

# Uncertainty in Verification and Validation: Recent Perspective

**Tim Trucano**

**Optimization and Uncertainty Estimation, 09211**

**PO Box 5800, MS 0370**

**Sandia National Laboratories**

**Albuquerque, NM 87185-0370**

**tgtruca@sandia.gov**

**(505)844-8812**

**2005 SIAM Conference on Computational Science and  
Engineering, February 12-15, 2005, Orlando, Florida**



# **“Is Your Calculation Probably Accurate?”**

SAND2005-0945C

Timothy Trucano, William Oberkampf, Martin Pilch, and Laura Swiler

---

**The premise of this talk is that it is important to view accuracy in complex computational science as uncertain. We will explain this claim and present some ideas that support it. These ideas include (1) the challenges of experimental validation; (2) reliability of computational science software; and (3) the use of under resolved computation. We conclude by emphasizing the impact that uncertainty in computational error has on predictability and application of computational science.**

**The purpose of computing is not insight. 😊**

---

## **ASCI –**

- The purpose of computing is to provide “high-performance, full-system, high-fidelity-physics predictive codes to support weapon assessments, renewal process analyses, accident analyses, and certification.” (DOE/DP-99-000010592)**

**Verification** – Are the equations solved correctly? (Math)

**Validation** – Are the equations correct? (Physics)

## ASC:

- **Verification:** The process of confirming that a computer code correctly implements the algorithms that were intended.
- **Validation:** The process of confirming that the predictions of a code adequately represent measured physical phenomena.

## AIAA:

- **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.

## IEEE:

- **Verification:** (1) The process of evaluating a [software] system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (2) Formal proof of program correctness.
- **Validation:** The process of evaluating a [software] system or component during or at the end of the development process to determine whether it satisfies specified requirements.

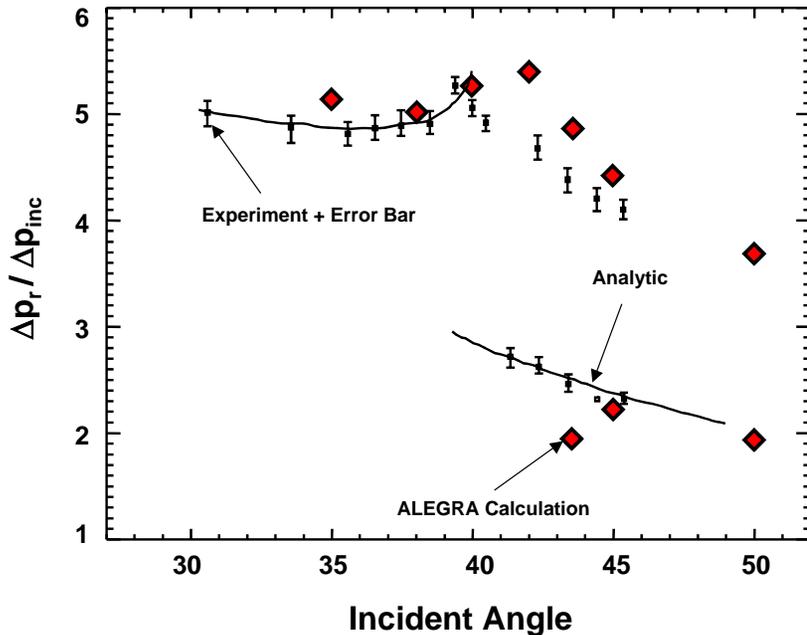
# This talk is about verification

---

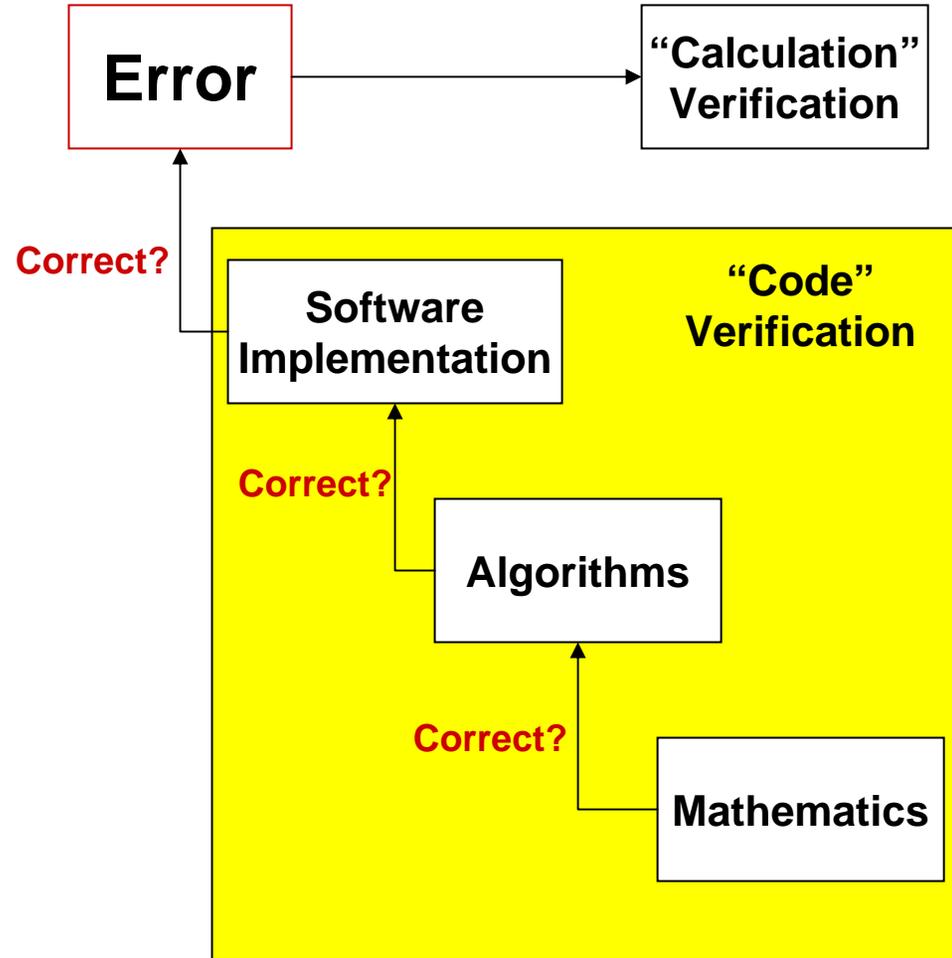
- **Verification centers on mathematical and computational accuracy.**
- **Validation, applications and decisions depend on verification.**
- **Are there reasons to believe that the accuracy of given calculations may have to be “understood” probabilistically?**
- **Are there ideas that help?**

# Consider the following “validation” exercise. SAND2005-0945C

This is math as well as physics.



- The calculations (diamonds) are not converged and I don't know what the *computational error* is.
- What does the comparison mean?





SAND2005-0945C

# Uncertainty In verification arises from:

---

## Software implementation errors – BUGS

- Code crashes are the least of our problems.
- Mutually reinforcing errors are also “easily” detectable.
- Mutually canceling errors are of greater concern.

## Inadequate algorithms and incorrect mathematics

- No amount of resolution will solve the problem.

## Inadequate resolution

- Resolution “solves” the problem but is probably unavailable for the hardest applications.

# Passage from determinism to uncertainty.

SAND2005-0945C

**Deterministic: The error  $\leq E$**

**Qualitative UQ: “I’m uncertain what the accuracy of this calculation is.”**

- Formal methods  (will they work for CS&E?)
- Software engineering  (“we don’t like to do this”)
- “Non-SQE” testing (classical verification)
- Deterministic error models (i.e. a posteriori error estimation)
- Probabilistic error models

**Quantitative UQ leap: “I need to apply probabilistic language to describe my understanding of the accuracy of this calculation”**



