SAND2007-7714P

On the Intersection of V&V and SQE

Timothy Trucano*
Optimization and Uncertainty Estimation, 1411
Martin Pilch
QMU and Management Support, 1221
William Oberkampf
Validation and Uncertainty Processes, 1544

Sandia National Laboratories
Albuquerque, NM 87185

***Phone: 844-8812, FAX: 284-0154**
***Email: tgtruca@sandia.gov**

Sandia Software Engineering Seminar

November 28, 2007

Abstract

"In this talk I will discuss the role of software engineering in performing verification and validation of high performance computational science software. I emphasize three areas where the intersection of software engineering methodologies and the goals of computational science verification and validation clearly overlap. These areas are: (1) the evidence of verification and validation that adherence to formal software engineering methodologies provides; (2) testing; and (3) qualification, or acceptance, of software. In each case I will discuss some challenges and opportunities presented by consideration of the SQE/V&V overlap. I will conclude by emphasizing two major computational science V&V concerns that fall outside the domain of SQE, that is, experimental validation and solution verification. (See SAND2005-3662P, on the CCIM Pubs page.)"

I'm going to play a little fast and loose with this abstract, and talk about what is currently on my mind:

(1) "V&V evidence" → "reliability;"

(2) "testing" → "testing;"

(3) "qualification" → "life cycle."

Verification and Validation (V&V) Definitions

Verification: Are the equations solved correctly? (Math)

Validation: Are the equations correct? (Physics)

ASC:

- Verification: The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- Validation: The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

V&V targets applications of codes, not codes.

V&V is an evidence-based activity.

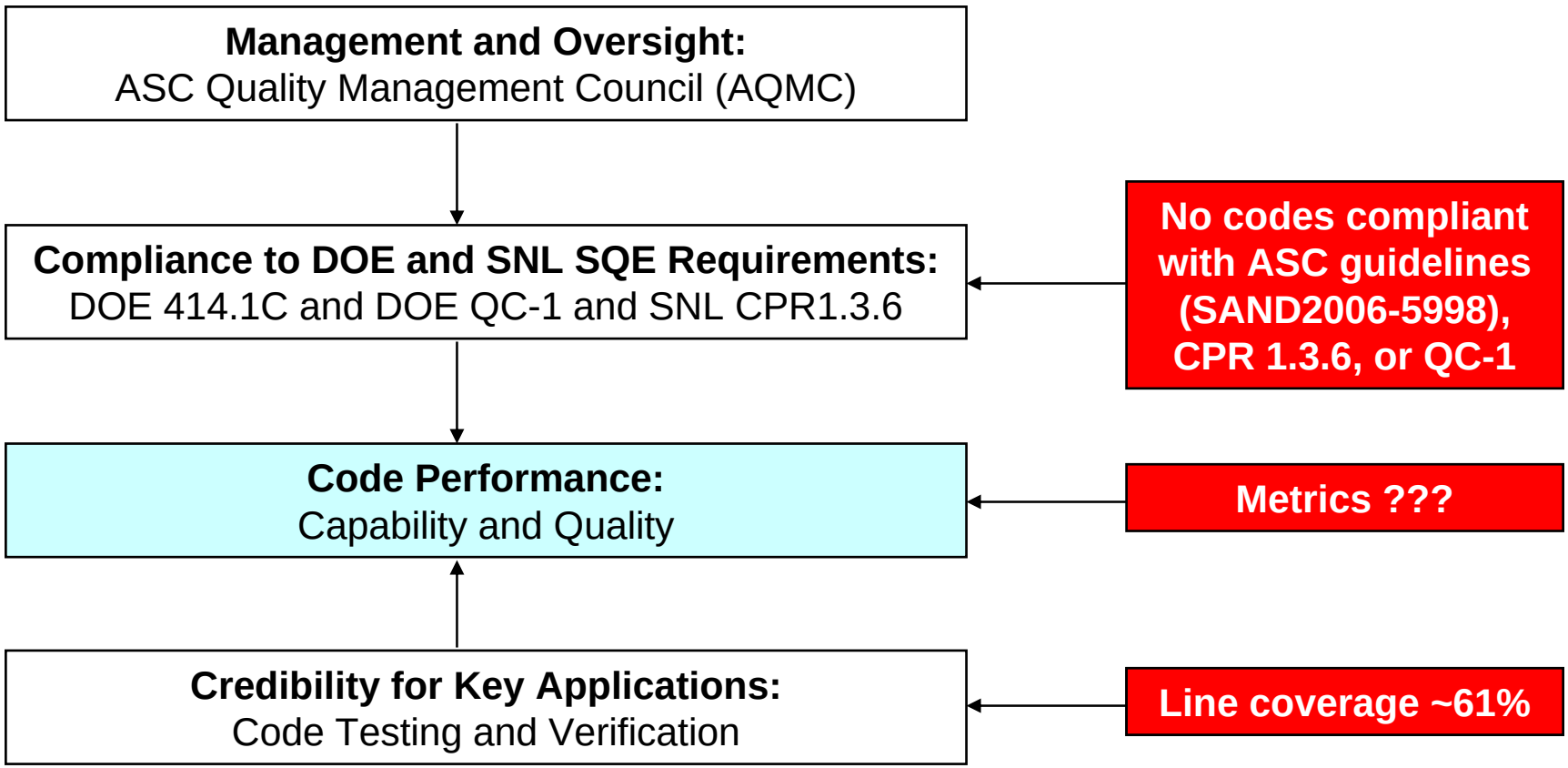
- **Expert opinion counts for *little or nothing* in my thinking about V&V.**
- **So, whatever I mean by V&V intersecting SQE I have evidence somewhere in mind.**
- **Evidence accumulates over time.**

