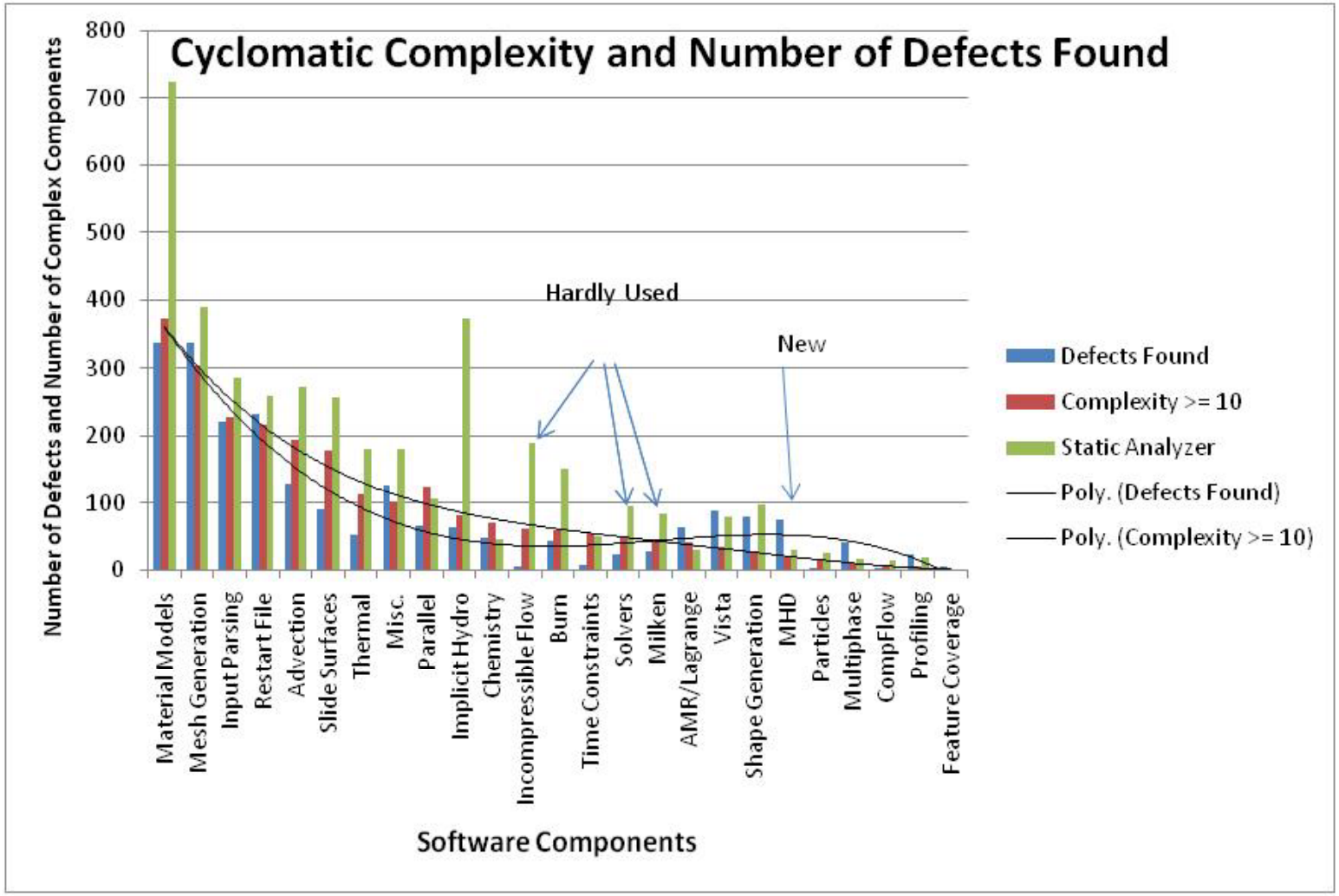
Risk Mitigation Using Code Refactoring

- Add unit tests
- Add assertions
- Pattern Insight, look for cut and paste, similar methods
- Klocwork Insite, extracted functions automated
- Klocwork Architect, Simplify Architecture
- Simplify Complex Code