# Is Probabilistic Software Reliability (PSR) SAND2005-0945C useful for computational science software?

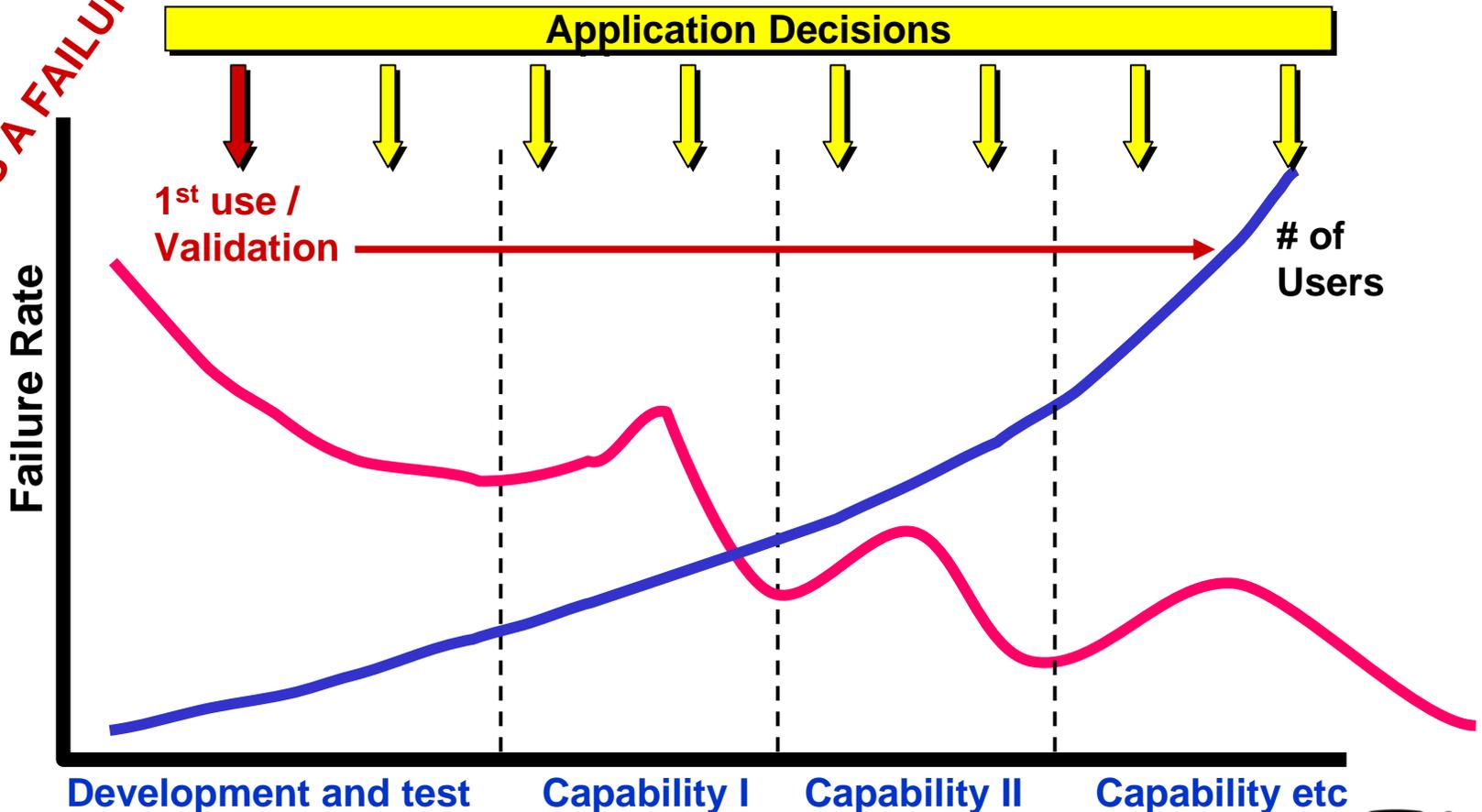
---

- We are test fixated in building scientific software, properly so:  
“Based on the software developer and user surveys, the national annual costs of an inadequate infrastructure for software testing is estimated to range from \$22.2 to \$59.5 billion.” (“The Economic Impacts of Inadequate Infrastructure for Software Testing,” NIST report, 2002.)
- Can we test software perfectly or can’t we?
- If we can’t test software perfectly testing alone does not solve the verification problem.
- Users are really important, but...
- Dumping lousy software on users and expecting them to find bugs and be happy about it because its “computational science” is not smart.

# My view of software "reliability" is correlation of decreasing # of "failures" and increasing # of "users".

A notional view of "reliability" for a general-purpose PDE code (say hydrocode):

WHAT IS A FAILURE?!





# What is probabilistic in software reliability and what isn't?

---

**Software reliability** is “the probability of failure-free operation of a computer program in a specified environment for a specified period of time.”

- Probability here is interpreted as the probability of detection, not probability of existence.
- Major complexity arises from accounting for software modification.
  - One assumption made is that new bugs are not introduced repairing old ones.
  - One reason why regression testing is performed.

**WHAT IS A FAILURE?!**

# Is probability here a frequency?

One interpretation of software reliability might look like the following:

**Software  
Module M**

“Equivalent Modules”

- “Same” language
- “Same” length
- “Same” complexity
- “Same” usage

“Failure” in time T?

Yes = 1

No = 0

$M_1$  Fail?

**Software  
Module  $M_1$**

$M_N$  Fail?

**WHAT IS A FAILURE?!**

$$P(M_{N+1} \text{ has failure}) \sim \frac{\# \text{ of failures}}{\# \text{ of modules}}$$

if N is large enough

**This is highly unlikely to be a useful model?**

## Quantities of interest:

$N(t)$  = # of failures in  $[0, t]$

$X_i$  = time between failures  $i-1$  and  $i$

$T_i$  = time to failure  $i$ ;  $T_i = \sum_{j=1}^i X_j$

$N(t)$  is treated as a stochastic process.

Canonical questions include:

- Model the number of failures up to a given time.
  - Poisson processes
- Model the times between successive failures.
  - Auto-regressive processes
  - Failure rate is controversial (read the references)

WHAT IS A FAILURE?!

[N. D. Singpurwalla and S. P. Wilson \(1999\), Statistical Methods in Software Engineering, Springer-Verlag \(Bayesian approach\).](#)

## Example: (Musa and Okumoto)

Note that  $\Lambda^*(t)$  ("mean value function) is the expected value of  $N(t)$ .  
The derivative of  $\Lambda^*(t)$ ,  $\lambda^*(t)$ , if it exists is failure rate at time  $t$ . Assume

$$\lambda^*(t) = \lambda_0 e^{-\theta \Lambda^*(t)}$$

Then,

$$\lambda^*(t) = \frac{\lambda_0}{\lambda_0 \theta t + 1}; \Lambda^*(t) = \frac{1}{\theta} \ln(\lambda_0 \theta t + 1)$$

This leads to

$$P(N(t) = k) = \frac{(\ln(\lambda_0 \theta t + 1))^k}{k! \theta^k (\lambda_0 \theta t + 1)^{1/\theta}}$$

**WHAT IS A FAILURE?!**

- Estimating parameters in this model is subtle. Standard maximum likelihood estimation is not recommended.
- See Singpurwalla and Wilson for tons more examples.

# Statistical testing is of great interest

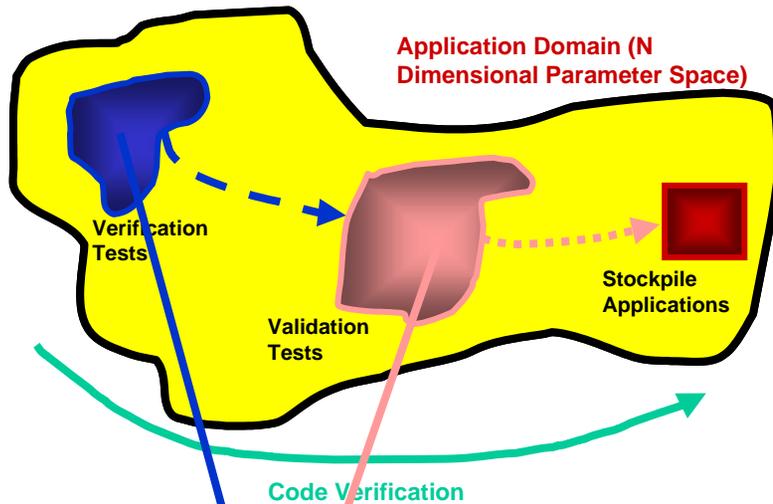
SAND2005-0945C

- G. M. Kaufman (1996), "Successive Sampling and Software Reliability," J. Statistical Planning and Inference, Vol. 49, 343-369.
  - How many faults remain in a software system at a given time with a given history of fault detection?
  - What is the waiting time to the next failure?
- T. L. Graves, et al. (2001), "An Empirical Study of Regression Test Selection Techniques," ACM Trans. Software Engineering and Methodology, Vol. 10, No. 2, 184-208.
  - Contrasts the effectiveness of random test designs for regression testing (random walks on a graph) with other methods.
  - Note that this involves how to correlate this test design (called **synthetic testing**) with the estimated operational profile of the software (**operational testing**); this should involve careful characterization of "users".
- B. Littlewood and D. Wright (1997), "Some Conservative Stopping Rules for the Operational Testing of Safety-Critical Software," IEEE Trans. Software Engineering, Vol. 23, No. 11, 673-683.
  - How to modify testing after N tests if M faults have been detected (related to weighted sampling without substitution).
  - How to stop testing after P testing cycles (requires a probabilistic reliability statement that is passes acceptance criteria).

WHAT IS A FAILURE?

# Using a reliability approach for verification to make application decisions – example:

SAND2005-0945C



Define Tests 1, ..., N;  
Define “**FAILURE**”;  
Stopping criteria as in Littlewood and Wright  
**Pass? → Perform application**  
**Fail? → Continue testing**

- Developing *stopping rules* is one of the areas of interest:  
**HOW MUCH VERIFICATION IS ENOUGH?**
- We are interested in detecting “FAILURES” over a high-dimensional space (e.g. space of all input parameters for the code).
- Can the stochastic reliability framework be extended to spatial processes?



# Are Probabilistic Error Models (PEM) useful for computational science software?

SAND2005-0945C

---

- Suppose we face up to uncertainty in the process and results of verification.
  - “When quantifying uncertainty, one cannot make errors small and then neglect them, as is the goal of classical numerical analysis; rather we must of necessity study and model these errors.”
  - “...most simulations of key problems will continue to be under resolved, and consequently useful models of solution errors must be applicable in such circumstances.”
  - “...an uncertain input parameter will lead not only to an uncertain solution but to an uncertain solution error as well.”
- These quotes reflect a new view of “numerical error” expressed in B. DeVolder, J. Glimm, et al. (2001), “Uncertainty Quantification for Multiscale Simulations,” Los Alamos National Laboratory, LAUR-01-4022.



# Empirical, sampling-based investigation of numerical error is studied.

SAND2005-0945C

---

- The goal is to understand structural features of probabilistic error models in the particular case of conservation law solution.
- This requires a fiducial.
  - In the case of verification test problems, this fiducial is exactly known for analytic tests.
  - In the absence of analytic solutions, fine grid solutions serve as the fiducial instead.
- A *verification metric* is required to define error as the difference between a calculation and the fiducial.
- Sources of uncertainty in the numerical simulation are given pdf's and sampled, creating an ensemble from which an (empirical) distribution for the error is created (a empirical random field).