For purposes of this talk, keep in mind that "valid" means "requirements are correct" and "verified" means "requirements correctly implemented."

SQE Intersect V&V:

- ***Statement*: ASC is delivering computational products that will be used to inform and support nuclear weapons stockpile decisions.**
- ***Hypothesis*: Quality of ASC software products is pretty important.**
- ***Question*: How do you define, characterize, and communicate the "Quality" of computational science?**

"Quality" is a many-splendored thing, but let's focus on SQE for the moment. The SNL "story" roughly looks like:



- **The LLNL story, for example, does not look like this: "LLNL tells a good SQE story." (Pilch – but I can so testify.)**
- **Is it important to tell a good SQE story?**

Observations:

- Sandia ASC SQE centers on
  - Graded approach (“risk-based”); and
  - Essentially no requirements (“guidelines”-based).
- ASC program software management is not metrics-based.
  - Metrics are being sought by the ASC program as I speak (led by Pilch).

TOO MANY CODES!

We Have an Obligation to Manage Our Software to Federal and Corporate Requirements

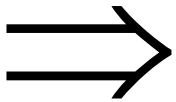
- Even if SNL ASC has no requirements.
- Federal requirements
  - DOE QC-1: http://prp.lanl.gov/documents/qc-1.asp
  - DOE 414.1C: http://prp.lanl.gov/documents/misc.asp
  - Must be prepared to respond to NNSA actions and assessments and DNFSB questions
  - Remember the "Green Document?"
- Sandia requirements
  - Sandia CPR 1.3.6: http://www-irn.sandia.gov/policy/leadership/software_quality.html
- ASC response
  - ASC guidelines: https://wfsprod01.sandia.gov/intradoc-cgi/idc_cgi_isapi.dll?IdcService=GET_SEARCH_RESULTS&QueryText=dDocName=WFS400032
  - ASC mappings: https://wfsprod01.sandia.gov/intradoc-cgi/idc_cgi_isapi.dll?IdcService=GET_SEARCH_RESULTS&QueryText=dDocName=WFS400033

My following comments are organized along the following principle:

Testing

V&V is more testing than anything else.

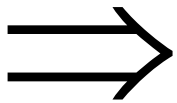
worrying about



Reliability

The accumulation of V&V evidence measures increasing M&S reliability in time.

worrying about



Life Cycle

Accumulation and preservation of V&V evidence is a life cycle issue.

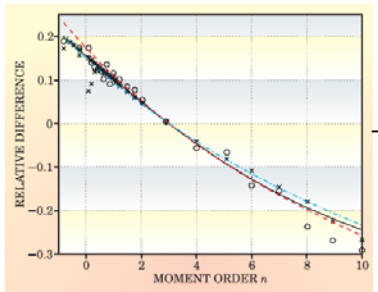
The purpose of Testing is to find defects.

