(2) Strukturele toetsing

- Nadeel van funksionele toetsing: sommige foute wat verband hou met die struktuur van die kode sal waarskynlik nie gevind word nie.
- Voorbeeld: alle toetsgevalle voer byvoorbeeld net die "then" deel van 'n sekere "if"-opdrag uit en daar is 'n fout in die "else" deel.
- Die *bronkode* word gebruik vir afleiding van toetsgevalle (die *struktuur* van die program word dus in ag geneem en daarom word dit *strukturele* toetsing genoem.

Dekking

- Ideaal: dek alle moontlike uitvoeringspaaie in die kode (onprakties—sommige lusse voer 'n onbekende aantal kere uit).
- Swakste kriterium: alle lyne kode moet ten minste EEN keer uitvoer (onvoldoende—waarom?).
- Sterker kriterium: dek alle beslissingspunte.
- Nog sterker: elke term in elke logiese uitdrukking moet EEN keer waar en vals wees. (waarom?)
- Programme is beskikbaar om dekking te toets. (Hoe word dit gedoen?)

Integrasietoetsing

- Aanvaar komponente is korrek.
- Toets koppelvlakke tussen komponente:
  - word parameters korrek oorgedra? (byvoorbeeld: 'n parameter wat hoogte bo seevlak voorstel kan in een komponent interpreteer word as *meter*, maar as *voet* in 'n ander).
  - word prekondisies van bewerkings bevredig?
  - Uitvoeringspaaie is belangrik: fokus op kontrolevloei vanaf elke module na ander modules *en terug*.
  - Onprakties indien vir baie modules tegelyk gedoen (te veel paaie).

Verskillende werkswyses

- Toets alle modules tegelyk (onprakties, want te veel paaie).
- Onder-na-bo (drywers nodig om in te staan vir ontbrekende komponente)
- Bo-na-onder (vereenvoudigde komponente nodig om in te staan vir ontbrekende komponente).

Onder-na-bo: evaluasie

- Stelsel word opgebou in logiese volgorde (begin met mees basiese bewerkings en vermeerder funksionaliteit deur komponente by te voeg).
- Lae-vlak komponente meer deeglik getoets (dikwels bevat hierdie komponente die meeste foute).
- Gedrag van voltooide stelsel word eers sigbaar teen einde van toetsing—kliënt mag dus eers aan einde van projek besef iets is verkeerd.

Bo-na-onder: evaluasie

- Gebruikerskoppelvlak eerste getoets—kliënt sal vroeg reeds kan sien of iets ontbreek.
- Ekstra moeite om laer-vlak komponente te simuleer aan die begin.
- Dit mag eers laat in die projek ontdek word dat sekere lae-vlak komponente nie effektief genoeg implementeer kan word nie—dit mag 'n verandering van ontwerp noodsaak en heelwat tyd sal verloor word.

Bestuur van toetsing

- Toetsers moet van die begin van 'n projek die ontwerp nagaan om 'n geskikte toetsskedule op te stel (volgorde waarin komponente gekodeer moet word).
- Toetsers moet die spesifikasie nagaan vir foute.
- Toetsers ontwerp toetsgevalle vir funksionele toetse op komponente.
- Programmeerders pas strukturele toetsing toe op hulle eie kode.
- Toetsing beteken verskillende mense moet saamwerk (toetsers, ontwerpers en koderingspan).

Hoeveel toetsing is genoeg?

- Dikwels word dit (verkeerdelik) bepaal deur wanneer die projek afgelewer moet word.
- Besluit *tydens projekbeplanning* hoeveel tyd nodig sal wees vir toetsing.
- Die koste van oorblywende foute sal bepaal hoe deeglik toetsing gedoen moet word:
  - hoe moeilik gaan dit wees om foute later reg te maak?
  - kan seker tipes foute lei tot lewensverlies?
  - selfs al is die kliënte vergewensgesind, moet die goeie reputasie van die ontwikkelaars nie skade lei nie.

Wanneer om te stop

- Wanneer 'n voldoende dekkingsvlak bereik is—gebruik programmatuur om dit te bepaal.
- Indien 'n groot persentasie van geplante foute uitgewys word deur die toetse.
- Uit ondervinding kan die aantal foute in 'n stelsel van gegewe grootte geskat word. Stop dan wanneer die aantal werklike foute ooreenstem met die skatting.
- Wanneer die aantal foute wat per week gevind word afneem onder 'n bepaalde peil—dan raak verdere toetsing te duur.