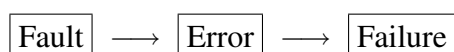


## 2 Dependability

### 2.1 Terminology

**Systems.** A *specification* describes the ideal behavior of a *system* at its *interfaces*. A system is called *dependable* if it follows the specification as closely as possible, even if faults occur. Typical faults are when some of parts of the system fail or when the environment behaves differently than assumed.

Three causally related terms have been defined to describe dependable systems [ALRL04]:



**Fault:** (Hypothetic) Cause of an error.

**Error:** Internal system state that does not correspond to specification, not visible at interfaces.

**Failure:** Deviation of system from specification at interfaces.

Recursive!

**Example 1.** In a computer system, fan in power supply congested by dust, airflow massively reduced, fan not effective, power supply overheats, some part burns out, system loses power, computer stops working.

- Fan system: dust = fault, reduced airflow = error, no cooling = failure.
- Power supply system: no cooling = fault, overheating = error, loss of power = failure.
- Computer system: broken power supply = fault, no power on mainboard = error, computer stopping = failure.

**Example 2.** RAID storage system (bits with ECC, disks with RAID mirroring).

**Example 3.** Typical software vulnerability, buffer-overflow attack exploited by a worm.

### 2.2 Attributes

- Availability — readiness for correct service
- Reliability — continuity of correct service
- Safety — absence of catastrophic failures
- Confidentiality — no unauthorized disclosure
- Integrity — no improper states or state changes

## 2.3 Techniques

**Prevention:** Formal design, access control, good engineering.

**Tolerance:**

- Error & fault detection — Failure detectors, intrusion detection systems.
- Recovery — Isolation, rollback, compensation, fail-over, database transactions (ACID).
- Redundancy — Replication, voting (ECC, RAID), diversity.

**Removal:**

- Formal verification of implementation w.r.t. specification
- Validation of specification w.r.t. real environment
- Fault injection and testing

**Forecasting:**

- Modeling, prediction
- Evaluation, testing (fault trees, attack graphs)

Commercial fault-tolerant systems employ a combination of the above techniques, but focus on sophisticated designs for fault-tolerance through hardware redundancy, combined with isolation and recovery methods for software [BS04]. Examples include the HP Integrity Non-Stop servers (formerly built by Tandem Corp.) [BBV<sup>+</sup>05] and IBM's System z mainframe servers and z/OS operating system (successors of S/390 servers and OS/390) [Hof97].

## 2.4 Measures

Assume that all failures occur independently of each other and with exponential distribution. Define the *failure rate* to be  $\lambda$ ; then the mean time to failure,  $MTTF = 1/\lambda$ . Similarly, define the *repair rate* to be  $\mu$ ; then the mean time to repair,  $MTTR = 1/\mu$ .

Now we have

$$Availability = \frac{MTTF}{MTTF + MTTR}.$$

With a high MTTF, availability can be further increased by reducing the MTTR — this is important for large-scale online services.

**Serial combination.** For a system  $Sys$ , consisting of a “serial” combination of components  $C_1, \dots, C_n$  without redundancy, we have

$$\lambda_{Sys} = 1 - \prod_i (1 - \lambda_i).$$

For  $\lambda_i \ll 1$ , a good approximation is

$$\lambda_{Sys} \approx \sum_i \lambda_i.$$

Hence,

$$MTTF_{Sys} \approx \frac{1}{\sum_i \frac{1}{MTTF_i}}.$$

**Parallel combination.** For a system  $Sys$  in which components  $C_1, \dots, C_n$  are redundant and used “in parallel,” we have

$$\lambda_{Sys} = \prod_i \lambda_i.$$

and thus

$$MTTF_{Sys} = \prod_i MTTF_i.$$

**$k$ -out-of- $n$  combination.** For a system  $Sys$  with  $n$  components  $C_1, \dots, C_n$  with equal failure rate  $\lambda$ , of which  $k$  are needed for  $Sys$  to function, we have

$$\lambda_{Sys} = 1 - \sum_{i=k}^n \binom{n}{i} (1 - \lambda)^i \lambda^{n-i}.$$

**Problems of MTTF & MTTR.** There are 8760h per year. Does a system with MTTF of 500'000h on average run for 57 years without failures, even though its manufacturer specifies a system lifetime of 5 years? No. MTTF and MTTR are only statistical measures that are relevant in a large population of samples, i.e., if you have 100 systems expect a faulty one every 0.57 years. Note the fundamental assumption that failures are independent.

## References

- [ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*, IEEE Transactions on Dependable and Secure Computing **1** (2004), no. 1, 11–33.
- [BBV<sup>+</sup>05] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen, *NonStop<sup>®</sup> Advanced Architecture*, Proc. International Conference on Dependable Systems and Networks (DSN-DCCS), June 2005, pp. 12–21.
- [BS04] W. Bartlett and L. Spainhower, *Commercial fault tolerance: A tale of two systems*, IEEE Transactions on Dependable and Secure Computing **1** (2004), no. 1, 87–96.
- [Gra85] J. Gray, *Why do computers stop and what can be done about it?*, Tech. Report 85.7, Tandem Corp., June 1985, Available from <http://research.microsoft.com/~gray/>.
- [Hof97] G. F. Hoffnagle, *Preface to the special issue on S/390 Parallel Sysplex Cluster*, IBM Systems Journal **36** (1997), no. 2, 170–371, On-line at <http://www.research.ibm.com/journal/sj36-2.html>.
- [Pat02] D. A. Patterson, *An introduction to dependability*, ;login: — The Magazine of the USENIX Association **27** (2002), no. 4, 61–65.
- [Ros00] S. M. Ross, *Introduction to probability models*, 7th ed., Academic Press, 2000.
- [SS98] D. P. Siewiorek and R. S. Swarz, *Reliable computer systems: Design and evaluation*, 3rd ed., A K Peters, 1998.