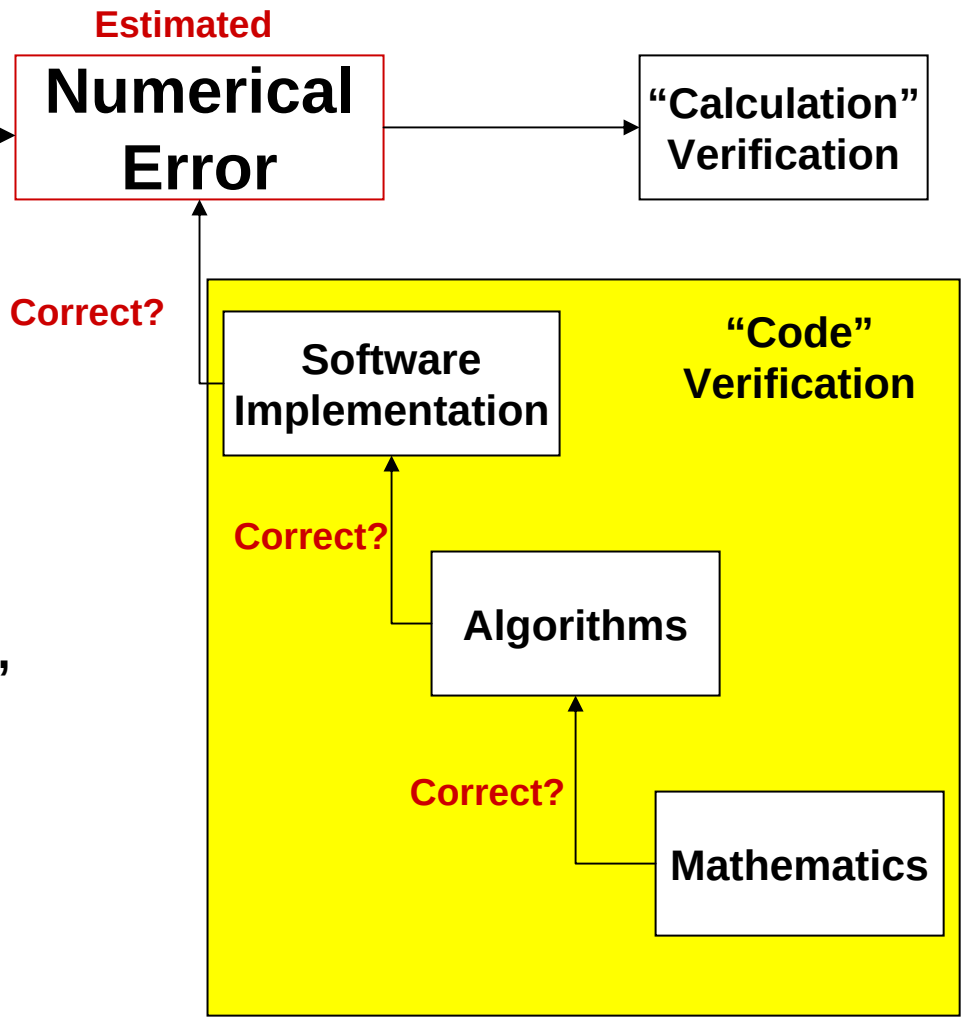
- Does defect detection imply “quality?”
  - Here we are associating “quality” with “reliability.”
  - “Reliability” is a very complex concept for computational science.
- What kind of testing? Why THAT testing?
  - Regression testing? (Line coverage = ~60%. Good? I don’t know. What did we promise?)
  - Verification testing? (Feature coverage = we don’t know. Bad? I’d say so.)

V&V is essentially a lot of sophisticated testing.

