Ultradependability

18-849b Dependable Embedded Systems

Michael Scheinholtz

Thursday, March 11th, 1999

Required Reading: Ultradependable Architectures, Siewiorek et al

Book: The Art of Systems Architecting

Carnegie Mellon

Overview: Ultradependability

- **Introduction**
  - Multi-disciplinary design
- **Key concepts**
  - Long hours and many machines
  - Fixing the weak link
  - How do you know it works?
- **Tools / techniques / metrics**
  - Hardware Techniques
  - Software?
- **Relationship to other topics**
- **Conclusions & future work**

YOU ARE HERE MAP

Description of Topic

- **Ultradepenable system: A system designed to run without any defects in its lifetime, or the lifetime of its fleet.**
- **Ultradependable Failure Rates**
  - 10^{-6} to 10^{-9} failures per hour.
  - 10^{-6} requires 1000 hours of testing on 1000 units.
- **Key points**
  - Many embedded systems are deployed in the hundred millions.
  - Redundancy may be limited due to cost factors
  - Testing an ultra-quality system can only prove the system doesn't meet the targeted failure rate.

Long hours and many machines

- **Failure Rates and large fleets**
  - Assume airlines have a failure probability of 10-9 failures per hour
    - Yields one failure per 73 years from 13.7 million fleet hours
  - 200 million cars in U.S. operating for more hours with the same failure rate
    - 82 failures per year (one every 4.5 days)
  - Large fleets mean that the same design will encounter exceptional conditions more often.

Fixing the weak link

- **Failure rates assuming near perfect environment**

Source of Error	Tandem (Gray 1985)	Tandem (Gray 1987)
Hardware	.18	.19
Software	.26	.43
Maintenance	.25	.13
Operations	.17	.13
Environment	.14	.12

- **What causes these failures?**
  - Amdahl's Law applied to dependability
    - Software is becoming the main source of failure
  - Tandem machines operate in a near perfect environment, normally the environment would have more of an impact
- **Fixing software**
  - Must use fault avoidance techniques

How do you know it works?

- **Testing gives a lower bound for dependability**
  - Hard to find errors when the failure rate is 10^{-9}, but if an error is found that means the system isn't ready.
- **Could use fault injection to increase error rate**
  - Again, this doesn't verify ultradependability.
- **Methods in use now**
  - Use reliable, proven components
  - Formal methods may prove that system meet requirements (then just get requirements correct…)
  - Use quality development process
    - Similar to quality engineering
    - Refine development process to assure quality product
    - Everyone one on the team is responsible for product quality.

Tools / Techniques

- **Hardware**
  - Proven components
  - N-versioning and redundancy work for hardware. But….
  - With the increased complexity of microprocessors, hardware starts to look like software.
- **Software**
  - Fault tolerance methods don't work well.
  - Check pointing can help, but requires correct requirements
  - Formal methods can also help, but require correct requirements
- **The design is the key**
  - High quality design process can help make high quality product
  - Refine process overtime.
  - Overlapping ultradependability techniques can help. Use diversity.

Metrics

- ????
- Very difficult to measure ultradependability
- Testing only reveals that a system is not ultradependable
- The best that can be done now is validate the process used to create the system.
- Systems can be refined as errors are found in the field.

Relationship To Other Topic Areas

- **Ultradependability relies on these areas**
- **Multi-disciplinary Design**
  - Ultradependability requires quality in every aspect of the system.
- **Software Reliability**
  - This is the bottleneck
- **Safety Critical Systems/Analysis**
  - Catches hazards at the design phase.
- **End-of-Life Wearout & Retirement**
  - An important problem for a long-life system
- **Maintenance and Reliability**
- **Diagnosis and Prognosis**
- **System Life Cycle**

Conclusions & Future Work

- **Ultradependability is very difficult, but growing in importance.**
  - Embedded systems have proliferated, and are sometimes responsible for human life
- **Must combine many techniques to achieve ultradependability**
  - Quality techniques
  - Proven components
  - Refinement over the life of the design
  - Diversity
- **Software is becoming the biggest stumbling block**
- **A high level of quality must be maintained in all phases of design.**

Ultradependable Architectures

- **"Produce a system that neither crashes nor fails during a 10 year operational life. Furthermore, these systems should require less than 5% more resources..."**
- **These bullets summarize the major points/approach**
  - Many different techniques must be combined to create ultradependable systems.
  - Key research areas
    - Design needs to be better understood.
    - Better analysis tools.
    - Better testing and validation/verification.
    - Improve security, fault tolerance, and parallel processing.