Responding to the Software Crisis in DOE Scientific Computing

September 14, 2015

DOE, Germantown, MD.

Gregory Pope CSQE

LLNL-PRES-676991

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

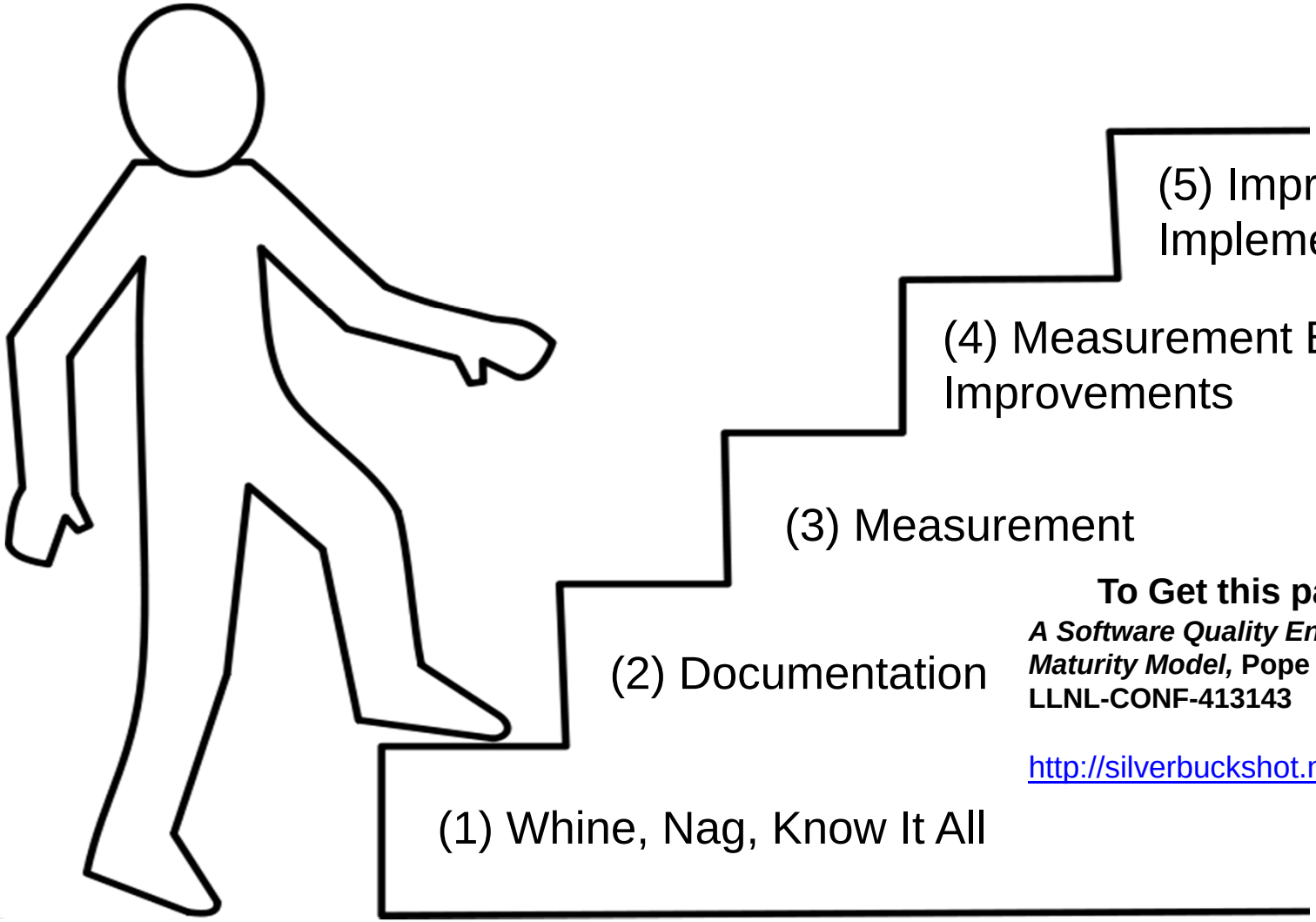
Lawrence Livermore National Laboratory

Topics

1. ASC Program Software Engineering Lessons Learned at LLNL
2. Modeling Code Reliability Requirements for Exascale
3. Study Group Results
4. Next Steps, Risk Management

ASC Program Lessons Learned at LLNL

- Software Engineering progress came in stages:



To Get this paper:

***A Software Quality Engineering Maturity Model,* Pope and Hill**
LLNL-CONF-413143

http://silverbuckshot.net/SQE-Model

ASC Program Lessons Learned at LLNL - Level 1

- Top Down Theory X management style is less effective with Knowledge Workers

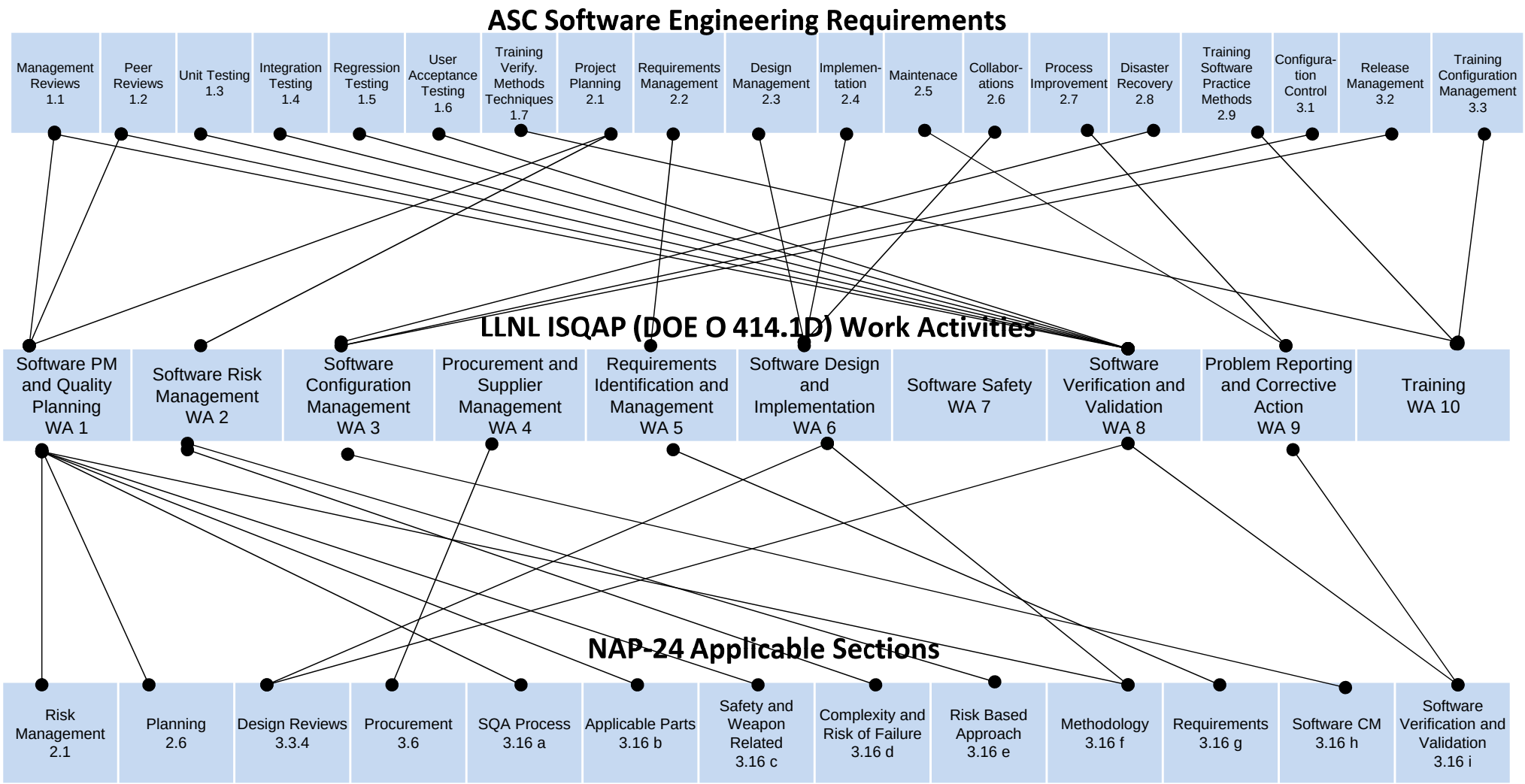
McGregor X - Y Theories

Theory X	Theory Y
* people need close supervision	* people want independence in work
* will avoid work when possible	* people seek responsibility
* will avoid responsibility	* people are motivated by self-fulfilment
* that they desire only money	* people naturally want to work
* people must be pushed to perform	* people will drive themselves to perform

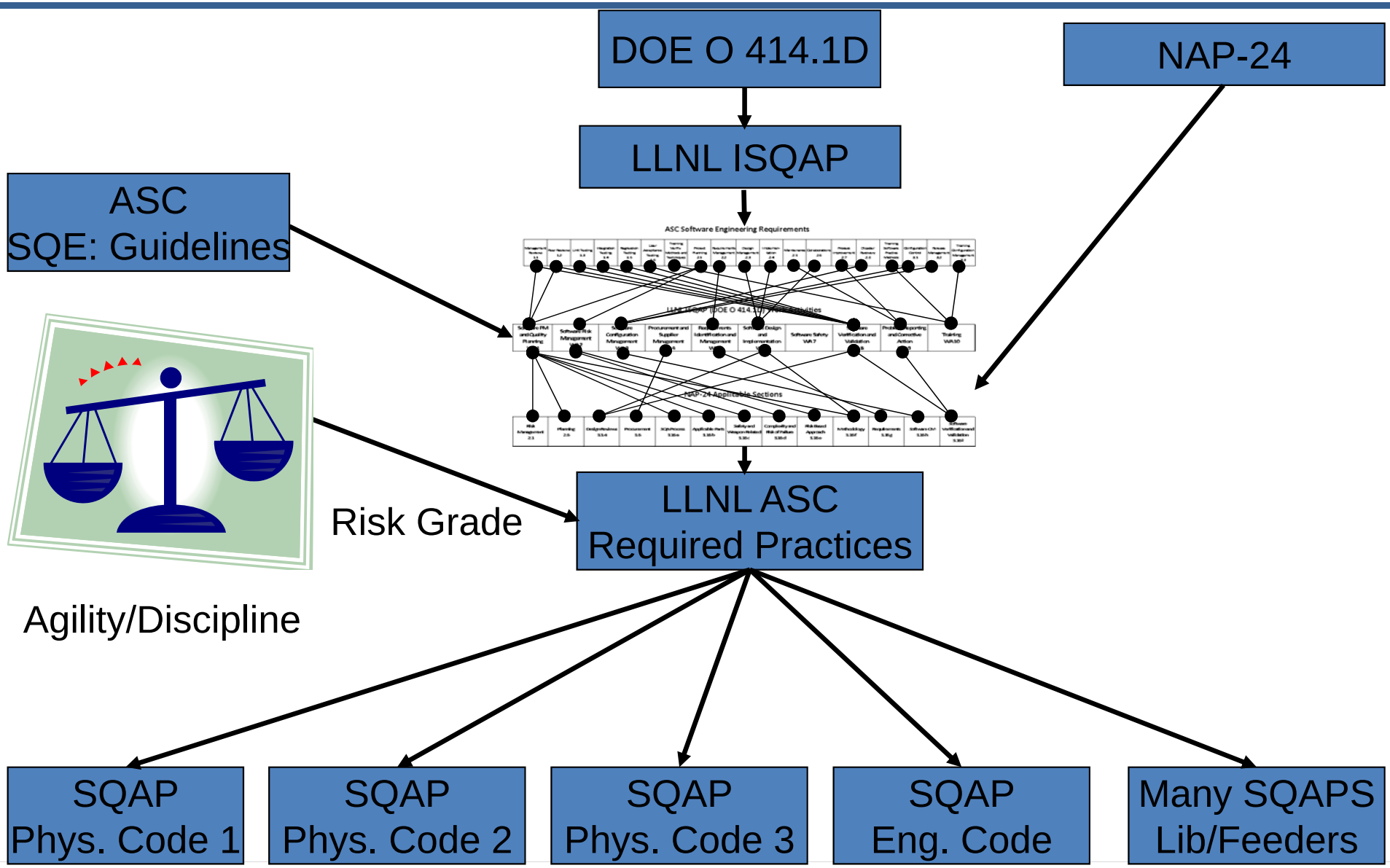
ASC Program Lessons Learned at LLNL – Level 1