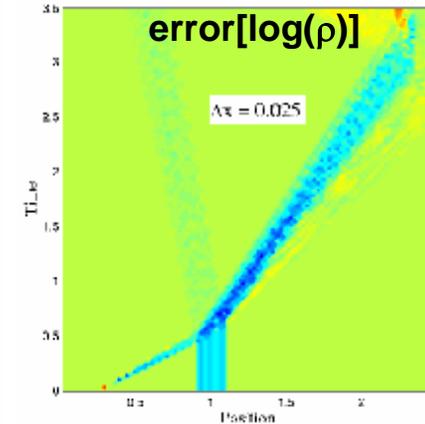
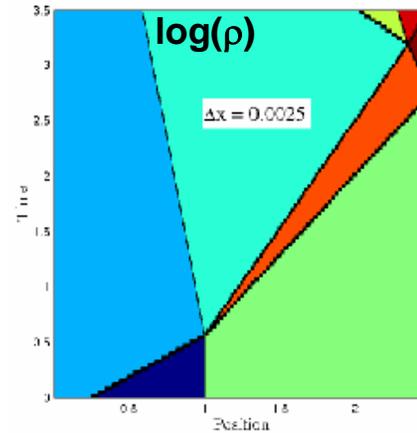
**CAN THE RESULTING ERROR CHARACTERIZATION BE USED TO PREDICT?**

# Error Uncertainty: Shock Tube Verification Problem

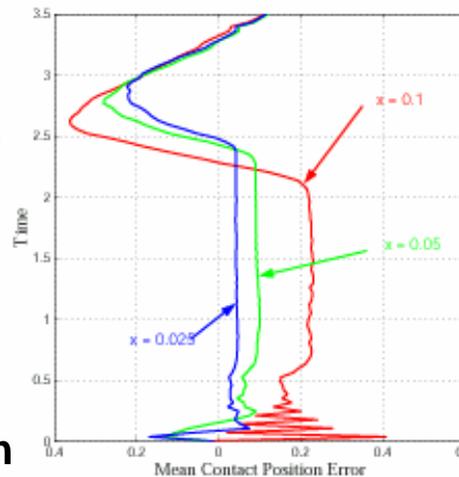
SAND2005-0945C



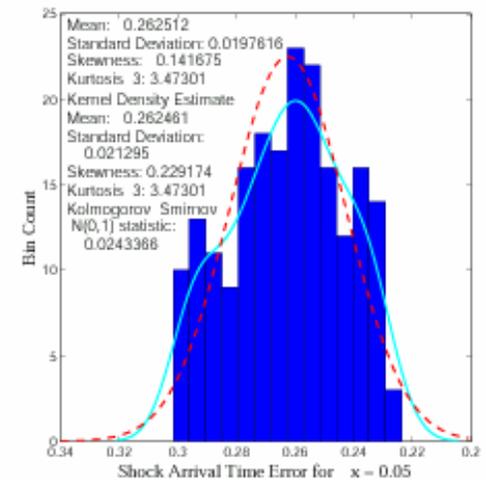
- The problem is a simple shock problem involving shock transmission and reflection from a contact discontinuity, with analytic solution.
- Key features are various error space-time trajectories and dependency on resolution.
- This is pretty interesting and pretty unusual.
  - Generalize to entire suite of test problems? Means what?
  - Way of looking at errors in reduced-order models?
  - Etc



mean error in contact x



empirical histogram of error in shock arrival time at wall



I'll revisit from validation perspective.



# Hunch: Multi-scale models are likely candidates for PEM.

SAND2005-0945C

---

- **Multi-scale models are by definition under-resolved.**
  - **Multi-scale modeling implies subgrid information which is often canonically treated by probabilistic techniques, so the resulting error already has an explicit probabilistic component.**
- **What is the effect of upscaling and downscaling requirements in multi-scale?**
- **Should we solve stochastic differential equation approximations to model multi-scale PEM?**
- **Current practice is to hope the subgrid “error” and micro-scale/macro-scale inconsistencies don’t blow up in our face.**
- **I really don’t have a clue how to “verify” such calculations with classical concepts.**

## CONCLUSIONS:

- We have uncertainty in the numerical accuracy of our most important calculations (verification) as well as the physical accuracy (validation).
- Verification uncertainty is a proper component of uncertainty quantification in computational science.

## QUESTIONS:

- Can probabilistic software reliability be extended to include “failures” typical of CSE (algorithm performance; resolution)
- Can CSE acceptance criteria be devised based on statistical software reliability ideas?
- How can we most usefully extend the concept of PEM (verification test suites; validation test suites; error prediction; how to calculate)?
- How does a probabilistic interpretation of our understanding of *numerical errors* influence decisions that rely upon numerical models?



# **“An Uncertain Foundation for Validation”**

SAND2005-0945C

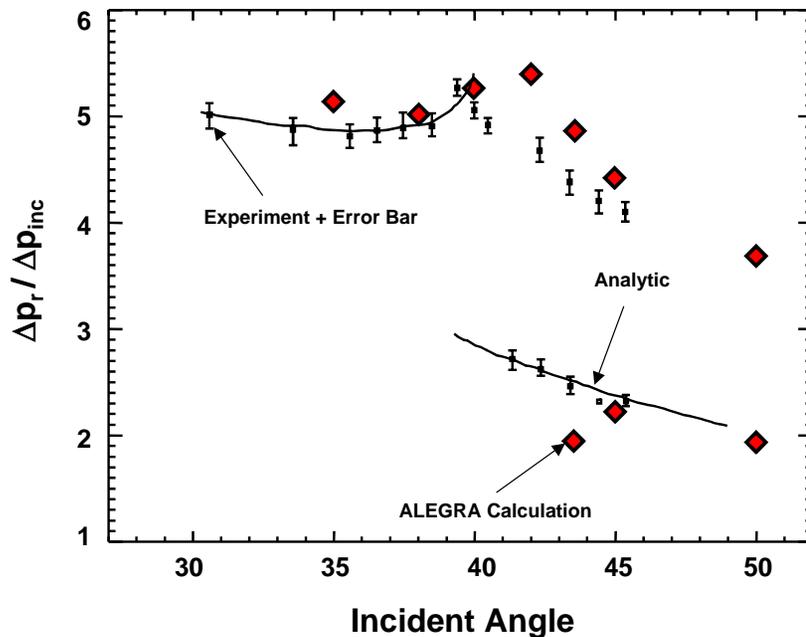
**Timothy Trucano, William Oberkampf, Martin Pilch and Laura Swiler**

---

**Validation targets the external logic of computational modeling, that is, quantification of the degree that modeling is in agreement with the physical world. (Verification targets internal logic, the foundation and measurement of accuracy of computational solutions.) Validation depends upon comparison of computations with physical observations. This comparison process fundamentally introduces uncertainty into the foundations of validation, which guides the methodology and constrains the conclusions. This talk briefly discusses some important elements of this problem.**

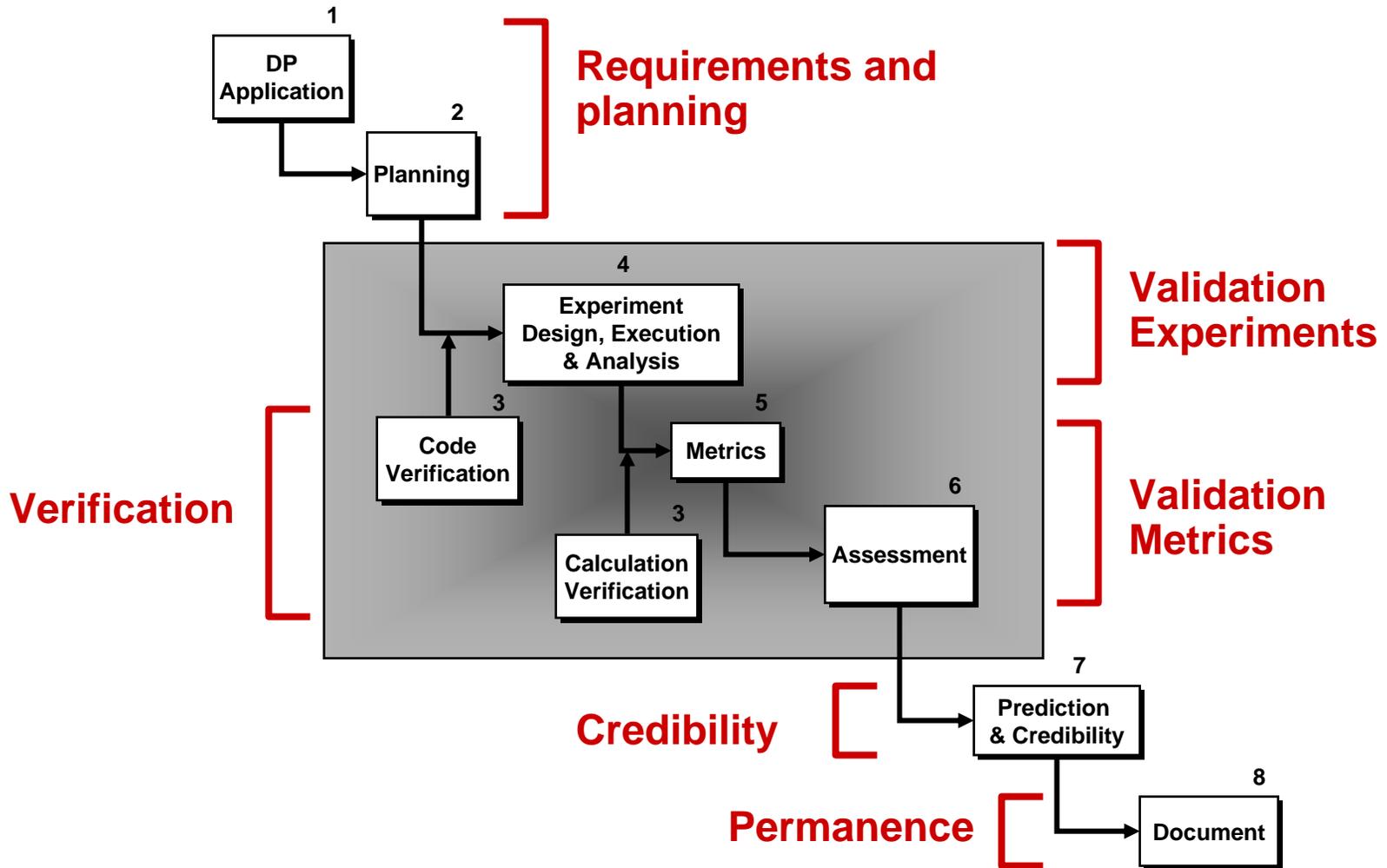
- Validation centers on physical accuracy, and is of special concern in predictive, high-consequence application domains.
- Uncertainty is prominent in validation.
- Are there ideas that help?

This is physics as well as math.

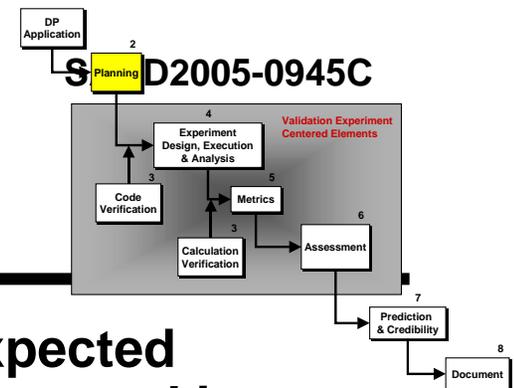


- Assume calculations (diamonds) are converged (sufficiently accurate); say the error bar is the size of the symbol.
- What does the comparison mean?

