Reliability

Terminology of failure or ‘Failing terminology’?

Based on a set of powerful and diverse tools...

Observable failure modes

Faults during different phases of design
• Inconsistent or inadequate specifications
• Program errors
• Component & communication system failures

Faults in the logic domain
• Non-termination / completion
• Range violations and other inconsistent states
• Value violations and other wrong results

Failure modes
• Systems ‘frozen’ in a deadlock state, blocked for missing input, or in an infinite loop
• Intermittent faults
• Range violations and other inconsistent states
• Value violations and other wrong results

Achieving reliability

System identification
• Static applications specifications.
• Physical sensors and converters constraints.
• Constraints of the employed, controlled network.
• Constraints of the environment.

Fault avoidance
• Full fault tolerance
• Fail safe
• Constrained degradation (fail safe)

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Hardware redundancy
• Adding extra hardware resources
• Software redundancy

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Faults in the time domain
• Observable failure modes
• Non-termination / completion
• Range violations and other inconsistent states
• Value violations and other wrong results

Final removal

Failing systems:
• Systematic removal of the system
• Systematic removal of the environment

N-Modular Redundancy (NMR)

Fault avoidance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Reliability, failure & tolerance

Faults during different phases of design
• Inconsistent or inadequate specifications
• Program errors
• Component & communication system failures

Faults in the logic domain
• Non-termination / completion
• Range violations and other inconsistent states
• Value violations and other wrong results

Achieving reliability

System identification
• Static applications specifications.
• Physical sensors and converters constraints.
• Constraints of the employed, controlled network.
• Constraints of the environment.

Fault avoidance
• Full fault tolerance
• Fail safe
• Constrained degradation (fail safe)

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Hardware redundancy
• Adding extra hardware resources
• Software redundancy

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Reliability, failure & tolerance

‘Terminology of failure’ or ‘Failing terminology’?

Based on a set of powerful and diverse tools...

Observable failure modes

Faults during different phases of design
• Inconsistent or inadequate specifications
• Program errors
• Component & communication system failures

Faults in the logic domain
• Non-termination / completion
• Range violations and other inconsistent states
• Value violations and other wrong results

Achieving reliability

System identification
• Static applications specifications.
• Physical sensors and converters constraints.
• Constraints of the employed, controlled network.
• Constraints of the environment.

Fault avoidance
• Full fault tolerance
• Fail safe
• Constrained degradation (fail safe)

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe

Hardware redundancy
• Adding extra hardware resources
• Software redundancy

Fault tolerance
• Full fault tolerance
• Constrained degradation (fail soft)
• Fail safe
Reliability

Fault tolerance

Triple Modular Redundancy

Example
- 3 identical primary flight computers distributed in the Boeing 777, each consisting of:
  - Independent power sources and inertia measurements.
  - Code built by 3 different Ada compilers.
- The project included some enforced diversity elements (a 64-page document).

Fault tolerance

The six-language project

The specification of software often requires the use of several different languages to ensure that different components of a system are developed independently.

N-version programming

N-version programming – Voting issues

- Voting issues: Violation of independence and erroneous outcomes.
- N-version programming: Independence and error detection.
- N-version programming: Other issues: Delayed provision of results.

Dynamic redundancy

Four constituent phases (Anderson and Lee, '90):

1. Error detection
   - Detects and responds to erroneous essential system failures.

2. Damage confinement and assessment
   - Diagnoses the damage, which occurs between the instantaneous error occurrence and:
     - Error recovery
     - Inability to prevent the propagation of the damage.

3. Error treatment
   - Inability to prevent the propagation of the damage.

N-version programming

Whole system failure probability:

\[ P_{0} = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \]

Reliability

N-version programming – Voting Issues

Voting failures

\[ P_{V} = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \]

Reliability

Dynamic Redundancy

Failure categories

Average failure rate:

\[ \lambda_{ave} = \frac{1}{T} \int_{0}^{T} \lambda(t) dt \]

Reliability

N-version programming

Voting issues

- Voting issues: Violation of independence and erroneous outcomes.
- N-version programming: Independence and error detection.
- N-version programming: Other issues: Delayed provision of results.

Dynamic redundancy

Four constituent phases (Anderson and Lee, '90):

1. Error detection
   - Detects and responds to erroneous essential system failures.

2. Damage confinement and assessment
   - Diagnoses the damage, which occurs between the instantaneous error occurrence and:
     - Error recovery
     - Inability to prevent the propagation of the damage.

3. Error treatment
   - Inability to prevent the propagation of the damage.

4. Fault treatment
   - Requires the re-computation of the system.

Safety and Dependability

Further refinements in the design (fault tolerant) design:

Restrict & Formalize

Restrict: Formalize, ... ?

Restrict

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Ada Zero Footprint profile

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Restrict

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Ada Zero Footprint profile

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Restrict

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Ada Zero Footprint profile

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Restrict

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Ada Zero Footprint profile

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Restrict

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.

Ada Zero Footprint profile

- No testing: The system is not tested in any way.
- No absence of faults: The system is not fault-free.
- No absence of failures: The system can fail.
- No absence of damage: The system can be damaged.
- No absence of error: The system can contain errors.
Ada Ravenscar profile

- Task type and object declarations at the library level
- Task discriminants
- Library level Protected objects
- Library level Protected objects

Ada Certification profiles

- Profile targeted to specific verification processes, e.g.
- D11-176M (DO-176C) (not for flight critical software)
- Federal Aviation Administration (FAA)
- European Aviation Safety Agency (EASA)

Temporary Logic

- Expanding predicate logic
- Adding a concept of ordering for events and states.

Real-Time Logic

- Suitable for event driven system, reactive systems

Linear Temporal Logic of Real Numbers (LTR)

- Expressiveness
- Embed (more) logic into your current programs

Reliability

- Formalize
- Event-Clock Temporal Logic
- Metric-Interval Temporal Logic

Termination

- Task termination is generally considered to be an error for real-time programs which are long-running and defines all of its tasks at start-up.

Fault avoidance

- N-version programming, and dynamic redundancy in software design.

Redundancy

- N-version programming, and dynamic redundancy in software design.