"Verification Testing" is a Code Verification Process

Example



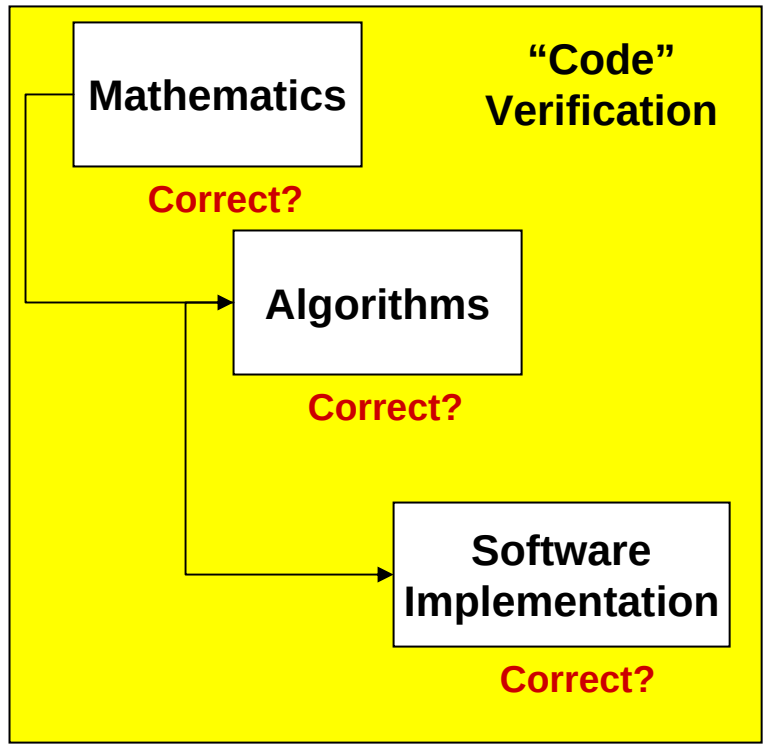


- To believe any numerical error statement requires "code verification."
- Verification testing uses benchmarks that assess the correctness of the mathematics, algorithms, and software implementation.
- See Oberkampf & Trucano (2007), "Verification and Validation Benchmarks," SAND2007-0853.

Sandia National Laboratories

Verification Testing is a necessary component of test engineering.

- Code verification evidence emerges from:
  - Mathematical proof
  - Empirical data
  - SQE processes
  - Verification tests
- (It is an unpleasant complexity that we often don't know whether an algorithm is working until we run the code.)
- It is an unpleasant complexity that verification testing is labor and computer intensive.
- Verification testing is a shared domain between code developers, users, and V&V in an advanced computational science program.



Testing – What should be covered?

Verification Test Problems

Kevin Dowding – CALORE Feature Coverage analysis

Features

PDE terms	wif_application3	wif_application2	wif_application1	rec_2x1_ss_rev1_2d	sph_shell_axi	x21y23z23_6	x21y23z23_5	x21y23z23_3	x21y23z23_1	source_parab_2d	qconst_fi	cyl_shell_2d	qconst_trap	x21y23z23_10	x21y23z23_12	x21y23z23_14	spherical_shell	x21y23z23_9	x11b11_nonlin	aniso_3d	source_parab	rad_gap	x22b10T0_nonlin_trap	1dnonlin_verify1	aniso_2d	rec_2x1_ss_rev1	x22b10T0_nonlin_fi	cyl_shell_3d
Conduction (diffusion term)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Capacitance (transient term)	X	X	X			X	X	X			X		X		X			X					X	X			X	
Src (source term)		X	X	X		X	X	X	X	X				X	X	X		X		X	X				X	X		
EnclRad	X	X	X																			X						
CM (chemistry source term)																												
Conductivity																												
k0 (constant conductivity)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			X	X				X		X
k1 (tabular T-dependant)	X	X	X																X				X				X	
k2 (user subroutine T-dependant, mostly)	X	X	X																	X				X	X			
k3 (defined variable)																												
k4 (anisotropic constant)		X	X																									
k5 (anisotropic tabular T-dependant)	X	X	X																									
Heat capacity																												
Cp0 (constant)	X	X	X			X	X	X			X		X		X			X										
Cp1 (tabular T-dependant)	X	X	X																				X				X	
Cp2 (user subroutine T-dependent)		X																						X				
Cp3 (user variable)																												
Density																												
D0 (constant)	X	X	X			X	X	X			X		X		X			X					X	X			X	
D1 (tabular T-dependent)	X																											
D2 (user subroutine T-dependent)																												
D3 (user variable)																												
D4 (volume dependant)																												
Source terms																												
G0 (constant)		X	X	X		X	X	X	X					X	X	X		X								X		
G1t (tabular, time varing)			X																									
G1T (tabular, temp varing)																												
G2 (user subroutine, time or temp varing)		X								X										X	X				X			
G3 (user variable)																												

63 ID'd features

- Lines?
  - How many lines? Meaningful to who? What do you infer from it?
- Features?
  - Needs to be done to test algorithms anyway
  - Driven by input decks so strong coupling to usage and user buy-in
  - Prioritized by validation plan (please don't ask "What Validation Plan?")
  - Hard to define verification test problems!

