SPIDER

On the Utility of Proof-based Evidence: Experiences from the Formal Analysis of Fault-Tolerant Systems

Paul S. Miner

p.s.miner@nasa.gov

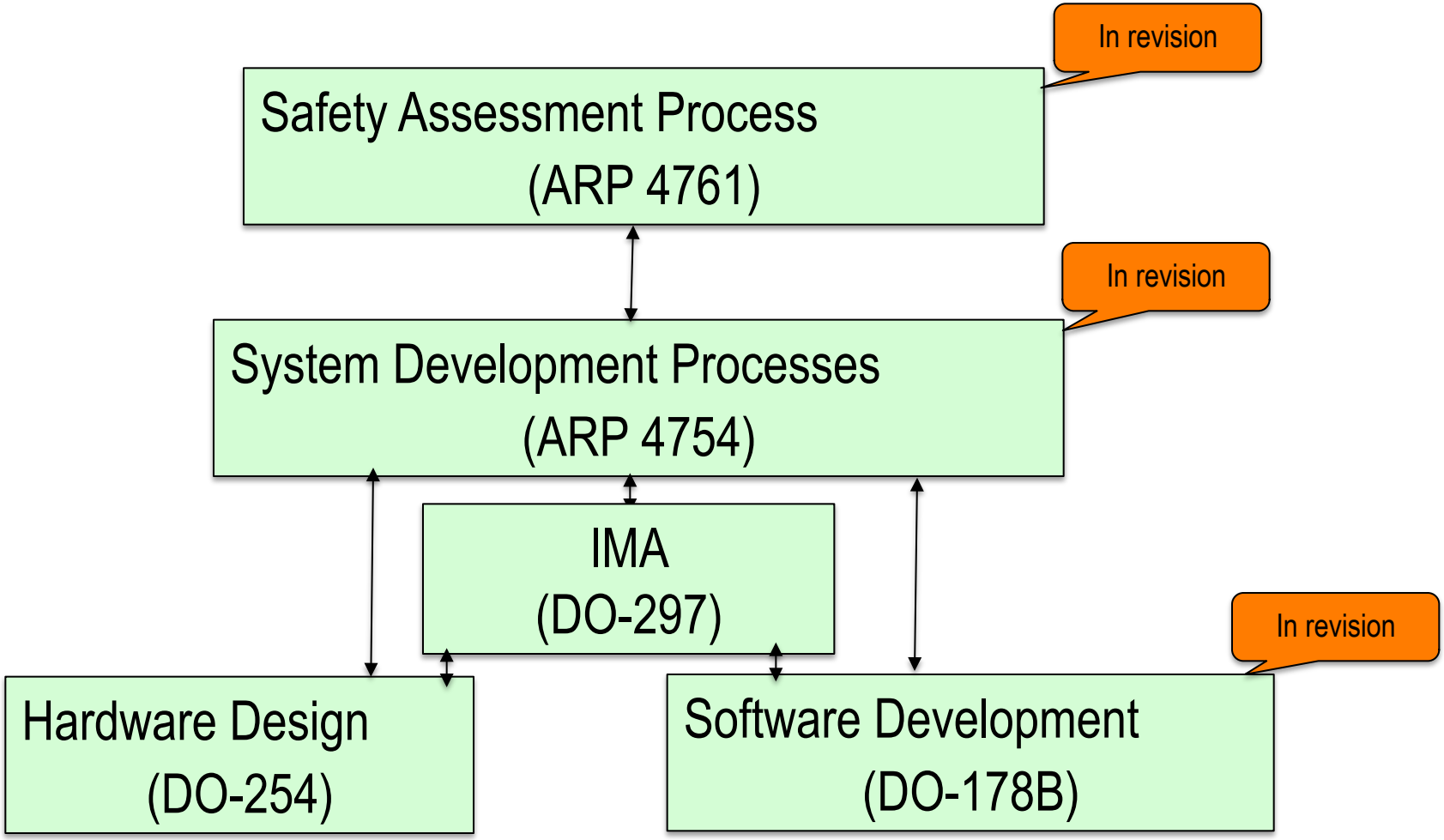
7 December 2010

Engineering of Complex Systems

"As Eads, Flad, and Pfeifer knew, the essence of sound engineering lay in clearly stating the assumptions upon which calculations are based so that they may be checked at all times for lapses in logic and other errors. It is thus imperative that engineering premises be set down clearly, and that the calculations that follow be systematically and unambiguously presented, so that they may be checked by another engineer with perhaps a different perspective on the problem."

