Dependability

↳ Quality of delivered service such that reliance can justifiably be placed on the service.

Service → is the system behaviour as it is perceived by the user of the system.

Failure → occurs when the delivered service deviates from the service specification that describes the desired service.

Error → It is that part of the system state that can lead to a failure.
Errors can be latent or effective. (active)

Fault → It is the root cause of the error.
It can be human caused or physical.

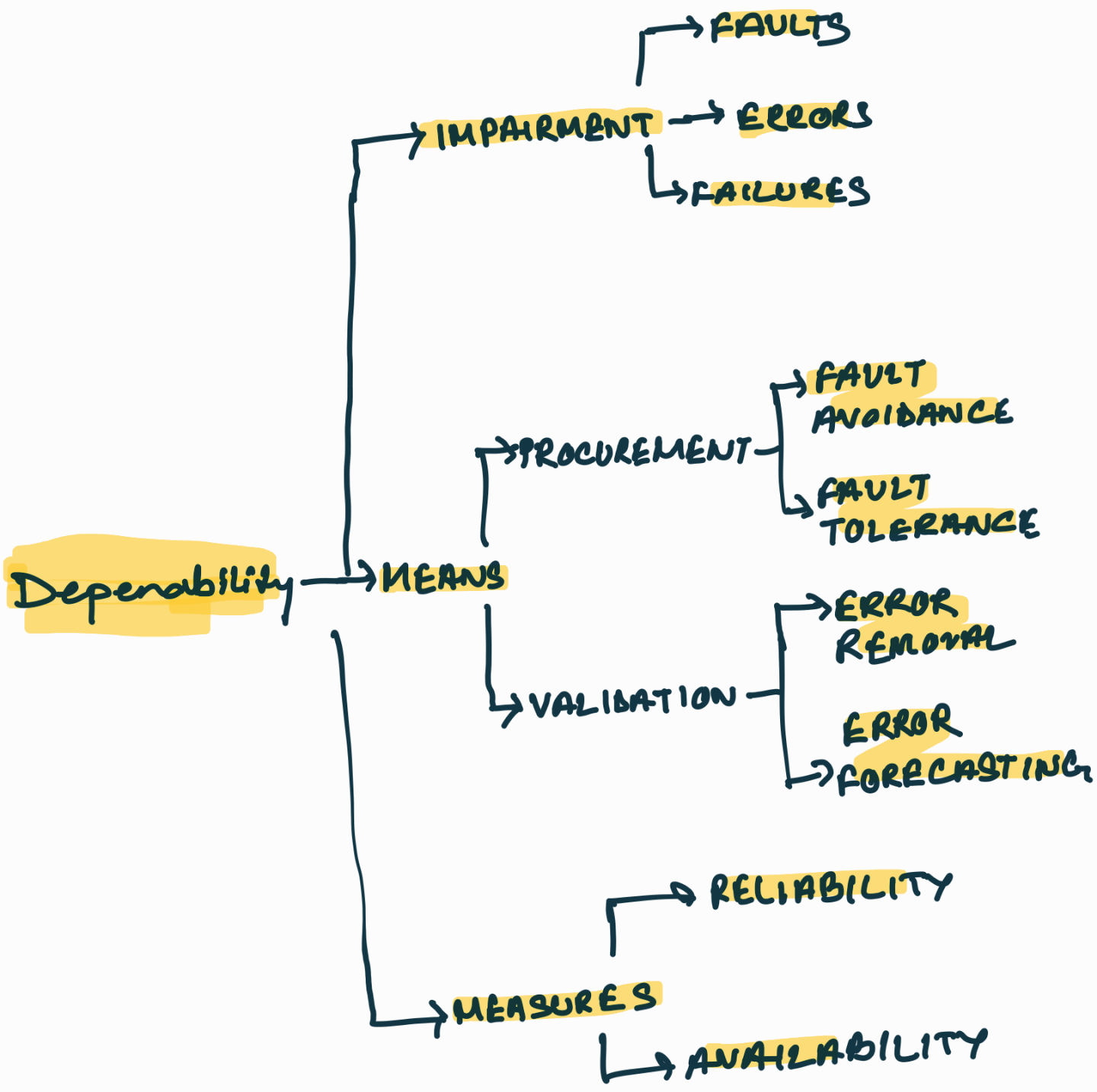
- Faults (eg. programmer's mistake) are the cause of —
- Latent errors (eg. a line of code with a bug in it) which become —
- Effective: if the system reaches a state

errors where the error can manifest (eg. executing the line of code containing the bug) which may cause—

- **Failures**; if the error causes a visible deviation from service from the user's perspective (eg. the program crashes)

Achieving a dependable system involves utilising 4 kinds of methods →

- **Fault Avoidance** : preventing, by construction, fault occurence.
- **Fault Tolerance** : providing, by redundancy, service complying with specification despite faults having occured/ occuring.
- **Error removal** : minimizing, by verification, the presence of latent errors
- **Error forecasting** : estimating, by evaluation, the presence, the creation and the consequences of errors.



Measures of Dependability →

- Availability → readiness for correct service.
- Reliability → continuity of correct service.

- **Safety** → absence of catastrophic consequences on the user & the environment.
- **Integrity** → absence of improper system alteration.
- **Maintainability** → ability for a process to undergo modifications & repairs.

Availability Measures:

✓ **Reliability** : measured in — Mean-Time Between Failures (MTBF)

✓ **Recoverability** : measured in — Mean-Time To Repair (MTTR)

$$\therefore \text{Availability (R)} = \frac{(MTBF)}{(MTBF + MTTR)}$$

Planning for Failures:

for critical system planning for failure is crucial -

- expected that physical devices will break.
- expected that softwares might fail
- identification of allowable failures vs. critical failures is key idea.

eg: DO178B

↳ Software is categorized in terms of critical level of failure.

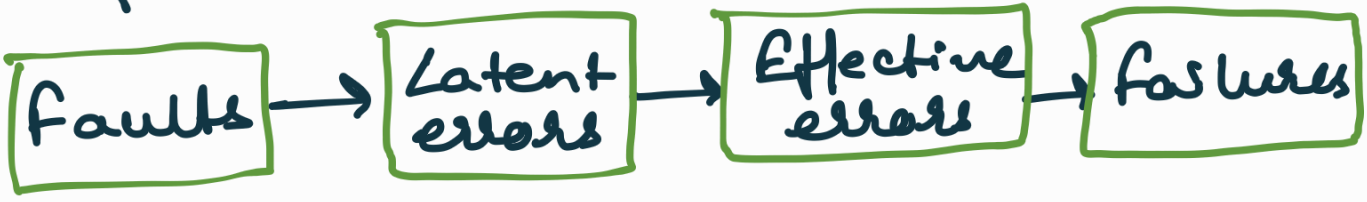
Mapping to Verification Activities:

The rigor of development process is driven by design assurance level (DAL)

Level	Failure Condition
A	Catastrophic
B	Hazardous
C	Major
D	Minor
E	No-effect

Recap:

✓ Software should be dependable.

✓ 

✓ Testing is only part of an effective strategy for error removal

- ✓ Error removal
- ✓ Error forecasting
- ✓ Fault avoidance
- ✓ Fault tolerance

✓ For critical systems, plan for failure

- Testing should include failures.
- Rigor of testing should be driven by criticality.