Sandia National Laboratories

Hard to define verification test problems …

- Defining verification test problems is a major effort (or we wouldn't even be having this discussion!).
- It's not just finding test problems – they must be relevant (or nobody cares) and they must be assessable (and assessed).
- (Aiming for a standard is also part of this particular Tri-Lab effort.)

DRAFT 11/25/07

Enhanced Verification Test Suite for Physics Simulation Codes

James R. Kamm, Jerry S. Brock, Francis X. Timmes
Los Alamos National Laboratory

Barna L. Bihari, John H. Bolstad, Scott T. Brandon, David S. Miller, Carol S. Woodward
Lawrence Livermore National Laboratory

Patrick Knupp, William J. Rider, Timothy G. Trucano, Gregory V. Weirs
Sandia National Laboratories

26 September 2007

LA-UR-07-XXXX ADC: RJB

Sandia National Laboratories

Testing for feature coverage continued…

- There are gaps in the feature coverage.
- An important question is testing feature interactions. These are complex because:
  - Difficulty in devising test problems for controlled interaction of features.
  - Inferring the nature of feature test interactions from more integral test problems.
  - Test interactions that replicate user profiles.
- Remaining questions:
  - Systematic grinds of such test suites (that is, making this type of testing part of the development process).
  - Pass/Fail assessments (human intensive and not like regression testing).
- Current ASC test engineering can't absorb this kind of testing.
  - (Increasing the level of incompleteness of verification, by the way!)

Do this for ALL THOSE CODES!?

Assuming the test engineering can be handled – What does testing do for us?

- "Detecting and removing defects doesn't mean the software has higher reliability. It means only that you have removed defects…"
  - "The purpose of defect detection is to remove defects."
- I have described (verification) testing, but I have not defined "defect." – "There is no agreement on what the word defect means."
  - The concept of defect becomes broader and more diffuse in computational science (e.g. perfect software producing wrong solutions).

The purpose of verification testing is to detect, then remove, math, algorithm, and software defects.

Whatever "defect" means, do we increase "reliability" from detection and removal?

- This is a BIG PROBLEM.
- An example thought process is to use a claimed empirical correlation (Capers Jones and others) that estimated "software defect density" reflects "software quality."
  - One then estimates defect density, e.g.

$$\text{Software Defect Density} = k \times \frac{\text{Defect Discovery Rate}}{\text{Usage Measure}}$$

- Unclear what "quality" means in this correlation.
- Unclear that it has much to do with solving PDEs accurately.
- User satisfaction is clearly part of the "quality" challenge.

And reliability means ???

- There is an insidious undercurrent that software “quality” is somehow the same thing as M&S “reliability,” which are both somehow the same thing as “predictive computational science.”
  - Is this really true?
  - We need to understand the details, especially the relationship to “defect” detection and removal.

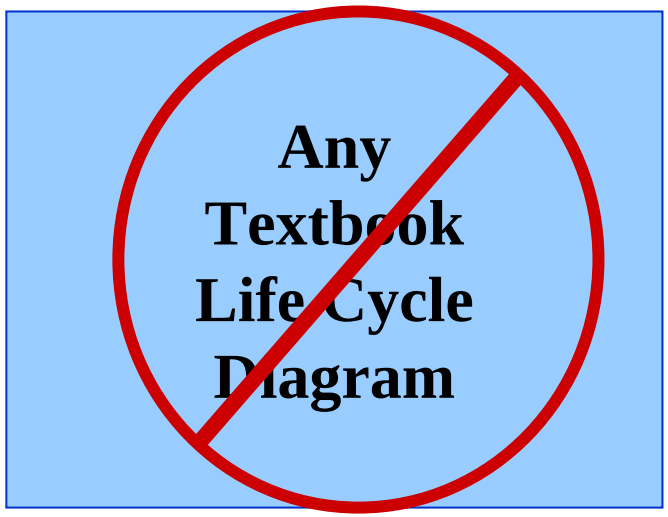
Why is this so important? Because it intersects life cycle management.