Henry Petroski, "Engineers of Dreams", p. 44, Alfred A. Knopf, New York, 1995.

Relevant Civil Aviation Standards



Formal Methods Case Study

- Design part of a fault-tolerant Integrated Modular Avionics (IMA) architecture concept
  - Fault-tolerance is inherently complex
  - System description is compact
- Case study applied to the Reliable Optical Bus (ROBUS) of the Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER)

SPIDER Contributors

Architecture and Protocol Development - Paul Miner, Wilfredo Torres, Mahyar Malekpour

Hardware Design and Implementation - Wilfredo Torres, Mahyar Malekpour

Formal Verification – Paul Miner, Jeff Maddalon, Alfons Geser (was NIA, now HTWK Leipzig), Lee Pike (now with Galois)

Integrated Modular Avionics

- Integrated architectures provide (computational) resources for several distinct aircraft functions
  - Aircraft functions have different levels of criticality determined by the potential severity of failure effects
  - If functions of different criticality share (computational) resources, then architecture must manage resources
    - Otherwise, failures of non-critical functions can prevent critical functions from accessing necessary resources
- Correct operation of IMA platform resource management is at least as critical as the most critical hosted application

ARINC 653

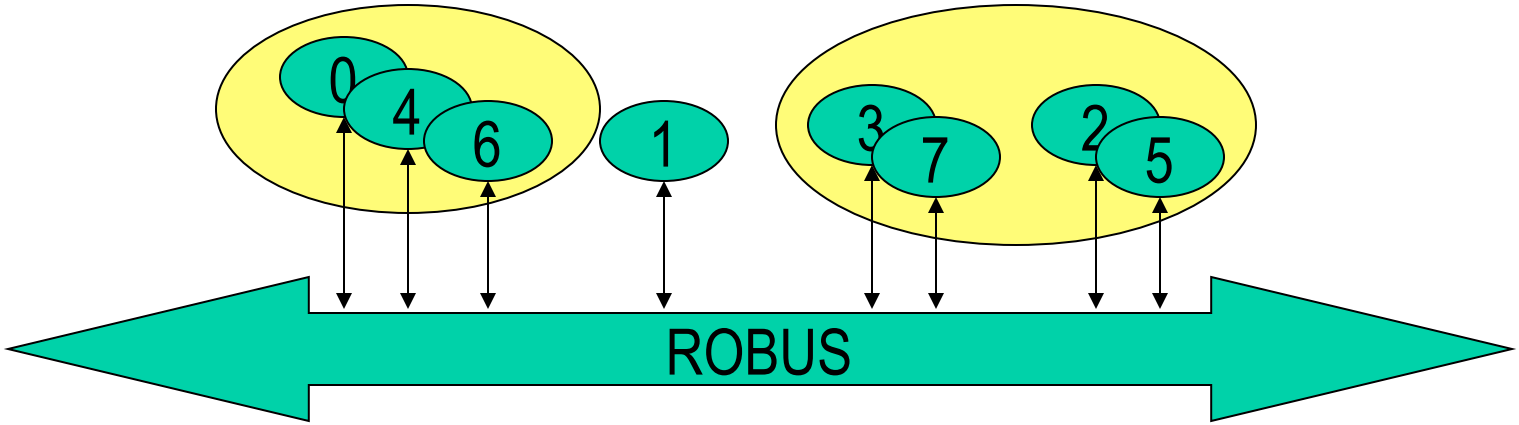
- Real-Time Operating System (RTOS) standard for safety-critical systems
- Provides for robust partitioning of computational resources on a single processor
  - Enforces guaranteed (scheduled) processor time allocation
  - Preserves integrity and availability of allocated memory
- Several commercially available options
  - Green Hills, Wind River, LynuxWorks, …
- Provides protection against software faults in other applications
  - No protection against hardware or network failures