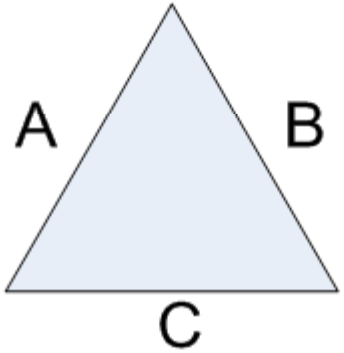
Risk Mitigation by Refactoring

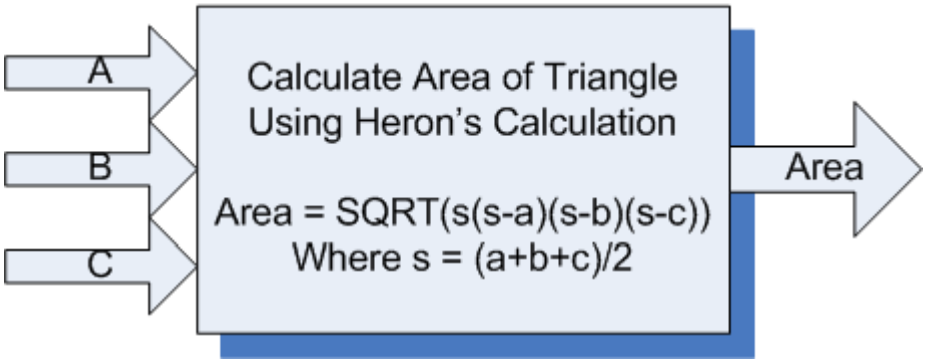


Five Year Study - Complexity and Defects in R&D Codes



Risk Mitigation of R&D Code Using Wrappers and Testing





Risk Mitigation Using Test Cases

Sides of a Triangle	a	b	c	Tri #	Area of Triangle
Try different side lengths in	3	4	17	1	Triangle Inequality is not Satisified (try other lengths)
the cells to the right	3	4	6	2	5.332682252
Notice What happens if you	3	4	7	3	0
enter lengths that do not satisfy	3	4	8	4	Triangle Inequality is not Satisified (try other lengths)
a+b>c, b+c>a, & a+c>a	3	5	8	5	0
The	3	4	2	6	2.90473751
Triangle Inequality Theorem:	2	7	8	7	6.437196595
Which says the sum of any two	14	15	12	8	78.92678569
sides of a triangle must be				9	0
greater than the third side					

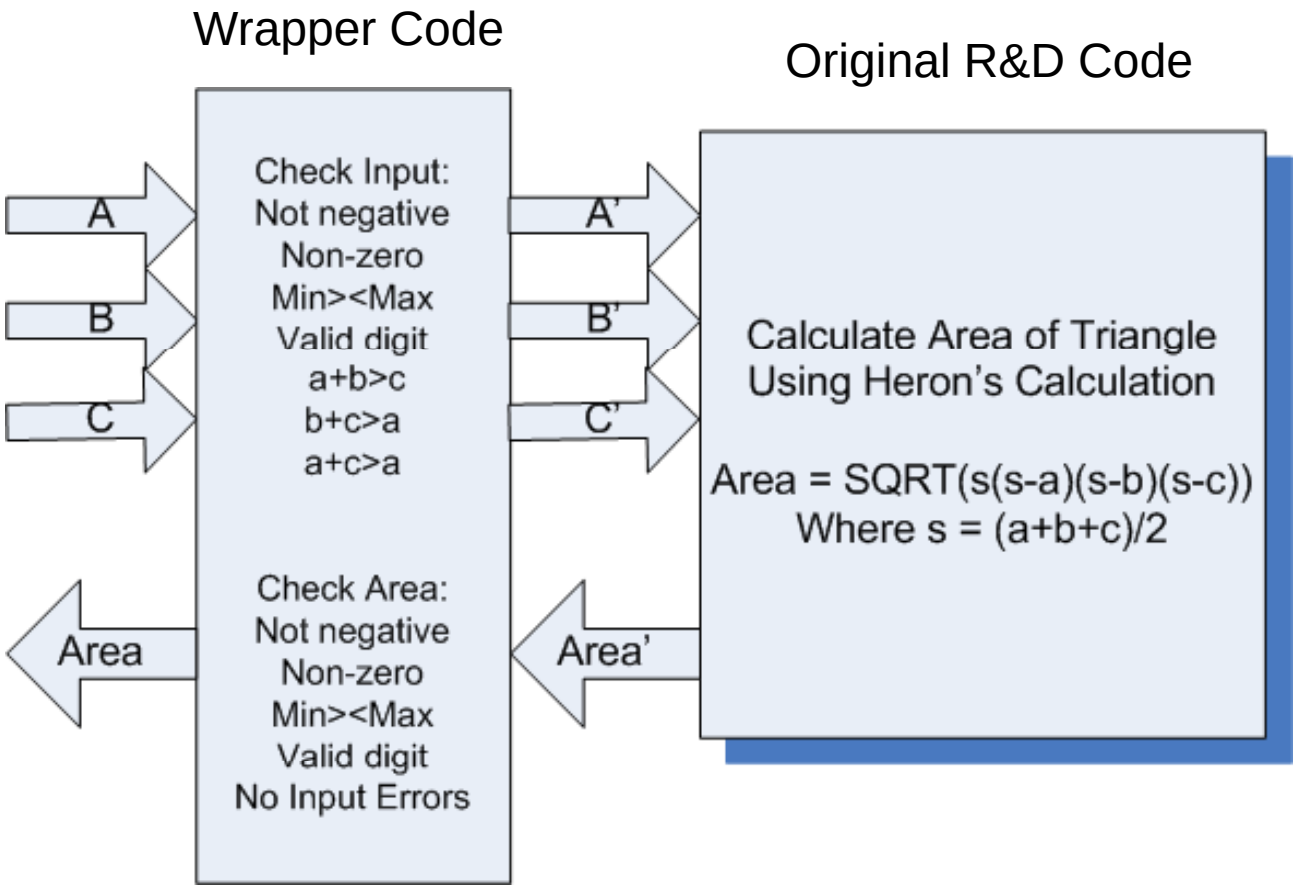
S	S1=	12
	S2=	6.5
	S3=	7
	S4=	7.5
	S5=	8
	S6=	4.5
	S7=	8.5
	S8=	20.5
	S9=	0