# Summary of the Sandia program methodology for experimental validation:

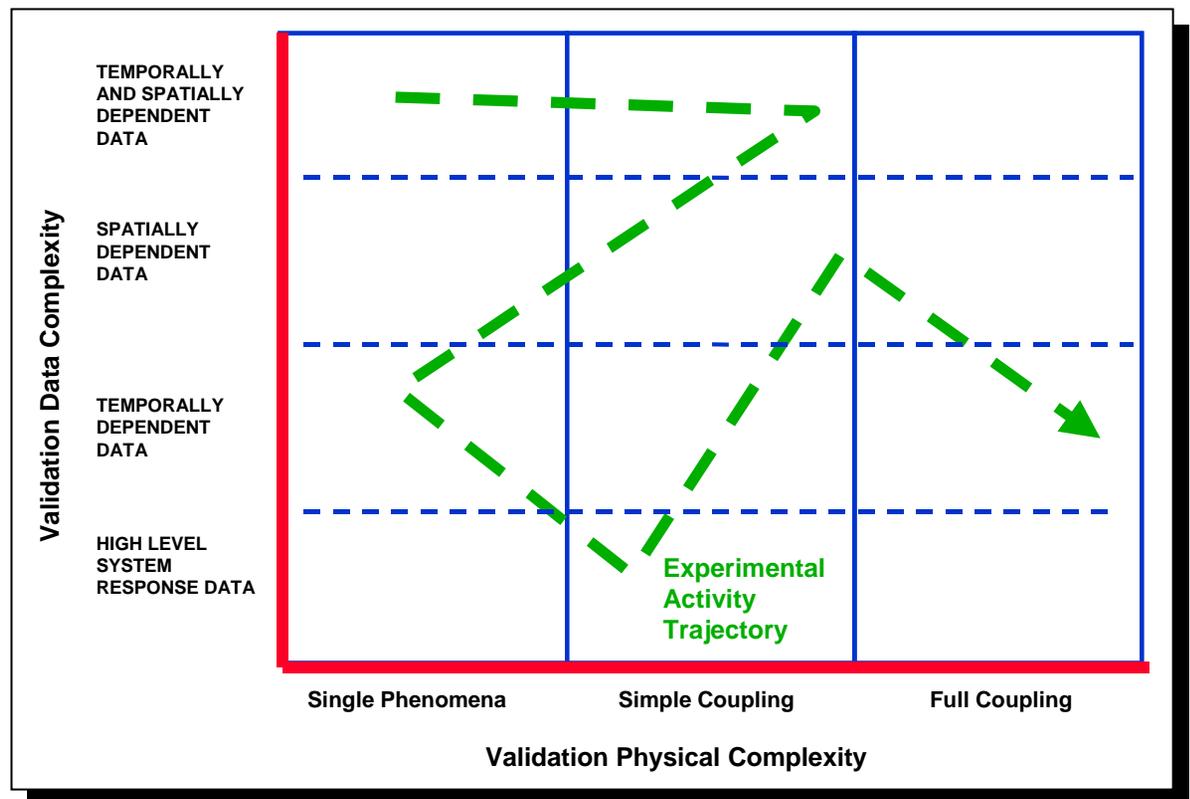


## Element 2: Planning is an essential component of experimental validation.



The PIRT directly defines the progression and expected connection of validation activities and reflects expected impact.

- Reflects the complexity of the defining application (and the application needs to be the “right size”)
- Avoids ad hoc experimental activities
- Maximizes impact of experiments on validation
- Defining application can be non-trivial (i.e. social models)



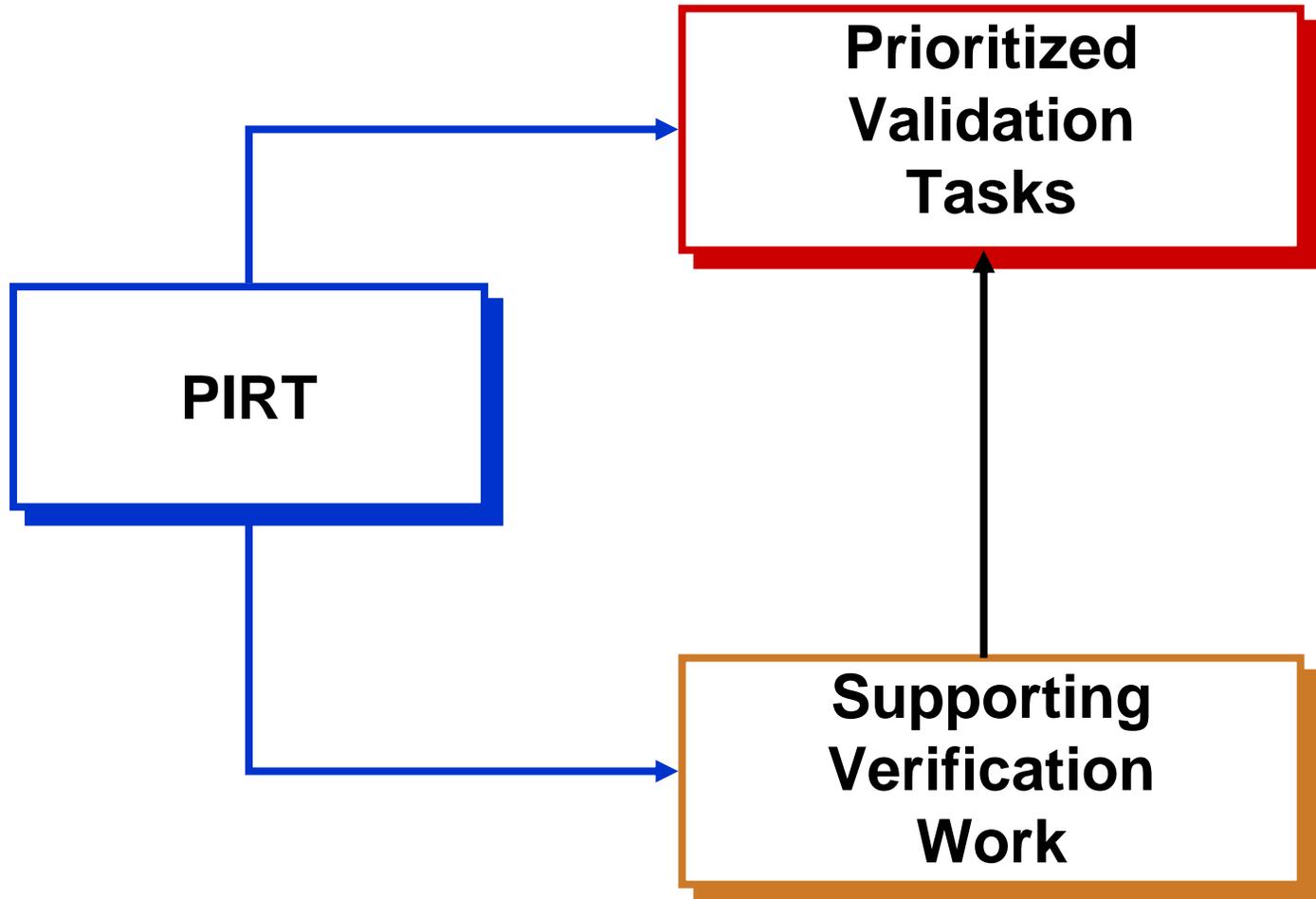
# PIRT – Phenomenology Identification and Ranking Table – Prioritizes the work.

	Experiment Analysis	Experiment Design
PIRT Element	Importance Adequacy	Importance Adequacy

- PIRTs are related to quality function deployment.
- PIRTs decompose and map the application to the M&S structure.
- PIRTs are built on ranking scales (typically 1-4); ranking depends on current knowledge.
- Gaps prioritize V&V work.
- PIRTs are most effective by achieving a balance between narrow and broad.
- PIRTs change based on new knowledge.
- **It's a grocery list! Not a maximization of expected collective utility.**