Safety-Critical Network Partitioning

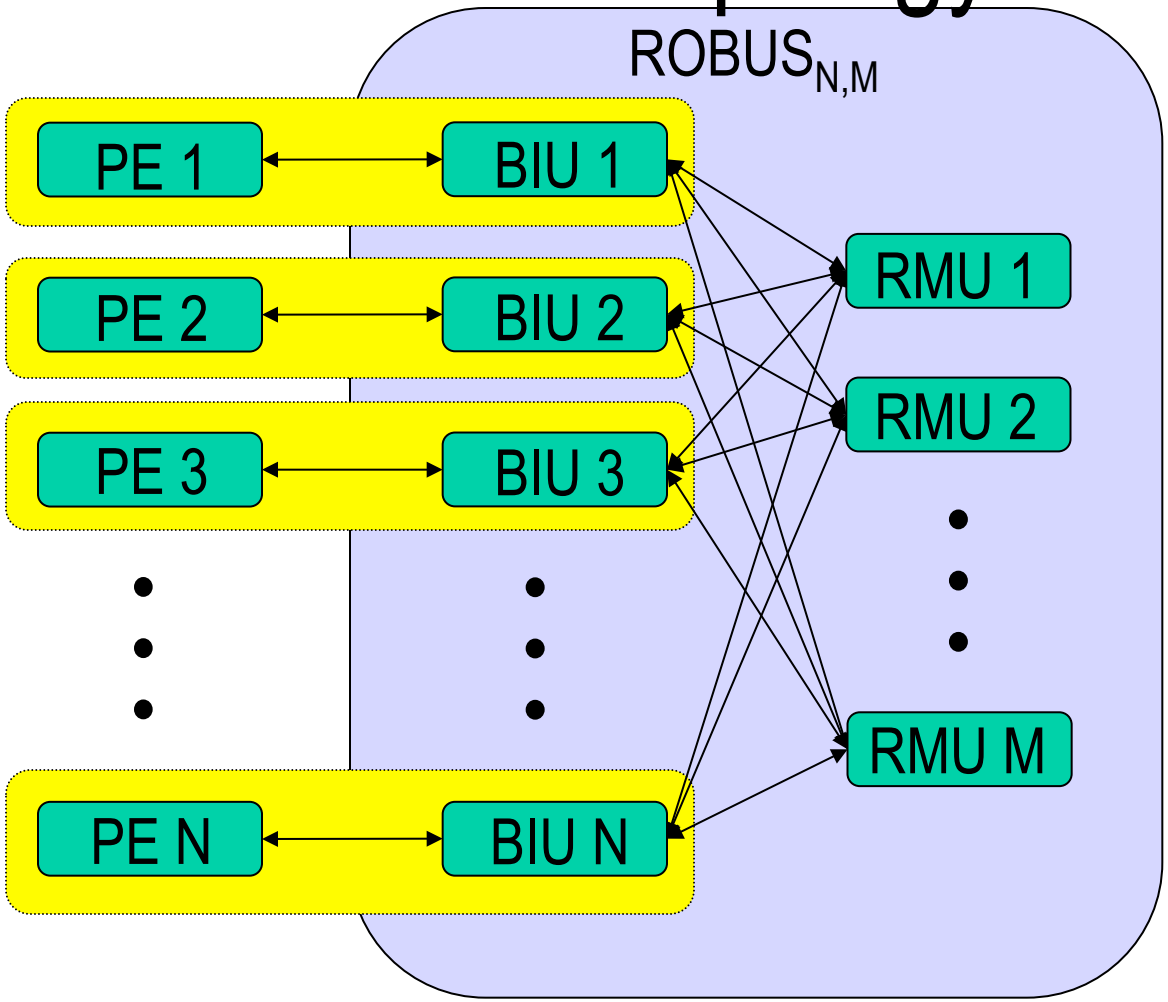
- For safety-critical systems, we also need guaranteed communication channels between redundant processing sites
- Protection against random hardware faults
- Nodes on the network can fail in arbitrary ways
- Communication mechanism must provide guaranteed communication integrity, bandwidth and latency, even if some attached devices are actively misbehaving