- Whining, Nagging, Know It All - is ineffective with knowledge workers

ASC Program Lessons Learned at LLNL
DOE/NNSA Compliance is Confusing – Level 2



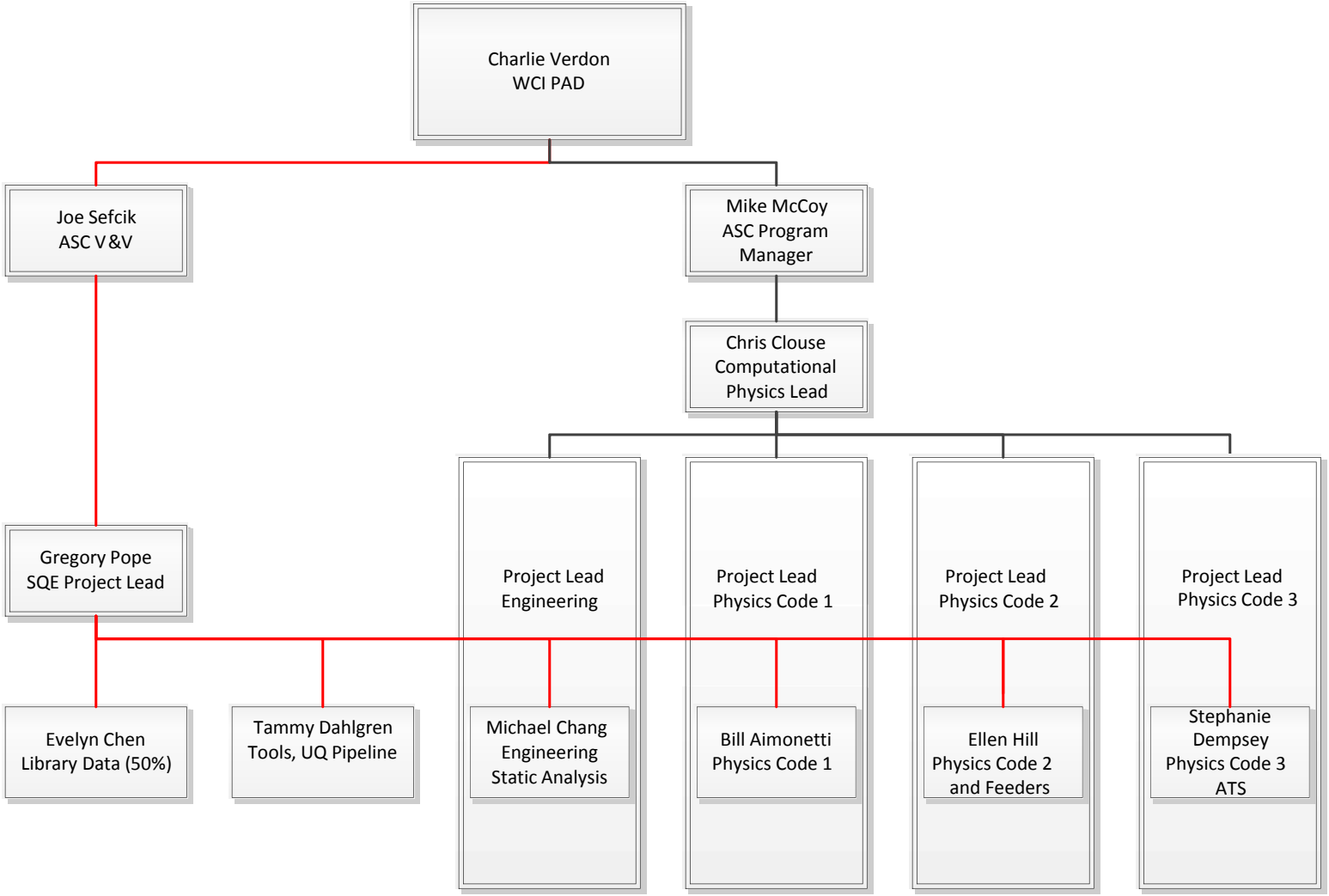
ASC Program Lessons Learned at LLNL
Compliance Flow Down and Simplification – Level 2