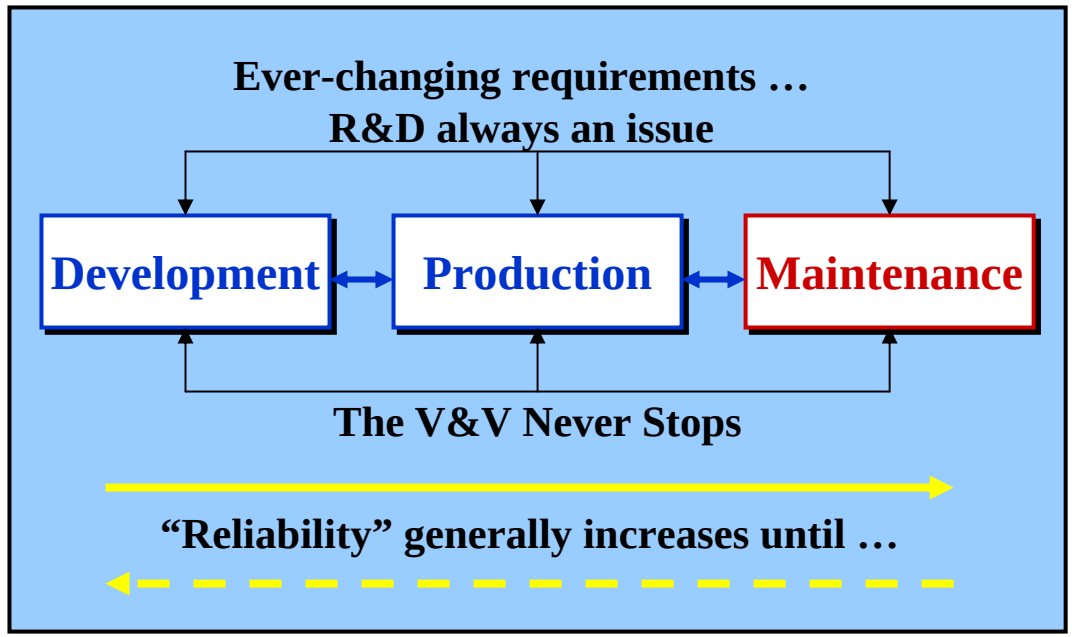
- To perform cost/benefit analyses, and properly manage ASC code projects, you need to understand what "defect" means, what "reliability" means, the link of reliability to defect detection and removal, and how to manage this process over the software life cycle.
  - Example: Is V&V too expensive? Based on what?
- Given the above statements, defect detection and removal never stops, so reliability is a fundamental life cycle issue.
  - Example: How much money should be spent on reliability? How does the $ flux depend on the life cycle (please don't ask "What life cycle?")?

The absence of a defined life cycle, and defect/reliability concepts appropriate for rigorous cost/benefit analyses, is of concern.

Life Cycle MANAGEMENT

- **The purpose of a life cycle from the perspective of this talk is to understand how to MANAGE PRODUCTION SOFTWARE PRODUCT.**





Where is the money coming from?

- **"Whatever program replaces ASC…"**

Sandia National Laboratories

A Modest Proposal:

- Link SQE in the ASC COMPUTATIONAL SCIENCE program with an understanding of what "quality" means: SQE should contribute to Predictive Computational Science in ASC.
- This understanding should be metrics-based.
- "Production" should be that software state in which "reliability" has increased – learn how to expect it and recognize it.
- Develop an institutional perspective on what "supporting production quality software" means, and what that requires.
- AND YOU HAVE TO DO THIS FOR ALL THOSE CODES!
- Does "engineering" properly describe what we are doing in ASC? Or what we SHOULD be doing?

It is hard to understand how V&V does not strongly intersect these issues.

- **V&V is part of the solution, not part of the problem.**