Logical view of SPIDER

(Sample Configuration)



ROBUS Topology



ROBUS Requirements

- All fault-free PEs observe identical message sequences
  - If the source is also fault-free, they receive the message sent
- ROBUS provides periodic synchronization messages
  - The PEs are synchronized relative to this
- ROBUS provides correct and consistent ROBUS diagnostic information to all fault-free PEs

All protocols analyzed with respect to the same maximum fault assumption

ROBUS Protocols

- Interactive Consistency (Byzantine Agreement)
  - loop unrolling of classic Oral Messages algorithm
  - Inspired by Draper FTP
- Clock Synchronization
  - adaptation of Srikanth & Toueg protocol to SPIDER topology
  - Corresponds to Davies & Wakerly approach
- Distributed Diagnosis (Group Membership)
  - Initially adapted MAFT algorithm to SPIDER topology
    - Depends on Interactive Consistency protocol
  - Verification process suggested more efficient protocol
    - Improved protocol due to Alfons Geser
    - Suggested further generalizations

Other Requirements?

- Primary focus is on fault-tolerance requirements
  - Other requirements deliberately unspecified
    - Message format/encoding
    - Performance
  - These are implementation dependent
- Product Family
  - capable of range of performance
  - trade-off performance and reliability
  - Formal analysis valid for any instance

Unified Consensus Protocol

- All SPIDER/ROBUS fault-tolerance protocols may be realized using different instances of a single abstract protocol
  - Generalization of Davies & Wakerly [IEEE ToC 1978]
  - Abstracted protocol consists of a cascade of data exchanges with a middle value selection vote at each stage
- Formal analysis using PVS presented at FORMATS-FTRTFT September 2004
  - Using Thambidurai & Park hybrid fault assumptions
    - Good, Benign, Symmetric, Asymmetric
  - Paper and PVS models available from SPIDER web site
    - http://shemesh.larc.nasa.gov/fm/spider/
- Subsequently generalized to encompass weaker fault assumptions and more flexible voting strategies
  - Added convergent voting for approximate agreement

Some Distributed Consensus Protocols

- Clock Synchronization
- Group Membership (Distributed Diagnosis)
  - Need to disambiguate faults in presence of imprecise information
- Interactive Consistency/Source Congruency
- Reintegration/transient fault recovery
- Start-up/Re-start

Want all of these in presence of specified number of faults

Distributed Consensus Properties

With respect to some critical {event/data/state/decision} *x*

Validity

All correctly behaving participants perception of *x* is consistent with the correct value of *x* (within some error tolerance)

Agreement

All correctly behaving participants have a consistent perception of *x* (within some error tolerance)

Distributed consensus properties provide a foundation for compositional verification

Multistage Properties

Validity: If there are *enough good* sources at every stage, then all *good* receivers select a value in the range of the *good* sources from the first stage

- Follows from induction on stages and single-stage validity result

Agreement Propagation: If there are *enough good* sources at every stage, and all *good* sources for the first stage agree, then all *good* receivers will agree