jdildine@uiuc.edu

Risk Mitigation Using Negative Test Cases

- One side zero
- Two sides zero
- Three sides zero
- One side negative
- One side smaller than smallest value allowed
- One side larger than largest number allowed
- One side is blank
- One side is null
- One side is CR/LF
- Historical problems …. Etc.

Wrapped Code and Testing Reduces Risk of Use



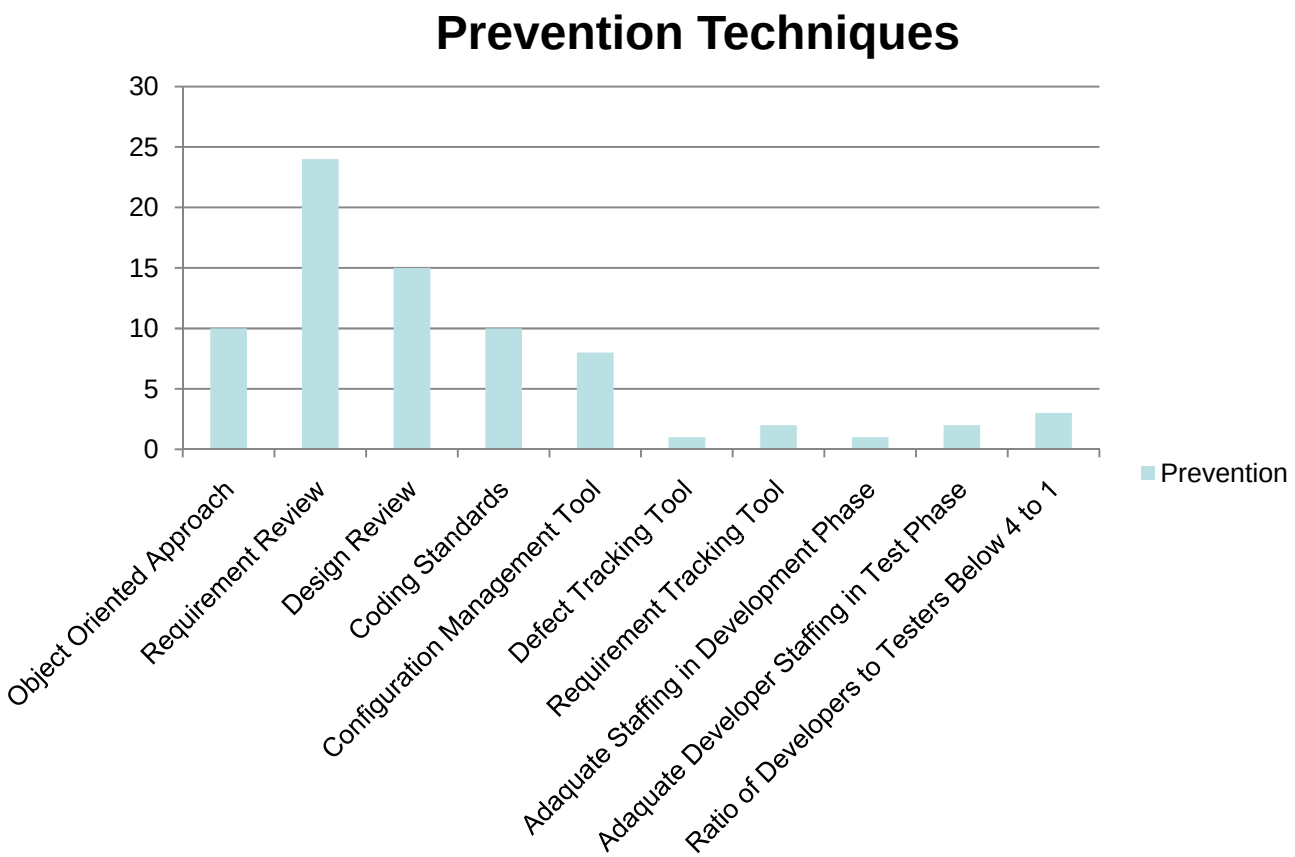
Risk Mitigation Using Recommended Process and Tools