ASC Program Lessons Learned at LLNL
ASC Program SE Lessons Learned at LLNL – Level 3

- Group of dedicated Software Quality Engineers (SQE's).
- Embedded SQE's into the development teams.
- Make compliance easy and automated.
- Find and roll out tools and processes.
- SQE's most effective if developers themselves.
- Now being requested by code teams to focus on automated building, testing, release, static and dynamic code analysis, tool research.

ASC Program Lessons Learned at LLNL
Dual Reporting Chain – Level 3



ASC Program Lessons Learned at LLNL
Measuring Things – Level 3

Major Codes	Lead	SQE	Req	Des	Imple	Oper	Maint	Retire	SCM	V&V	PM		Overall
		RG	6	6	9	8	8	n/a	8	8	6		7.4
		JH	5	5	8	8	8	n/a	7	8	6		6.9
		NK	7	7	8	8	8	n/a	8	8	6		7.5
		BO	6	3	6	8	8	n/a	6	6	4		5.9
		BO	6	4	8	8	8		8	7	6		6.9
		EH	5	4	8	7	7	n/a	7	7	6		6.4
		SA							7	6			6.5
		EH	4	2	6	8	8	n/a	5	6	4		5.4
		EH											
		SA	4	2	5	7	7	n/a	6	7	3		5.1
		BO	6	6	8	8	7	n/a	7	7	6		6.9
		BO	6	6	8	8	7	n/a	7	7	7		7.0
		BO											
		BO											
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		EH	6	6	6	7	7	n/a	8	7	6		6.6
		BO											
		EH											
		BO											
		BO											
		BO											
		EH											
		BO											
		BO											
		BO											
		BO											
		BO											
		RG	5	3	8	8	9	n/a	6	6	4		6.1
		EH											
		BO											
AVERAGES			5.0	3.7	6.6	7.5	7.4	n/a	6.6	6.3	4.6		6.0

ASC Program Lessons Learned at LLNL
Improve Things – Level 4

2008 to 2015

Major Codes	Lead	SQE	Req	Des	Imple	Oper	Maint	Retire	SCM	V&V	PM		Overall
		RG	6	6	9	8	8	n/a	8	8	6		7.4
		JH	5	5	8	8	8	n/a	7	8	6		6.9
		NK	7	7	8	8	8	n/a	8	8	6		7.5
		BO	6	3	6	8	8	n/a	6	6	4		5.9
		BO	6	4	8	8	8		8	7	6		6.9
		EH	5	4	8	7	7	n/a	7	7	6		6.4
		SA							7	6			6.5
		EH	4	2	6	8	8	n/a	5	6	4		5.4
		EH											
		SA	4	2	5	7	7	n/a	6	7	3		5.1
		BO	6	6	8	8	7	n/a	7	7	6		6.9
		BO	6	6	8	8	7	n/a	7	7	7		7.0
		BO											
		BO											
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		SA	4	2	5	7	7	n/a	6	5	3		4.9
		EH	6	6	6	7	7	n/a	8	7	6		6.6
		BO											
		EH											
		BO											
		BO											
		BO											
		EH											
		BO											
		BO											
		BO											
		BO											
		BO											
		RG	5	3	8	8	9	n/a	6	6	4		6.1
		EH											
		BO											
AVERAGES			5.0	3.7	6.6	7.5	7.4	n/a	6.6	6.3	4.6		6.0

Major Codes

Feeder Codes

Major Codes	Lead	SQE	Req	Des	plementati	Operationa	Maint	Retire	SCM	V&V	PM		Overall
		SD	9	9	9	10	10	n/a	10	10	10		9.6
		EH	10	10	10	10	10	n/a	10	10	10		10.0
		NK	10	10	10	10	10	n/a	10	10	10		10.0
		BO	9	6	9	10	10	n/a	10	10	7		8.9
		BO											
		BO											
		BO	6	6	8	8	8		8	7	6		7.1
		EH	7	6	8	7	7	n/a	9	7	6		7.1
		EH	7	6	8	8	8		7	6	8		7.3
		EH											
		EH	9	7	9	9	9	n/a	10	10	10		9.1
		EH	9	7	9	9	9	n/a	10	10	10		9.1
		EH	9	7	9	9	9	n/a	10	10	10		9.1
		EH	9	7	9	9	9	n/a	10	10	10		9.1
		EH	9	7	9	9	9	n/a	10	10	10		9.1
		EH	5	5	6	9	9	n/a	10	10	10		8.0
		EH	8	8	9	9	9	n/a	10	9	9		8.9
		BO	4	4	5	7	7	n/a	6	7	5		5.6
		BO	7	6	9	9	8	n/a	10	10	6		8.1
		BO	7	6	9	9	8	n/a	10	10	7		8.3
		BO	7	7	9	9	9		8	7	7		7.9
		BO	7	7	9	9	9		8	7	7		7.9
		GP	7	6	5	7	7	n/a	5	6	4		5.9
		GP	7	6	5	7	7	n/a	5	6	4		5.9
		GP	7	6	5	7	7	n/a	5	6	4		5.9
		GP	7	6	5	7	7	n/a	5	6	4		5.9
		GP	7	6	5	7	7	n/a	5	6	4		5.9
		EH	10	7	9	7	7	n/a	10	10	9		8.6
		EH											
		EH	8	8	9	10	10	n/a	10	10	8		9.1
		JH	6	6	8	8	8		8	7	6		7.1
		BO											
		EH											
		JH											
		JH											
		BO	7	6	9	9	8	n/a	10	10	7		8.3
		BO	7	6	8	8	9		9	8	7		7.8
		SD	9	5	8	8	9	n/a	10	10	10		8.6
		EH	9	9	10	10	10	n/a	10	10	9		9.6
AVERAGES			7.7	6.6	8.0	8.5	8.4	n/a	8.6	8.5	7.5		8.0