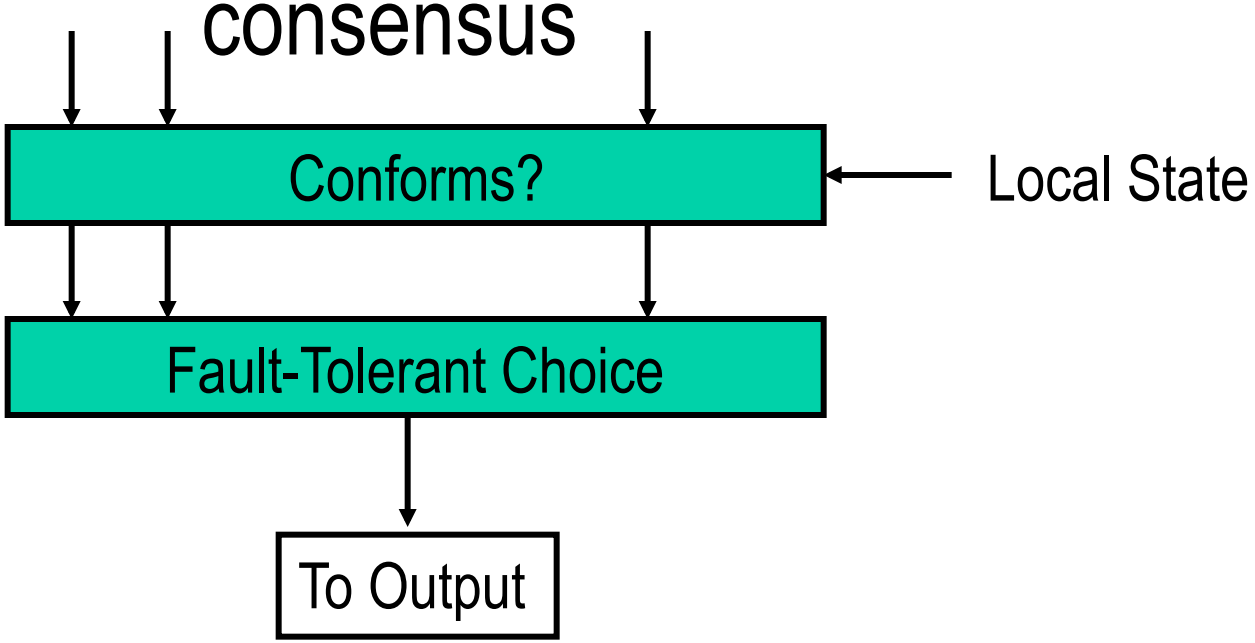
- Follows from induction on stages and single-stage agreement propagation
- Also a corollary of multi-stage validity

Agreement Generation: If there exists a stage with no *asymmetric* sources and there are *enough good* sources at every subsequent stage, then all *good* receivers will agree

- Follows from single-stage agreement generation and multi-stage agreement propagation

Convergence: If there are *enough good* sources at every stage and there exists at least one converging stage, then the multi-stage exchange converges

Abstracted node behavior for generalized consensus



- Input conformance check based on local state (possibly including expected value), message timing, and message format
  - All nonconforming messages are ignored by FT-choice function
- FT choice could be majority, middle value select, ft-midpoint, …

Functional Unit Verification

- A representative data path function is a majority voter
- The function is specified abstractly
  - Existing algorithm in literature: R. S. Boyer and J S. Moore. MJRTY - A Fast Majority Vote Algorithm, 1991
  - Specified and verified this algorithm against the abstract specification using PVS
  - Coded the algorithm in synthesizable VHDL
  - Used review to establish correspondence between PVS model and VHDL code
    - Possible to automate translations

Strength of Formal Verification

- Proofs equivalent to testing the protocols
  - for all specified configurations
  - for all possible combinations of faults that satisfy the maximum fault assumption for each specified configuration
  - for all possible message values
- The formal proofs provide verification coverage equivalent to an infinite number of test cases
  - This only matters if the formal model of the protocols is faithful to the system realization, and that the system satisfies all assumptions necessary for correct operation of the protocols

Independence Assumption

- All redundancy mechanisms assume faults affect redundant pieces (e.g. bits, messages, channels) independently
- System must be engineered to satisfy this assumption (with respect to the covered class of faults)
- For systems based on state-machine replication, there is the notion of a Fault-Containment Region (FCR)
  - Conservative realizations of FCRs provide for both spatial and electrical isolation
  - Less conservative realizations have exhibited violations of independence

