Design for Test

(while you’re designing)
Design for Test

- Products will have faults ("bugs").
- The purpose of testing is to be able to find those faults, to enable their correction.
- Testability can be designed into the product.
- This is easier than adding it later.
Levels of Testing

- **Unit tests:**
  - Test of methods or functions
  - Tests of entire classes

- **Sub-system tests:**
  - Tests of inter-operation of related functions and classes

- **Integration tests:**
  - Test of entire system
White-Box Test Coverage

Devise tests to cover every aspect of the code:

- Every statement gets exercised
- Every loop gets exercised
- Every variable gets used
Trace Capability

- Generally speaking, it is not a good idea to interleave i/o statements with computational ones:
  - Keep the i/o in a separate section
  - Interact with the outside through well-defined interfaces.

- Consider creating a log facility that can be toggled on or off. The program describes what is doing in the log.
Testing a GUI

- Testing GUI’s can be difficult.
- Design for test can help:
  - As with other i/o, keep event handling separate from computation.
  - Consider creating a synthetic driver that will simulate your GUI but be driven from a script.
Monolithic Design (Bad)

Your Program

screen

the Internet

mouse

keyboard
Layered Design (Good)

Your Product

computation

net handler

output handler

event handler

the Internet

screen

mouse

keyboard
Test Harness

- computation
- simulated net input
- simulated events
- output logger
- log
- script
- the Internet
- mouse
- keyboard
- screen
Log Design

- Repeat inputs and events in the log, so that stimulus/response can be checked.
- Design log for diff-type comparison with other logs.
  - Do not include things like memory addresses in log.
  - Prefer sequence numbers rather than timestamps.
  - Prefer relative quantities to absolute ones.
Log Analysis

- Log analysis can be a separate step
  - Possibly off-line program
- Do not try to put much analysis in the logger itself
  - Introduces error possibilities
  - Want to see the results in relatively unprocessed form
Regression Testing

- Put test drivers in the Makefile and run them with every build (not necessarily every compile).
- Compare output with a known "standard".
- Initially, this can be a previous output of the program. Eventually this aspect must be checked carefully.