ASC Program Lessons Learned at LLNL Implement Improvements – Level 5

- Cannot check in code without passing smoke test.
- Cannot run smoke test without a peer review check.
- Dedicated System Testing resources.
- Distributed CM tools like STASH/GIT to product main branch.
- Design trade offs embedded into code – e.g. Doxygen.
- Requirements tracked in tracking and linkage tools – e.g. JIRA.
- Acquisition and maintenance automated testing tools – e.g. ATS.
- Nightly or CI automated test reports/DB, visualization.
- Static and Dynamic Code Analysis.
- Create records as artifacts of Value Added Tools.

Modeling Code Reliability Requirements

Application Domain	Number Projects	Error Range (Errors/KESLOC)	Normative Error Rate (Errors/KESLOC)	Notes
Automation	55	2 to 8	5	Factory automation
Banking	30	3 to 10	6	Loan processing, ATM
Command & Control	45	0.5 to 5	1	Command centers
Data Processing	35	2 to 14	8	DB-intensive systems
Environment/Tools	75	5 to 12	8	CASE, compilers, etc.
Military -All	125	0.2 to 3	< 1.0	See subcategories
§ Airborne	40	0.2 to 1.3	0.5	Embedded sensors
§ Ground	52	0.5 to 4	0.8	Combat center
§ Missile	15	0.3 to 1.5	0.5	GNC system
§ Space	18	0.2 to 0.8	0.4	Attitude control system
Scientific	35	0.9 to 5	2	Seismic processing
Telecommunications	50	3 to 12	6	Digital switches
Test	35	3 to 15	7	Test equipment, devices
Trainers/Simulations	25	2 to 11	6	Virtual reality simulator
Web Business	65	4 to 18	11	Client/server sites
Other	25	2 to 15	7	All others

Reifer, Donald, “Industry Software Cost, Quality, and Productivity Benchmarks,”

DoD Software Tech News, June 2004, http://www.softwaretechnews.com/stn7-2/reifer.html.

Tracking and Comparing Fault Density on ASC

Physics Code	Estimated	Measured
A	3.0	2.5
B	3.5	1.6
C	2.0	.91

Application Domain	Number Projects	Error Range (Errors/KESLOC)	Normative Error Rate (Errors/KESLOC)	Notes
Scientific	35	0.9 to 5	2	Seismic processing

So how long before a .91 defects/KSLOC fails?

Converting Defect Density (.91) to Failure Rate Single 1.6GHz Processor (12 GFLOPS), 578K SLOCs

Single 1.6GHz Processor (12.8 GFLOPS), 578K SLOCs

Hours	Years	Probability of Failure Free			Probability of Failure	
1		99.99%			0.01%	
3		99.97%			0.03%	
8		99.91%			0.09%	
160		98.24%			1.76%	
480		94.81%			5.19%	
1000		89.48%			10.52%	
5000	0.6	57.37%			42.63%	
10000	1.1	32.91%			67.09%	
50000	5.7	0.39%			99.61%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 99.91%

Encyclopedia of Software Engineering, William W. Everett, John D. Musa

Get the Article: http://onlinelibrary.wiley.com/doi/10.1002/0471028959.sof327/abstract