Byzantine Faults

- Characterized by asymmetric error manifestations
  - different manifestations to different fault-free observers
  - including dissimilar values
- Can cause redundant computations to diverge
- Can cause distributed decisions to conflict
- If not properly handled, a single Byzantine fault can defeat several layers of redundancy
- Many architectures neglect this class of fault
  - Assumed to be rare or impossible

Byzantine faults are real

- Several examples cited in *Byzantine Faults: From Theory to Reality,* Driscoll, et al. (SAFECOMP 2003)
  - Byzantine failures nearly grounded a large fleet of aircraft
  - Quad-redundant system failed in response to a single fault
  - Typical causes are faulty transmitters (resulting in indeterminate voltage levels at receivers) or faults that cause timing violations (so that multiple observers perceive the same event differently)
- Heavy Ion fault-injection results for TTP/C (Sivencrona, et al.)
  - more than 1 in 1000 of observed errors had asymmetric (Byzantine) manifestations
- STS-124 pre-flight (May 2008)
  - http://www.nasaspaceflight.com/2008/05/sts-124-frr-debate-outstanding-issues-faulty-mdm-removed/

First Picture of a Byzantine Fault?

At 12:12 GMT 13 May 2008, a NASA Space Shuttle was loading hypergolic fuel for mission STS-124 when a 3-1 split of its four control computers occurred. Three seconds later, the split became 2-1-1. During troubleshooting, the remaining two computers disagreed (1-1-1-1 split). ***Complete system disagreement.*** But, none of the computers or their intercommunications were faulty! The ***single fault*** was in a box (MDM FA2) that sends messages to the 4 computers via a multi-drop data bus that is similar to the MIL STD 1553 data bus. This fault was a simple crack (fissure) through a diode in the data link interface.

Figure 1. Two views (90 degrees apart) of a fissure that appears to go through the silicon dice - Red arrows.

From Kevin Driscoll's Keynote Presentation at SAFECOMP 2010 (with permission)

Hybrid Fault Assumption

- Architectures designed assuming only Byzantine faults can be brittle
  - David Powell, *Failure Mode Assumptions and Assumption Coverage*, FTCS-22, 1992
- We must handle Byzantine faults
  - But, other failure modes *are* more common
- Systems designed against hybrid fault models gracefully accommodate either a few Byzantine faults or combinations of several less severe faults
  - Avoids assumption coverage difficulties

Hybrid Fault Hypothesis

- Byzantine faults are real, protocols must be designed against this worst case fault manifestation
- To tolerate *f* Byzantine faults requires
  - 3f + 1 independent fault containment regions (FCR),
  - 2f + 1disjoint communication paths between every pair of FCRs, and
  - f + 1 stages of communication
- Easier to handle faults occur more frequently
  - Hybrid fault models established for various manifestations of misbehavior
    - transmission (incorrect value) vs. omission (fail-silent, fail-stop),
    - symmetric (same to all receivers) vs. asymmetric
    - multiple distinct value asymmetry vs. at most one value
  - Omission faults can be ignored, provided there is still a good source
  - Asymmetric/transmission combination includes Byzantine manifestations
- *Classification is a function of state of receivers!*

Further Generalizations?

Fault-Tolerant Avionics SOA:

Competing Philosophies

Asynchronous

- Common in primary flight control systems
- Unsynchronized, but common frame rate
  - Coarse synchronization needed for mode change (I.e. exact agreement)
- Imprecision of sensor data
- Approximate Agreement w/ Threshold voting
  - Incorrect threshold is a common source of failure (diagnostic ambiguity)

Synchronous

- Common in flight management systems
- Precise Synchrony
- State replication
- Reliable Data Distribution
- Exact agreement / Majority voting
- Approximate agreement only for synchronization