# Rough Information Flow:

---

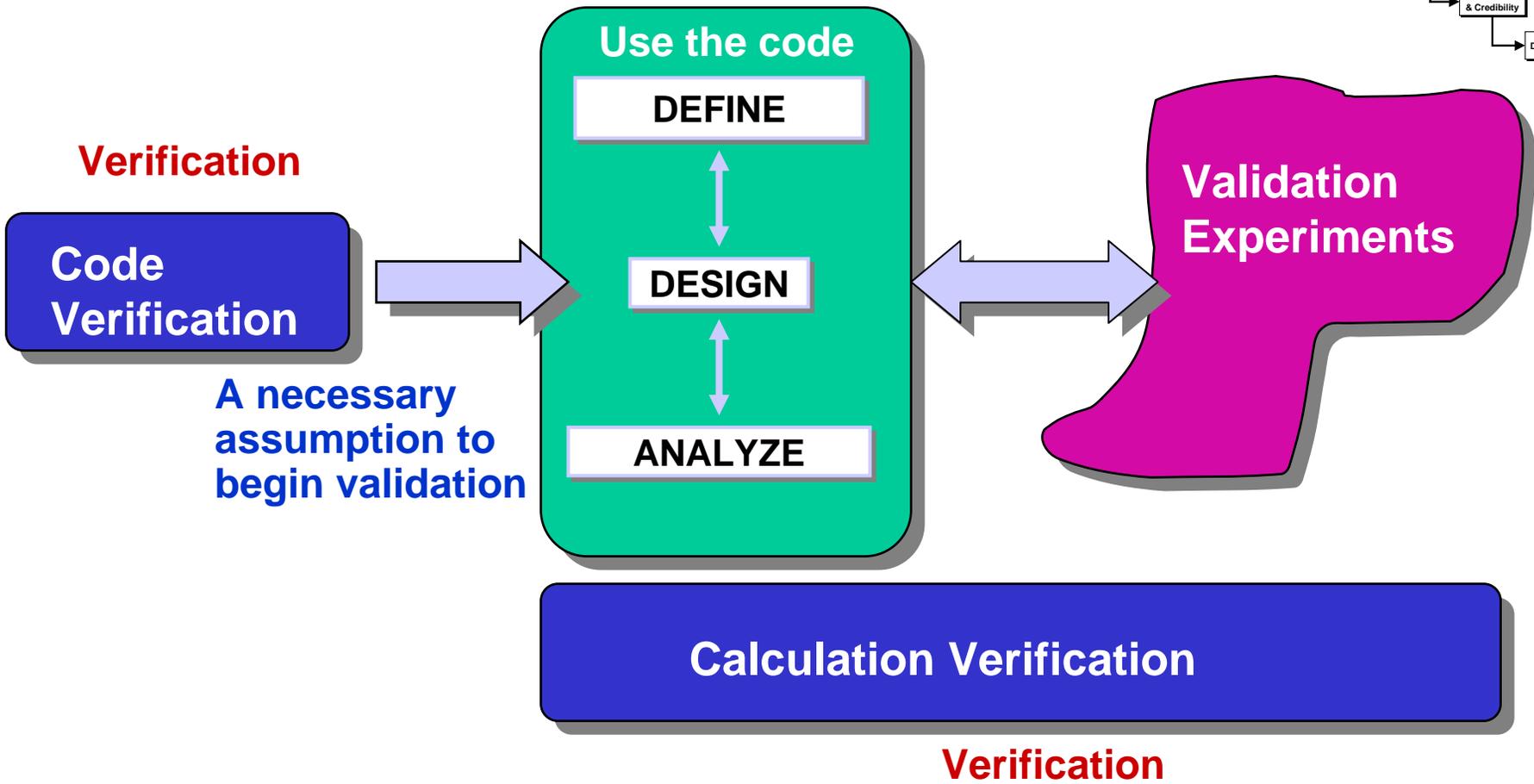
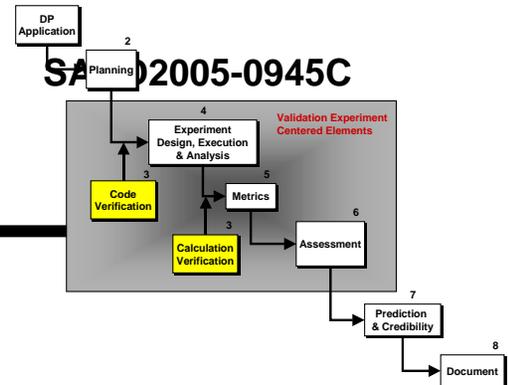


# Example of a PIRT.

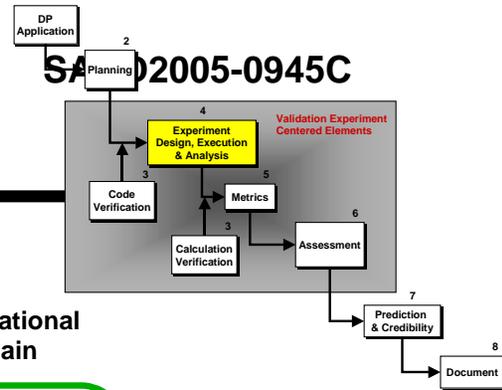
Phenomena	Import	Adequacy			
	Phen	Mod	Code	Val	Mats
<b>Radiation Processes</b>					
Radiation heat transfer	H	H	H	L	
Emission: mesoscale (~1cm) turbulent mixing (affecting flame area)	H	M	L	L	
Emissive flux: combustion chemistry (affects soot temperature)	H	M	L	L	
Emissive flux: fine scale turbulent strain (~1mm) (affects soot temperature)	U	L	L	L	
Emissive flux: soot diffusive transport (affects soot mass fraction and soot temperature)	H	L	L	L	
Emissive flux: soot formation chemistry (affects soot mass fraction)	H	M	L	L	
Emissive flux: soot oxidation chemistry (affects soot mass fraction)	H	M	L	L	
Emissive flux: gas emission	L	M	L	L	
Emission: Spectral dependence	L	M	L	L	
Absorption: mesoscale source term	M	H	L	L	
Scattering: mesoscale source term	L	H	L	L	
<b>Convective Processes</b>					
Convective heat transfer to the	M	M	M	L	
<b>Transport Processes</b>					
Large scale turbulent mixing affecting flame geometry and radiation view factors	H	M	M	L	
Fuel vaporization from surface of spilled fuel	M	M	L	L	
<b>Couplings</b>					
(CALORE to FUEGO)	M	M	L	L	
<b>Material Properties</b>					
Soot optical properties (extinction coefficient)	H				M

Gap analysis represented by the color coding.

# Element 3: Verification and validation are coupled.



# Dedicated validation experiments.

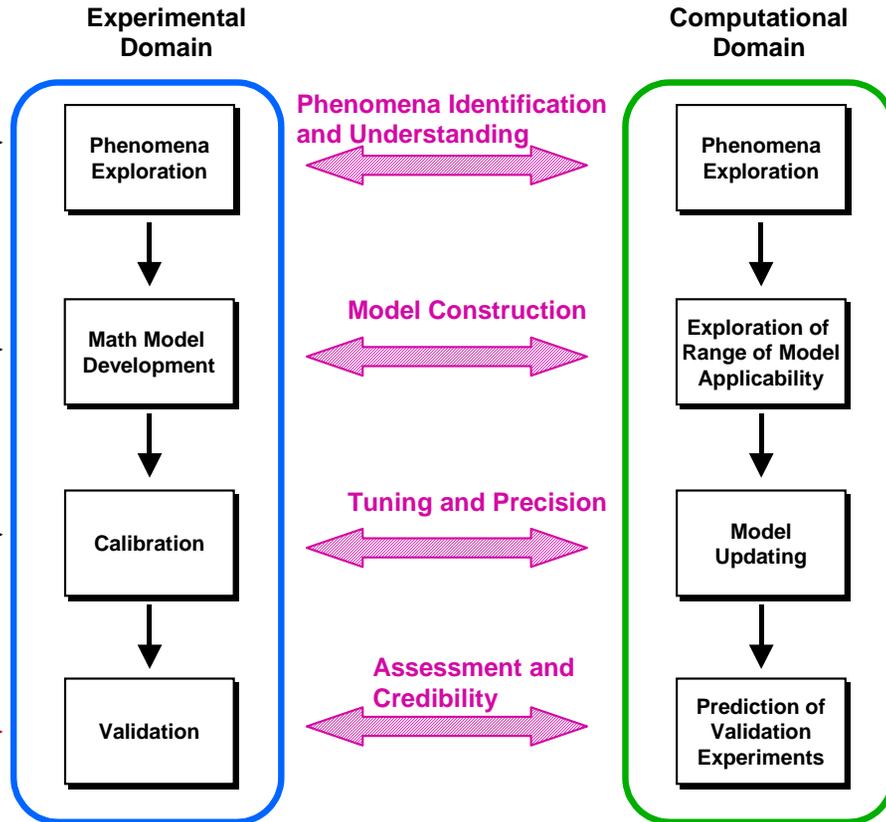


Phenomena Exploration

Model Development Experiments

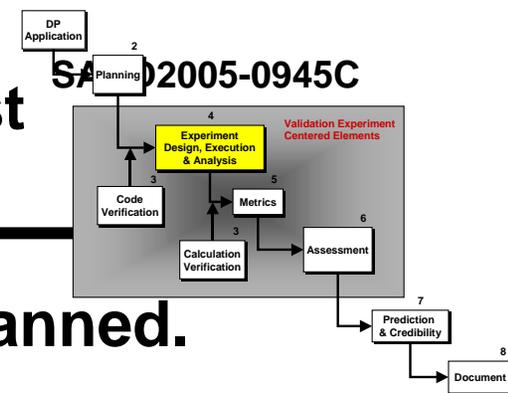
Calibration Experiments

Dedicated Validation Experiments



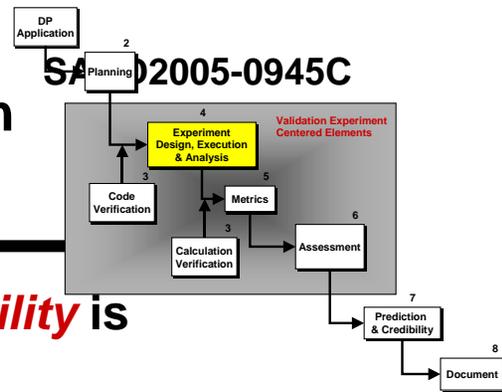
Foundation of Credibility: Getting The Right Answer for the Right Reasons