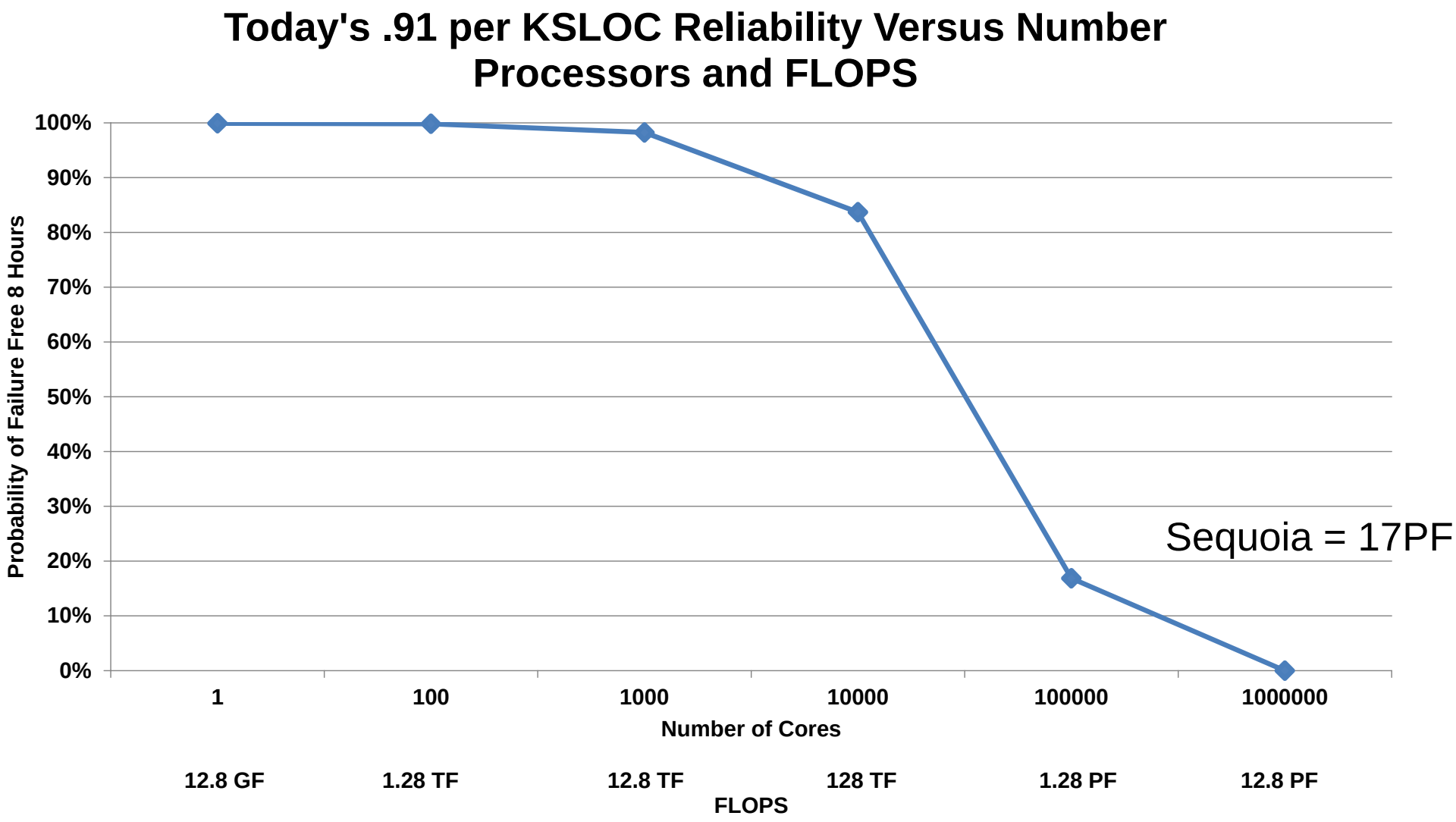
Converting Defect Density to Failure Rate 100 Processor Case

100 1.6GHz Processor (1.28 TFLOPS), 20K SLOCs

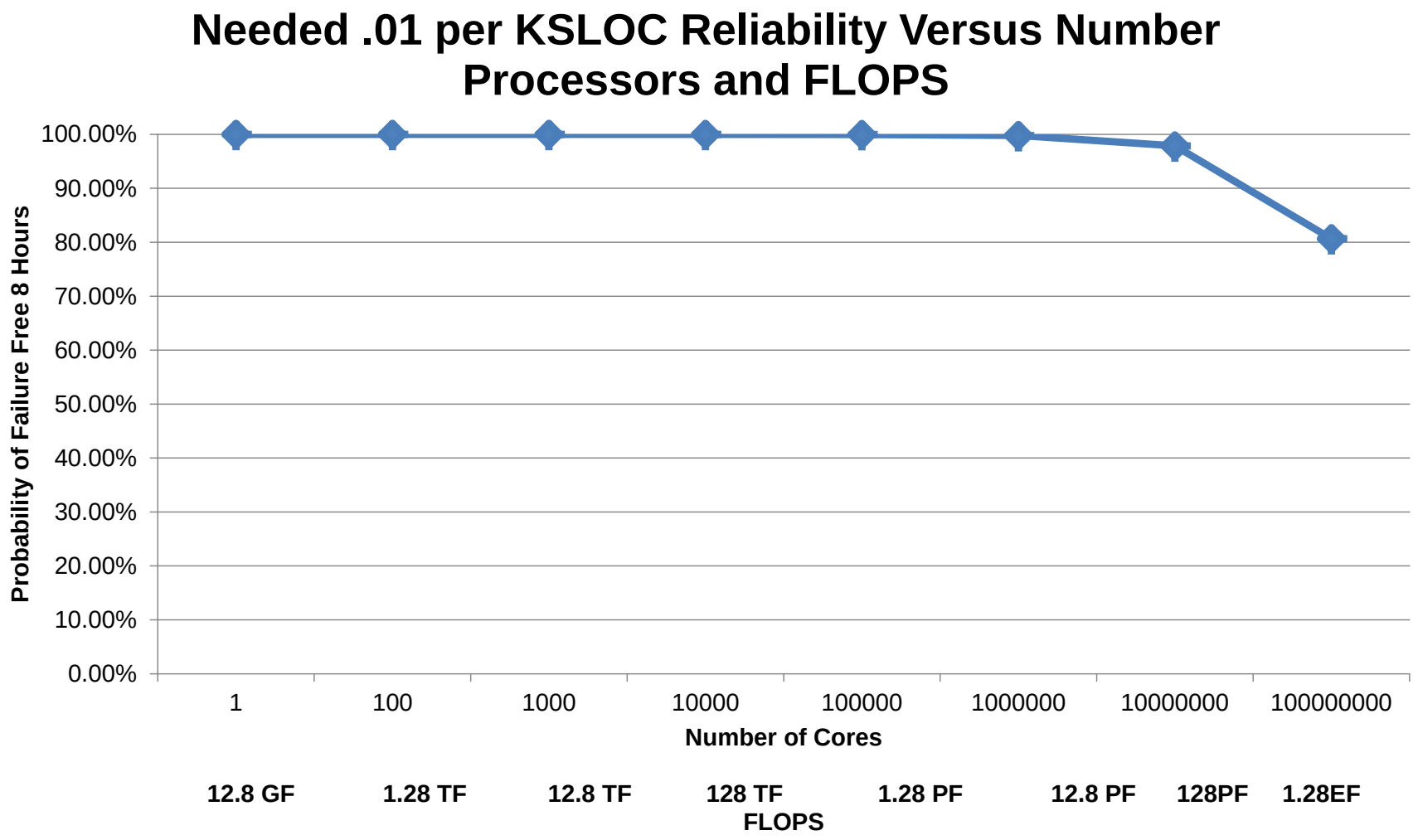
Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	100.00%				
0.016666667	Minutes	100.00%				
1		99.98%			0.02%	
3		99.93%			0.07%	
8		99.82%			0.18%	
160		96.50%			3.50%	
480		89.87%			10.13%	
1000		80.05%			19.95%	
5000	0.6	32.87%			67.13%	
10000	1.1	10.80%			89.20%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 99.82%