All models are wrong but some are useful

George Box

Modeling Questions

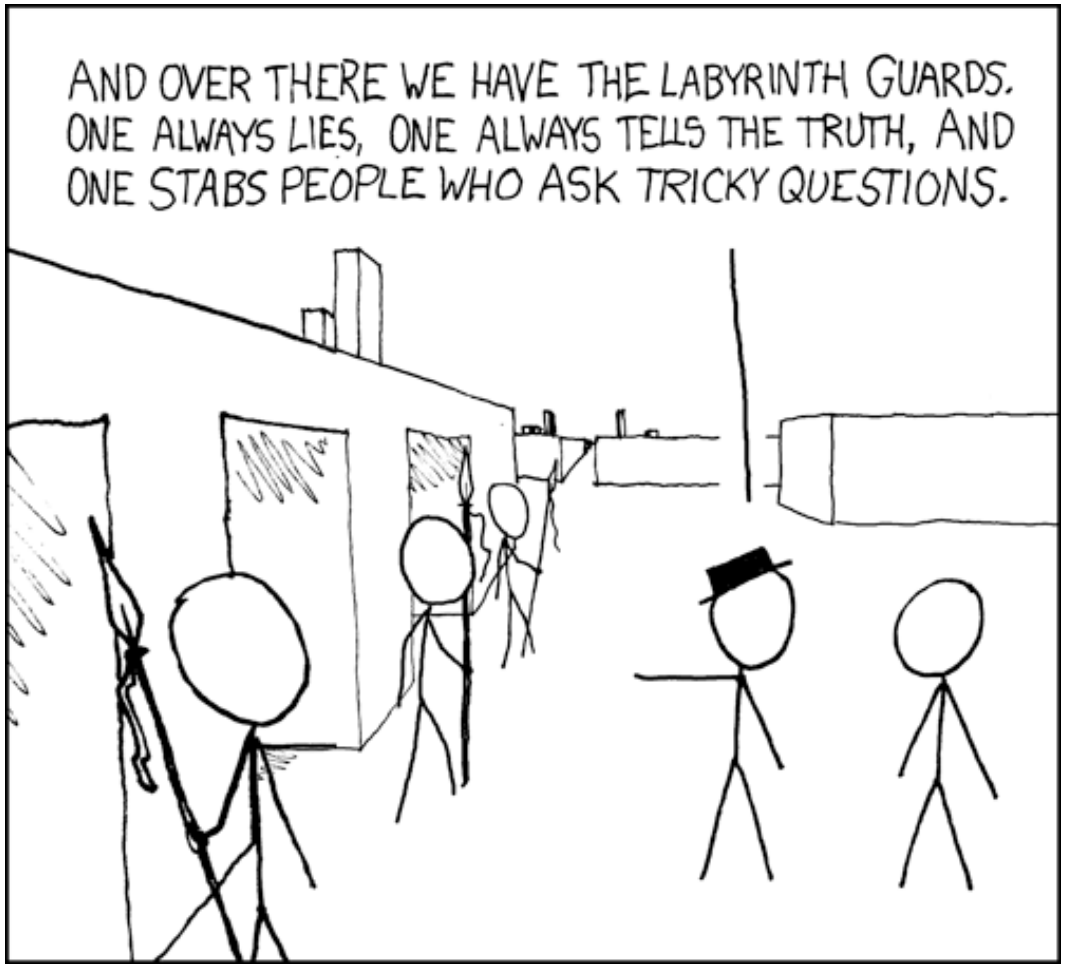
- Are the models meaningful?
  - Are abstractions valid?
    - e.g. synchronous composition, functional abstraction
  - Are assumptions satisfiable?
    - Is there a typical case?
    - Are assumptions true for initial conditions?
    - Are assumptions preserved through execution of protocol?

More Modeling Issues

- How are the formal models related to the modeled artifact?
  - Extraction of model from VHDL source?
  - Translation of model to VHDL?
  - Review?