## Element 4: Dedicated validation experiments must be designed rationally.

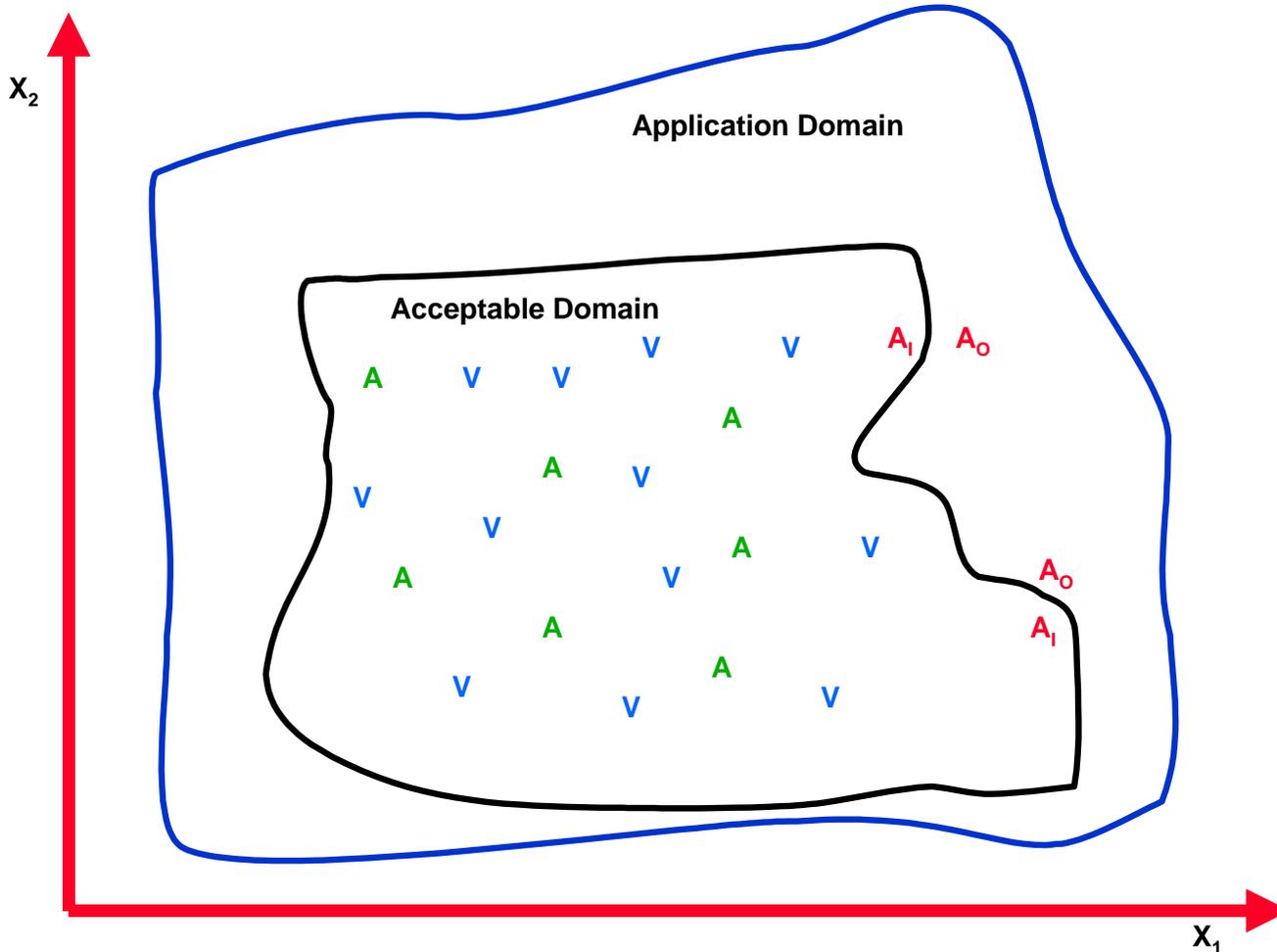


- Validation experiments should be planned.
- The anticipated use of their outcomes should be planned.
- The subject code should be engaged in the definition and design of the experiments, not only in the analysis of their outcomes.
- And yes, we did suggest the possibility of using statistical design of experiments.
- The rational design of **validation experiments** emphasizes the multidisciplinary team that should be engaged on **experimental validation**.

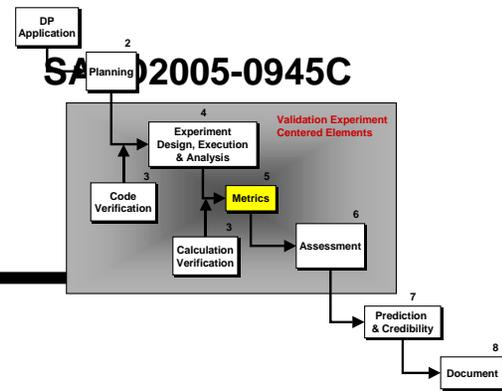
One reason for rational experimental design is to determine boundaries with some accuracy.



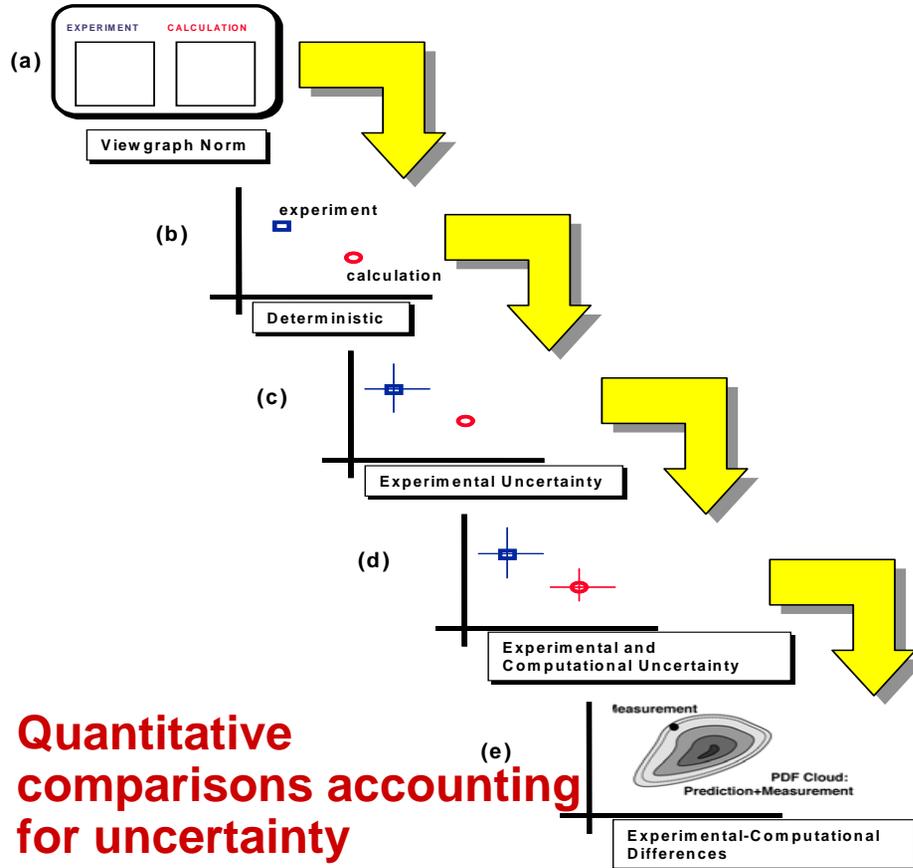
Some understanding of the *boundary of applicability* is needed for decisively predictive calculations.



# Element 5: Metrics define how we compare calculations with experimental data.



## Viewgraph Norm



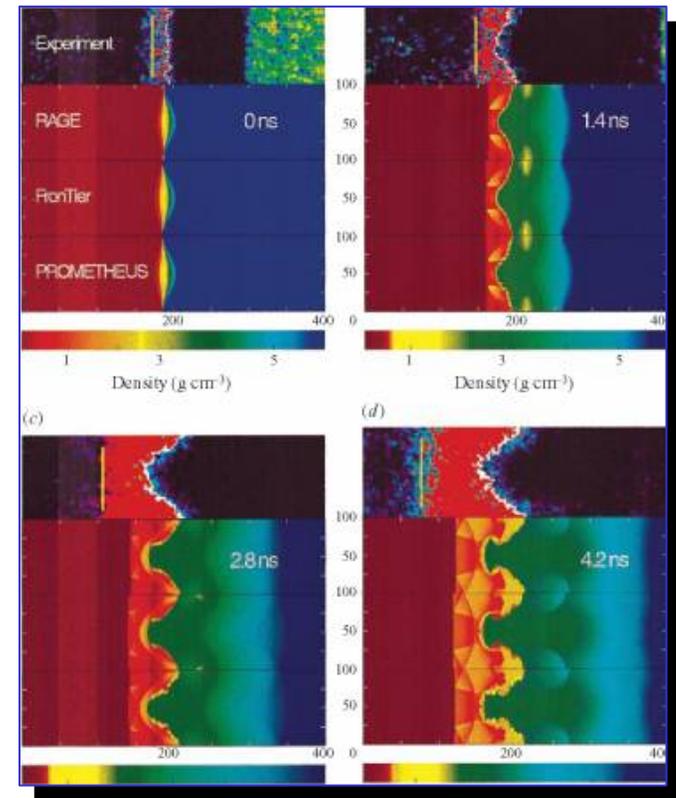
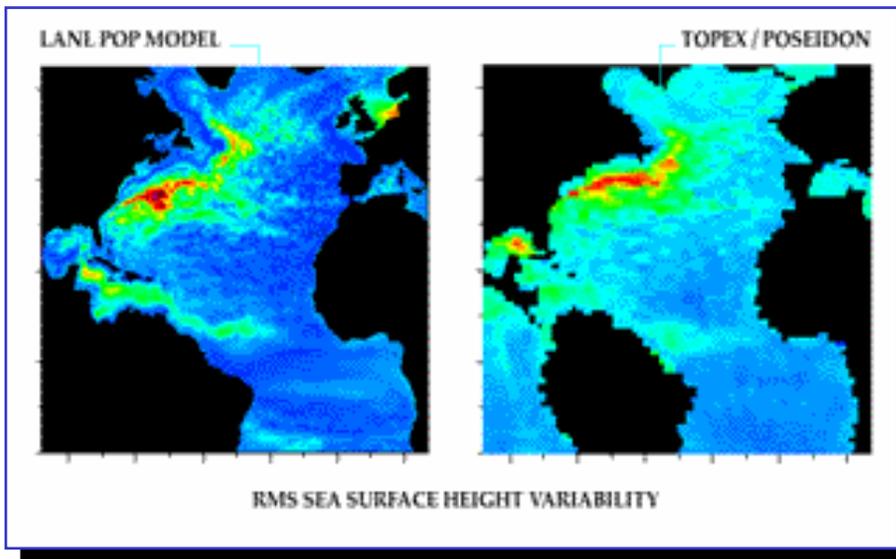
Quantitative comparisons accounting for uncertainty

- Quantitative
- Focus on differences that include both experimental and computational uncertainty.
- Understanding (not confusion) in the use of the code for the defined application should either increase or decrease as a result of these metrics.
- In an ideal world, **success/failure criteria** associated with these metrics will be defined prior to application of the metrics. The world is not ideal.
- See Trucano, et al (2001), "Description of the Sandia Validation Metrics Project," SAND2001-1339.

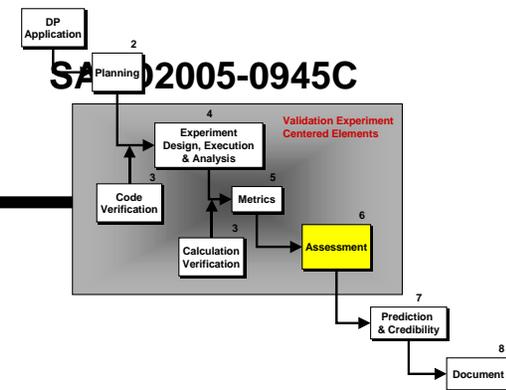
# Viewgraph norms are not validation metrics

SAND2005-0945C

- Possibly useful for science, these are useless, if not dangerous, for validation.
- Uncertainty must be quantified in validation.