What makes code “Bad”?
Audience Participation.

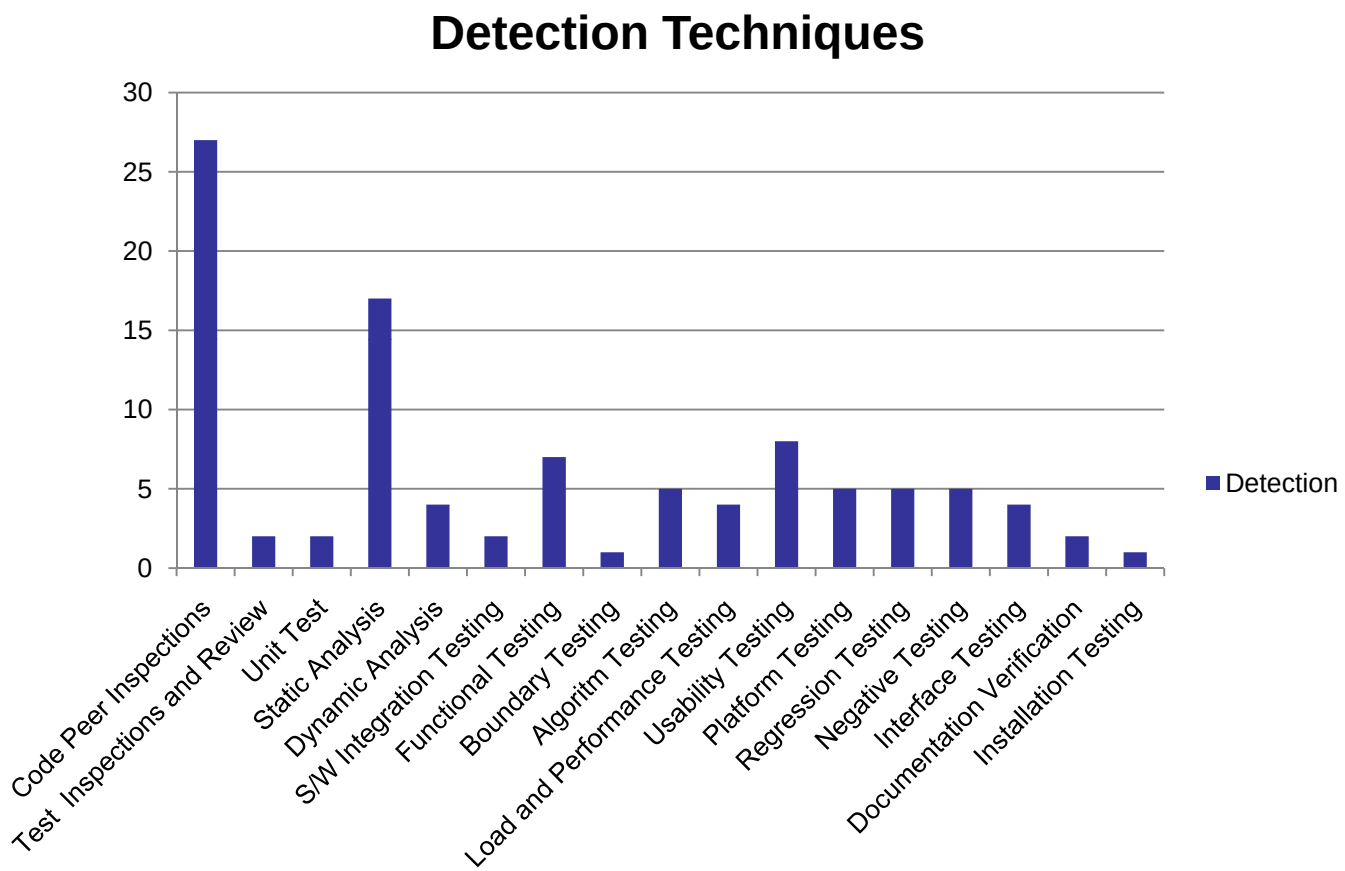
Symtoms	Antidote:	Object Oriented Approach	Requirement Review	Design Review	Coding Standards	Configuration Management Tool	Defect Tracking Tool	Requirement Tracking Tool	Adaquate Staffing in Development Phase	Adaquate Developer Staffing in Test Phase	Ratio of Developers to Testers Below 4 to 1	Code Peer Inspections	Test Inspections and Review	Unit Test	Static Analysis	Dynamic Analysis
Dead code			P	P								D			D	
Exceptions not handled			P	P	P							D			D	
Wrong Verisons of Code/Data						P										
Build errors						P										
Uncontrolled branches or versions						P										
Numerous patches and quick fixes		P														
Complex code		P		P	P							D			D	
traceability documents																
Uninitialized variables												D			D	
Memory Leaks												D			D	D