Formal Proof Issues

- Have you proven the claim you intended to prove?
  - Sanity checks:
    - For each hypothesis, demonstrate why proof fails when hypothesis removed (may be an informal argument)
    - Confirm that you haven't assumed the conclusion
    - Confirm that models of system components only have access to same set of data as the modeled component

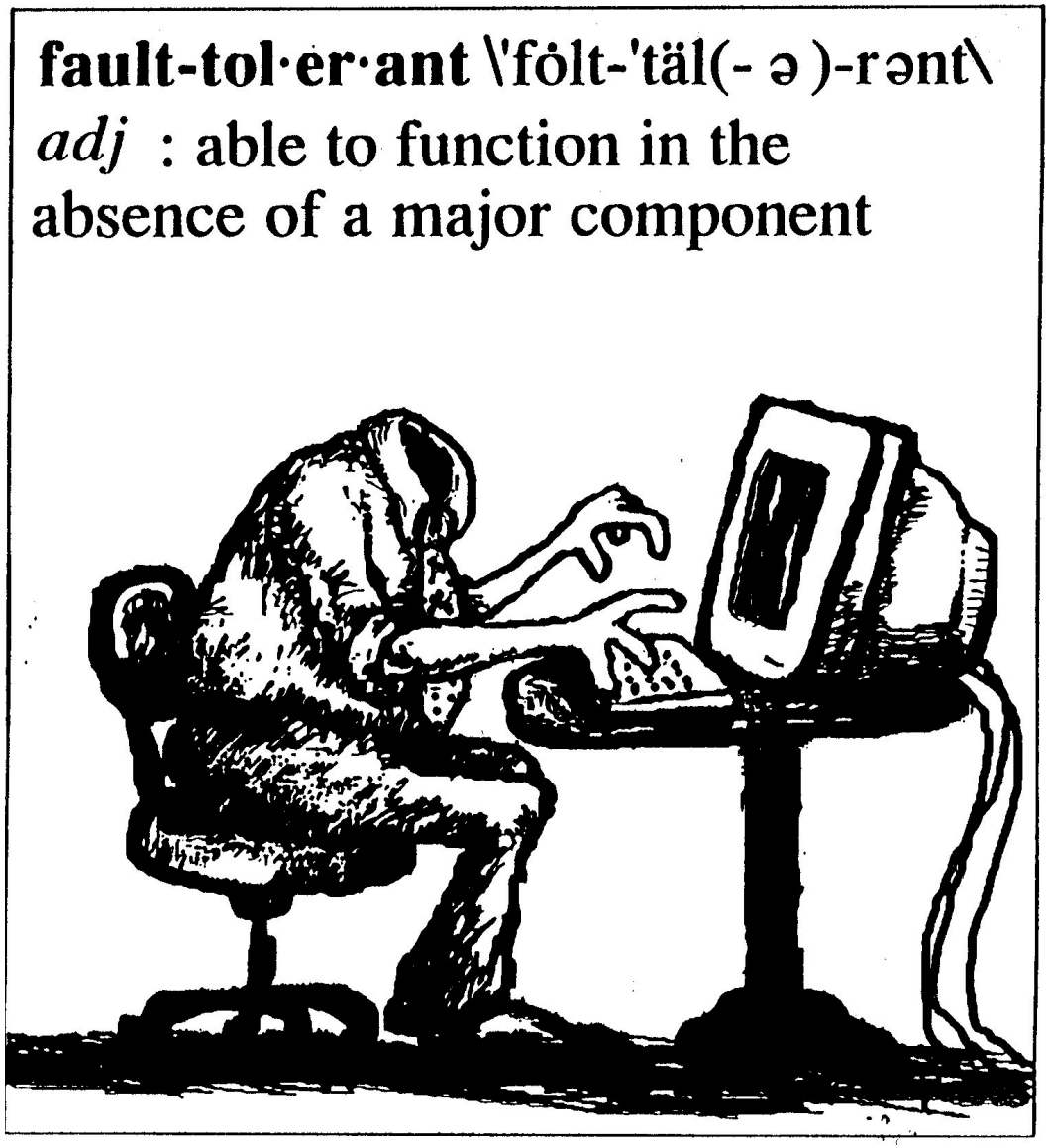


Questions?

Downloaded from http://xkcd.com/246/

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

– Albert Einstein


fault-tol·er·ant \'fȯlt-'täl(- ə)-rənt\
adj : able to function in the absence of a major component