## Element 6: Assessment



Assessment could be coupled to the definition of validation metrics – isolating it as a separate element emphasizes it is hard.

- **Quantification enables assessment!**
- Was the comparison **GOOD?** (“Success”)  
**BAD?** (“Failure”)
- Why? And how will this information be used for future improvement of experiments and calculations?

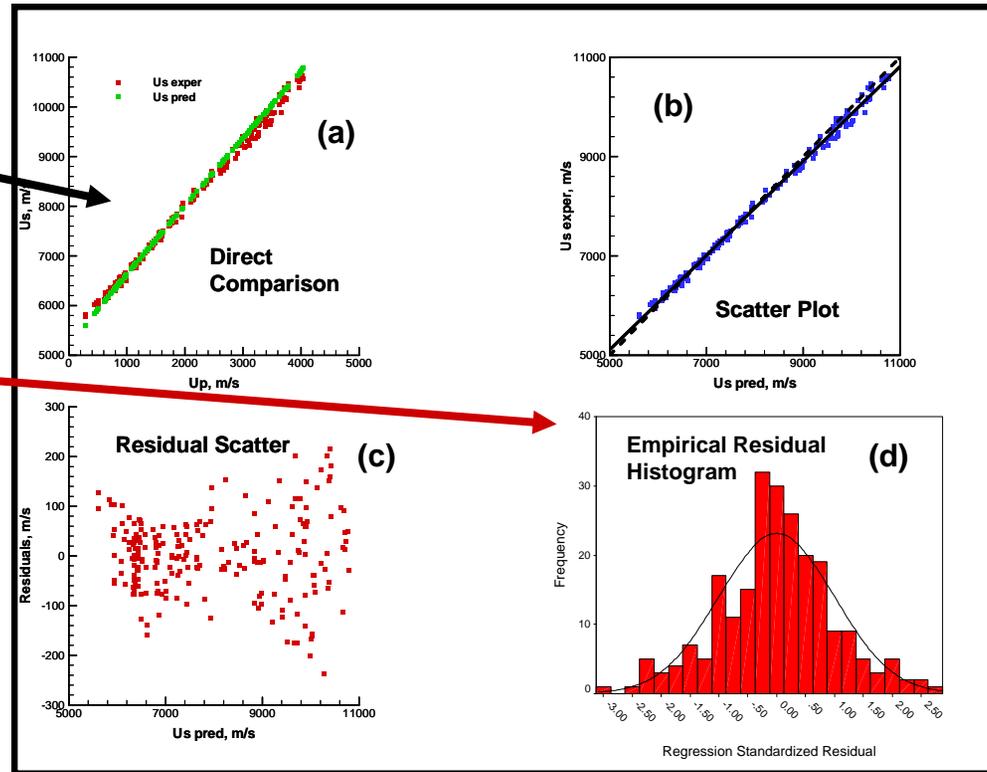
# Uncertainty analysis and quantitative validation metrics move us from “looks pretty good” to “something is wrong” ...

SAND2005-0945C

Comparison of calculations and experimental data look good here.

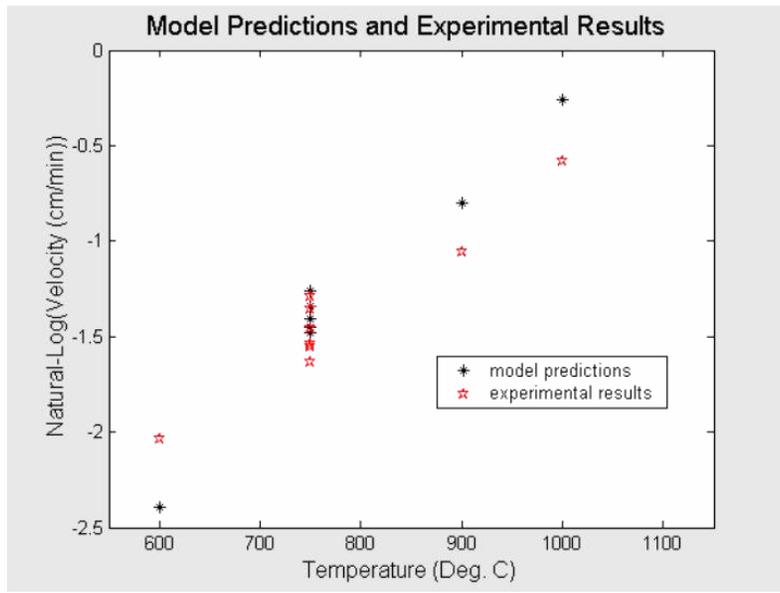
Statistical analysis of the differences shows errors are too large.

Hills and Trucano (2001), “Statistical Validation of Engineering and Scientific Models with Application to CTH,” SAND report.

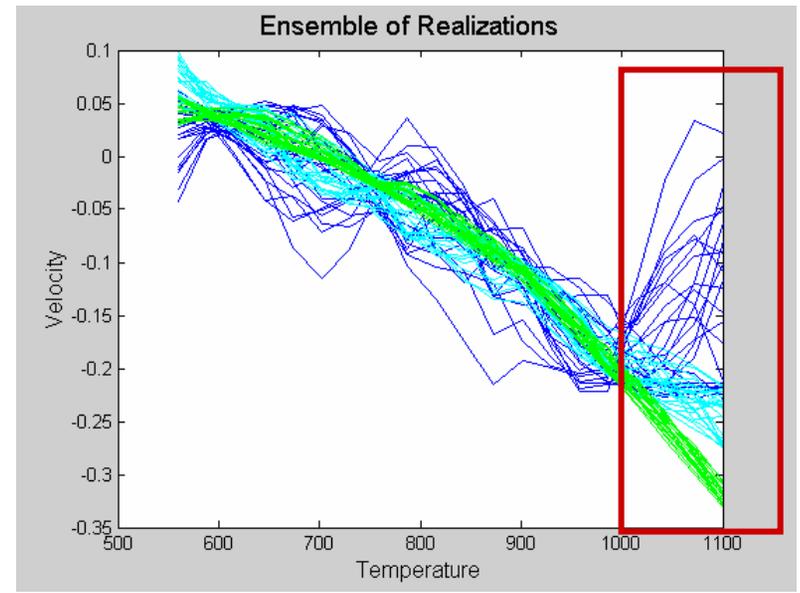


# Exploring the probabilistic overlap between the computational model and the experimental data.

Brian M. Rutherford, Kevin J. Dowding, "Model Validation and Model-Based Prediction Polyurethane Foam Case Study," Sandia, SAND2003-2336. (Decomposition front velocity as a function of forcing temperature in an organic foam.)

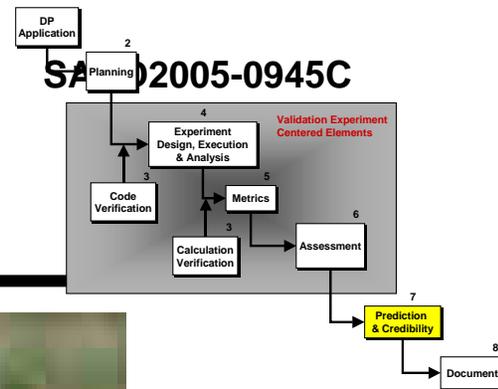


Raw overlap of computational and experimental variability. The difference is interpreted as a random process.



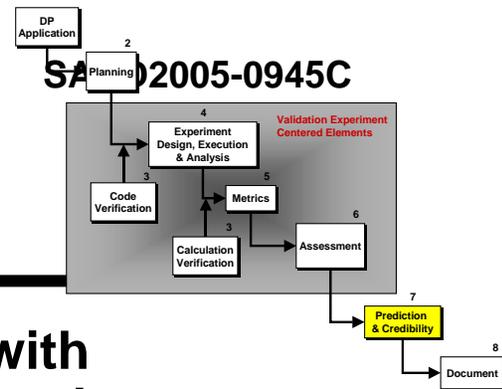
Stochastic modeling of model-experiment difference process, based on response surfaces. Outlined region is extrapolation and is particularly sensitive to response surface form.

**Element 7: Understanding predictive credibility of the code for the defined application is the real goal.**



**Don't ask the question if you won't like the answer.**

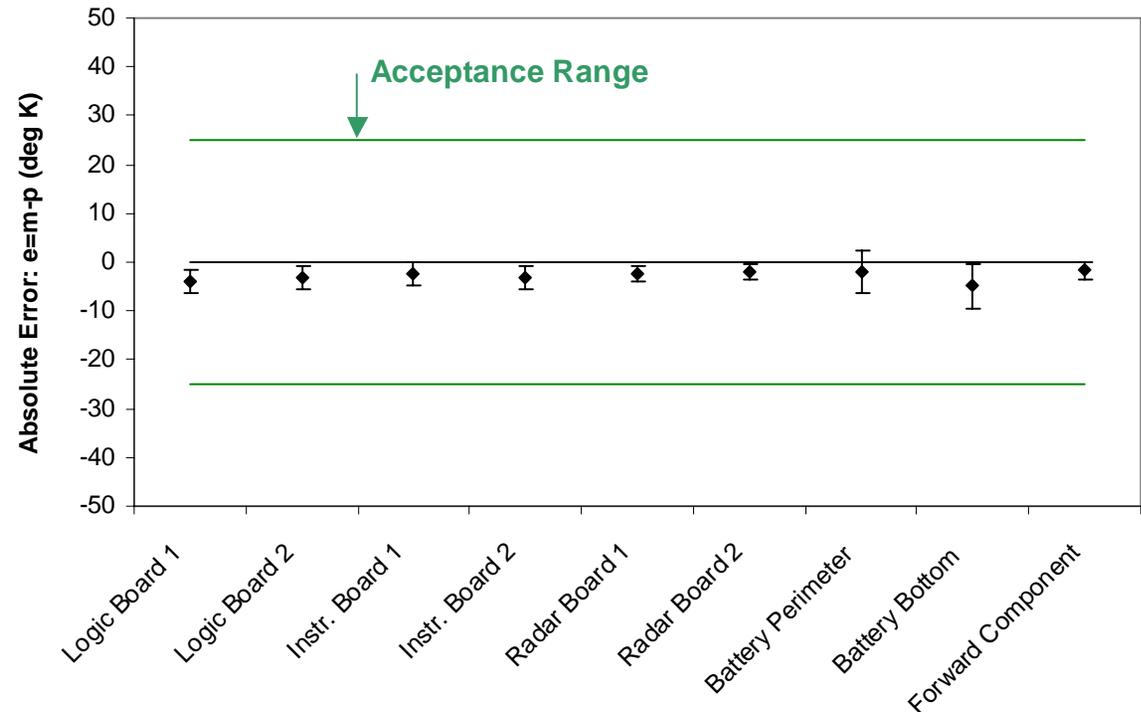
# Assessing predictive credibility is not simple.