Failure Rate Versus Speed (Today)



Failure Rate Versus Speed (Future)



Some Thoughts

- Need code 10 times better than space shuttle.
- Assuming when we find bugs we fix them.
- Then we must find and fix bugs up to 100 times faster.
- We must improve our abilities to prevent, find, and fix bugs or have short simulations.
- Blue Gene/Q (Sequoia) is 17 PFLOPS Rmax, we have not come close to reaching capacity.

Study Group Ground Rules

- The technique used was to solicit elevator speeches: a short and concise write up done as if the author was a speaker with only a few minutes to convince a decision maker of their top issues.

Study Group Contributors: Multiple Labs, Academia, DOE.

Tom McAbee LLNL
Patty Loo INL
Andy Salinger, SNL
Rich Hornung LLNL
Jeffery Carver, University of Alabama
Rob Neely LLNL
Robert Blyth, DOE-ID
David E. Bernholdt ORNL
Roscoe A. Bartlett, ORNL
Michael Heroux, SNL
Jim Willenbring, SNL

Tamara Dahlgren LLNL
Ellen Hill LLNL
Greg Pope LLNL
Burl Hall, LLNL
David Jefferson LLNL,
Stephanie Dempsey LLNL
Tom Epperly, LLNL
Mark Miller LLNL
Jeff Keasler LLNL

To Get the Report: http://silverbuckshot.net/study-group

Study Group Highest Results (2.8-2.5)

- Using Appropriate Software Engineering Practices.
- Upping Perceived Values of Software Engineering.
- Adequate Testing Resources.
- Use of Graded Approach.
- Computational science community needs more awareness, knowledge, understanding, and experience with good software engineering practices for scientific applications.

Nest Steps Risk Management

- RISK: Development in hero mode and code correctness/maintainability suffer.
- RISK: Best practices not communicated.
- RISK: Testing treated as an afterthought.
- RISK: Level of rigor too high or too low.
- RISK: Contemporary tools not available.

MITAGATION: Adopt and follow a team based software development model.

MITAGATION: Hold Best Practices meetings, exclude management.

MITAGATION: Plan for robust testing environment, separate from release and development.

MITAGATION: Use Risk grading tool, compliance automated with tools.

MITAGATION: Target use of open source and COTS tools.

Conclusion

- Good Software Engineering requires an organization and dedicated resources, it does not happen without effort.
- We will need to get much better at Software Engineering practices for scientific codes (Error prevention, detection, and removal).
- Our DOE complex developers know this.
- Manage today's concerns as risk mitigation, learning and improving as we go.

Back Up Slides

Converting Defect Density to Failure Rate 1,000 Processor Case

1000 1.6GHz Processor (12.8 TFLOPS), 20K SLOCs

Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	100.00%				
0.016666667	Minutes	100.00%				
1		99.78%			0.22%	
3		99.33%			0.67%	
8		98.24%			1.76%	
160		70.05%			29.95%	
480		34.37%			65.63%	
1000		10.80%			89.20%	
5000	0.6	0.00%			100.00%	
10000	1.1	0.00%			100.00%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 98.24%

Converting Defect Density to Failure Rate 10,000 Processor Case

10000 1.6GHz Processor (128 TFLOPS), 20K SLOCs

Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	100.00%				
0.016666667	Minutes	99.96%				
1		97.80%			2.20%	
3		93.54%			6.46%	
8		83.69%			16.31%	
160		2.84%			97.16%	
480		0.00%			100.00%	
1000		0.00%			100.00%	
5000	0.6	0.00%			100.00%	
10000	1.1	0.00%			100.00%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 83.69%

Converting Defect Density to Failure Rate 100,000 Processor Case

100000 1.6GHz Processor (1.28 PFLOPS), 20K SLOCs

Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	99.99%				
0.016666667	Minutes	99.63%				
1		80.05%			19.95%	
3		51.30%			48.70%	
8		16.86%			83.14%	
160		0.00%			100.00%	
480		0.00%			100.00%	
1000		0.00%			100.00%	
5000	0.6	0.00%			100.00%	
10000	1.1	0.00%			100.00%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 16.86%

1000000 1.6GHz Processor (12.8 PFLOPS), 20K SLOCs

Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	99.94%				
0.016666667	Minutes	96.36%				
1		10.80%			89.20%	
3		0.13%			99.87%	
8		0.00%			100.00%	
160		0.00%			100.00%	
480		0.00%			100.00%	
1000		0.00%			100.00%	
5000	0.6	0.00%			100.00%	
10000	1.1	0.00%			100.00%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Probability of Failure Free in 8 hours = 0%

Achieving Todays Failure Free Rates on Tomorrows Computers

So what defect rate would be necessary for >90% failure free for 8 hours?