Perspective	Symtoms	Antidote:	Object Oriented Approach	Requirement Review	Design Review	Coding Standards	Configuration Management Tool	Defect Tracking Tool	Requirement Tracking Tool	Adaquate Staffing in Development Phase	Adaquate Developer Staffing in Test Phase	Ratio of Developers to Testers Below 4 to 1	Code Peer Inspections	Test Inspections and Review	Unit Test	Static Analysis	Dynamic Analysis	S/W Integration Testing	Functional Testing	Boundary Testing	Algorithm Testing	Load and Performance Testing
Developers	Code hard to maintain or understand			P												D						
Developers	Code brittle			P				P								D						
Developers	Logic and Syntax Errors															D		D				
Developers	Internal communication errors			P		P	P									D						
Developers	Lack of time				P						P	P	P						D			
Developers	Large consumer of resources, Ram, or CPU cycles															D						
Testers	Not sure of expected result				P								P			D	D	D				
Testers	Requirements not traced to tests																	D				
Testers	Large defect backlogs											P										
Testers	Insuffient Time to Test															P						
Testers	Fixed bugs reappear							P		P												
Users	Poor Help				P																	
Users	Hard to Use, Not Intuitive, Easy to Make Mistakes																					
Users	User Documentation Confusing				P			P														
Users	No tutorials or self-training				P																	
Users	Application breaks on some platforms/OS/browsers				P											D						
Users	More errors found by customers than by system testing																	D	D	D		

Risk Mitigation Using Prevention Techniques



Risk Mitigation Using Detection Techniques



Conclusion

- Risk Based Graded Approach to Quality Assurance
- NEAMS software risk levels grow over time
- Risk Mitigation applied as R&D codes mature
- Balanced Agility and Discipline using Key Concepts:
  1. Simplified Compliance Flow Down
  2. Risk Assessment Tools
  3. Flow Down to Feeders and Libraries
  4. Required/Actual Gap Analysis
  5. Process Improvements to Close Gaps
  6. Automation and Tools Support
  7. Embedded SQE's and Independent Reporting Organization
  8. Shared Best Practices Approach
  9. Natural Metrics to Prioritize Improvements