- Assessing whether a calculation “agrees” with experimental data to some degree of success is a [local inference](#).
- Understanding what this assessment means in terms of the intended application of the code is a [global inference](#).
- All validation tasks associated with a specified application of a code point at this goal.
- The question is simple, but hard to answer:
  - “What does the result of the validation task – in particular the metric comparisons and their assessment – tell me about using the code?”
  - If a conclusion can not be drawn, even “I don’t know for the following reasons...”, then doubt is cast on the value of the validation task.

# Quantitative metrics actually help ...

- Ideally, the “zero line” should pass through the  $\pm 2\sigma$  error bars.
- Not true here (from statistical analysis); therefore, we conclude that model has a bias.
- Practically, we can say with high confidence that the predictions fall within customer supplied acceptance range.



- **Calibration**: Improve the agreement of codes with defined benchmarks (verification or validation, for example) through improved choices of input parameters.
  - Example: adjusting the mesh to achieve better agreement with experimental data is calibration.
- Calibration in the face of model uncertainty is a very interesting problem.
  - V&V highlights and quantifies model uncertainty.
  - Does incorporating model uncertainty improve predictability?
- One perspective is the Bayesian interpretation of model “improvement” through new information generated by V&V.
- Another perspective is application of Bayesian methods in optimization under uncertainty (optimization using imperfect codes).

- Laura Swiler and I are working on this, with an emphasis on optimization and the connection to V&V.
- Also a hot topic at LANL.
- Starting point is: M. C. Kennedy and A. O'Hagan (2002), "Bayesian Calibration of Computer Models," (with discussion), Journal of the Royal Statistical Society (Series B), Vol. 58, 425-464.
- The key starting point is to transform a relationship of the form:

$$\text{Observation}(x) \sim \text{Code}(x,p) + \text{Error}(x)$$

(where Error is a random process characterizing observation error)

to one of the form:

$$\text{Observation}(x) \sim \text{Code}(x,p) + \text{Discrepancy}(x) + \text{Error}(x)$$

and calibrate this.

- V&V help constrain, if not quantify, the discrepancy

# Example:

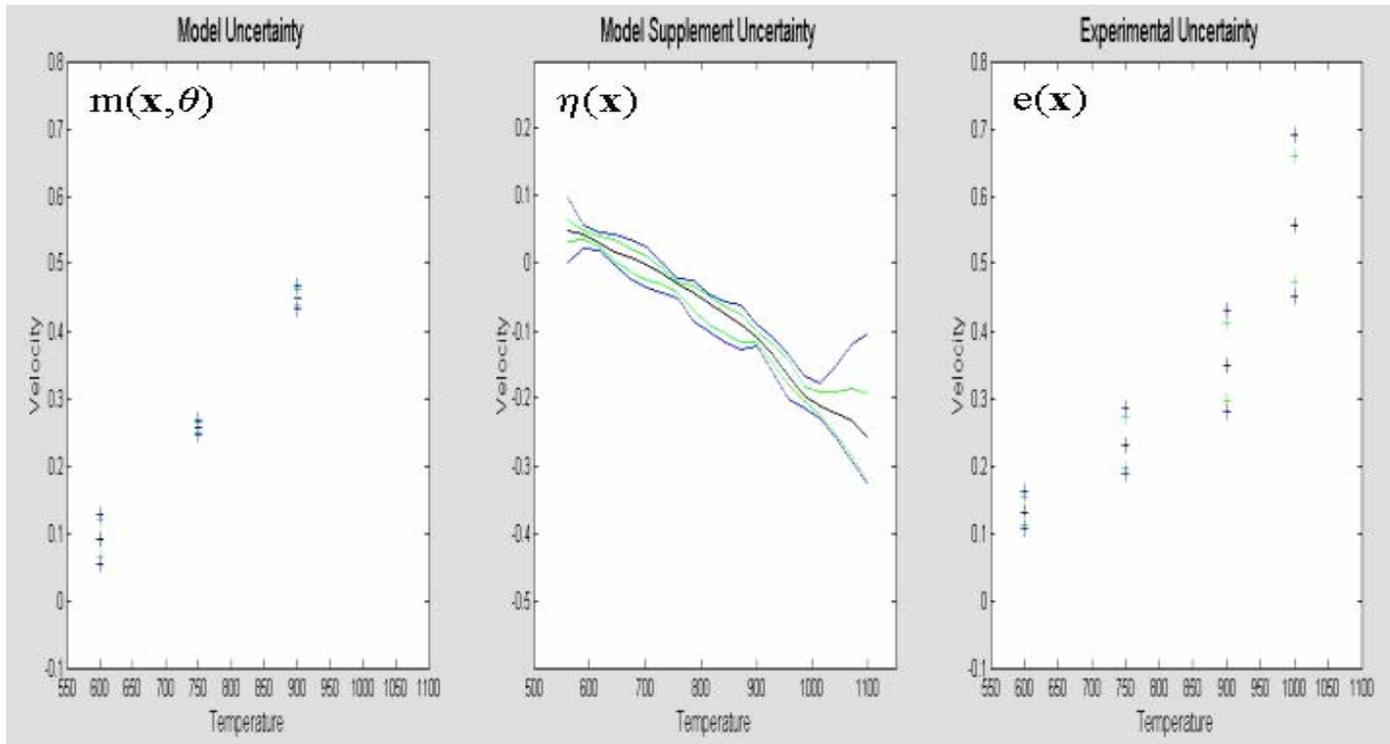
SAND2005-0945C

Returning to Rutherford and Dowding: they use a form of model discrepancy in their development of the stochastic model for the validation metric that I showed on slide 37.

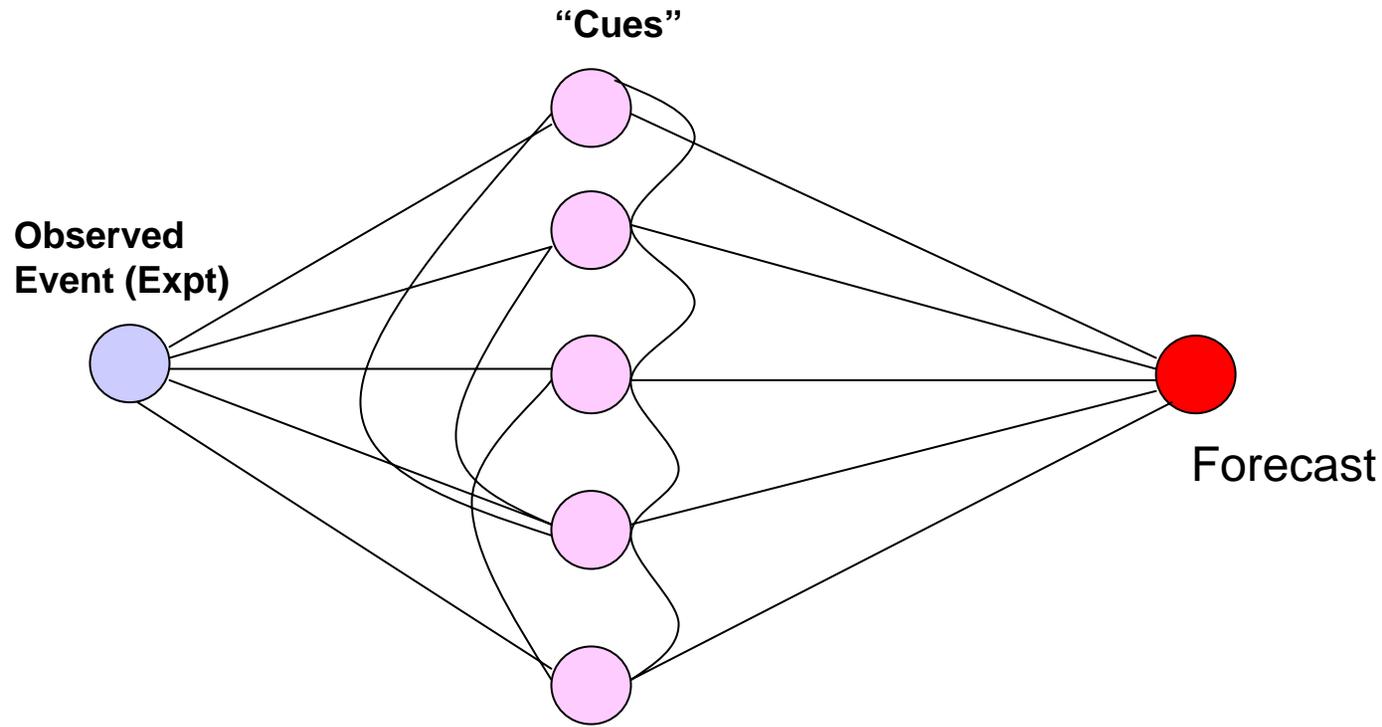
**Code**

**Discrepancy**

**Error**



- 
- We can define necessary conditions for validation; but what are *sufficient* conditions?
  - Sufficiency in one respect is a expected utility maximization problem. Do Arrow-like impossibility theorems make this task hopeless?
  - From a more general multi-objective OUU perspective, validation sufficiency may be more of a “satisficing” problem – “good enough” may be far removed from rigid notions of computational and physical fidelity.
    - Use of reduced-order models in national security problems is one example.
    - Social modeling is another.
  - Validation is currently a matter of judgment. Why not embrace this?



- **Forget causality, everything is correlation.**
- **Methods for assessing forecast skill.**
  - **Implies “sufficient validation” means “sufficient skill”?**
- **R. Cooksey, Judgment Analysis, Academic Press, 1996.**