Hours	Years	Probability of Failure Free			Probability of Failure	
0.000277778	Seconds	100.00%				
0.016666667	Minutes	99.99%				
1		99.33%			0.67%	
3		98.00%			2.00%	
8		94.77%			5.23%	
160		34.14%			65.86%	
480		3.98%			96.02%	
1000		0.12%			99.88%	
5000	0.6	0.00%			100.00%	
10000	1.1	0.00%			100.00%	
50000	5.7	0.00%			100.00%	
100000	11.4	0.00%			100.00%	

Fault Density of .05 gives 94.77% failure free for 8 hours
18.2 times better than our best scientific research code
Half the defect rate of the Space Shuttle code

Study Group Results (2.4-2.3)

- HPC Software Engineering Centers
- Tools for Automated Processes
- Staffing, Hiring, and Retention
- Standards Support
- Management of Advanced Code Projects
- Simulation Ensembles

Study Group Results (2.2-2.0)

- Schedule Expectations
- Maintaining a large scientific software use on multiple architectures
- Data, Libraries, Operating Systems
- COTS Tools
- Onboard SQA
- Data and Algorithm Organization ion Physics Code
- Lean/Agile Lifecycle Model for Research Driven CSE/HPC software
- Visual Debugging

Study Group Results (1.9-1.8)

- Fine Grained Parallelism Challenges
- Managing Ever Increasing Need for Compliance Driven Agile Development
- Compliance to DOE Standards
- Thread Rescoping Tool
- Attributes of a highly Effective Software Environment

Study Group Ranking

					Management:
22	2.2	2	8		Schedule Expectation:[1]
24	2.4	5	4	1	**Staffing:**[2] **Hiring and Retention of Staff:**[3]
19	1.9	2	5	3	Managing the Ever-Increasing Need for Compliance-Driven Agile Development:[4]
23	2.3	4	5	1	Management of Advanced Code Projects:[5]
					Process:
27	2.7	7	3		Graded Approach:[6]
28	2.8	9		1	Use of Appropriate Software Engineering Practices[7]
28	2.8	8	2		Upping the Perceived Value of Software Engineering:[8]
16	1.8	3	1	5	Attributes of a highly effective software environment:[9]
28	2.5	7	3	1	The computational science community needs more awareness, knowledge, understanding, and experience with best practices in software engineering: [10]
20	2.0	3	4	3	Lean/Agile lifecycle model for research-driven CSE/HPC software. [11]
24	2.4	5	4	1	HPC Software Engineering Centers:[12]
19	1.9	4	1	5	Compliance to DOE Standards:[13]
23	2.3	4	5	1	Standards support:[14]
					Tools:
21	2.1	2	7	1	COTS Tools:[15]
21	2.1	3	5	2	On Board SQA:[16]
22	2.2	5	2	3	Maintaining a large scientific software base on multiple architectures (RAJA)[17]
24	2.4	5	4	1	Tools for Automating Processes[18]
23	2.3	4	5	1	Simulation Ensembles:[19]
18	1.8	2	4	4	Thread Rescoping Tool[20]
19	1.9	3	3	4	Fine-grained parallelism challenges:[21]
20	2.0	2	6	2	Visual Debugging:[22]
27	2.7	8	1	1	Testing Resources[23]
					Data:
22	2.2	2	8		Data, Libraries, and Operating Systems:[24]
					Libraries[25}
					Operating Systems[26]
17	2.1	3	3	2	Data and algorithm organization in physics codes:[25]

Next Steps

- RISK: Schedule overly optimistic
- RISK: Inability to attract CS top talent
- RISK: Attrition of existing employees
- RISK: Standards and compliance overly restrictive
- RISK: Lack of SE experience
- RISK Best practices not communicated
- RISK: Contemporary tools not available
- RISK: Dev approach not agile enough
- RISK: Level of rigor too high or too low
- RISK: Development in hero mode and code correctness/maintainability suffer.

Risk Management

- RISK: Funding for hardening, productizing, and maintaining tools Is not available.
- RISK: Compliance to standard will be cumbersome
- RISK: Latest compilers, Libraries and operating systems not available or supported
- RISK: Not able to take advantage of main stream productivity tools
- RISK: Insufficient resources to support user needs
- RISK: Manual recoding adds defects to reliable codes
- RISK: Not testing all platform types supported
- RISK: Not having sufficient simulation result management tools
- RISK: Code modification for porting causes decrease in code reliability

Risk Management

- RISK: Not having a quick way to identify anomalies at scale decreases productivity
- RISK: Testing treated as an afterthought
- RISK: Development environment overly specialized or expensive
- RISK: Architecture sprawls and is not optimized