Software Testing Course
Module 2: Unit Testing

Test Design:
Functional Testing

In this Lecture

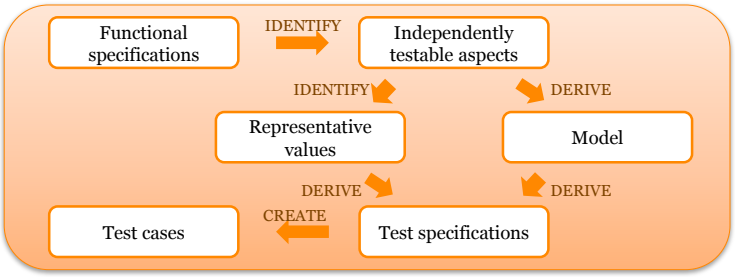
- Functional Testing Basics
- Systematic Functional Test Design
- Input-Space Based Techniques
  - Equivalence Partitioning
  - Boundary Value Analysis
  - Combinatorial Testing
  - Random Testing
- Model-Based Testing

Functional Testing Basics

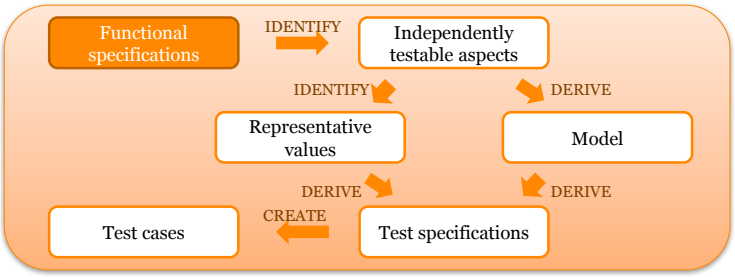
- **Basic idea**: Test cases are derived based on the functional specifications of the software under test
  - What does the specifications say that the software is supposed to do?
  - What does the specifications say that the software is *not* supposed to do (if this is explicitly stated)?
- Implementation of the software under test is not considered in test case design
- Hence, functional testing is also historically known as *Black-Box Testing*

A Systematic Approach to Functional Test Design

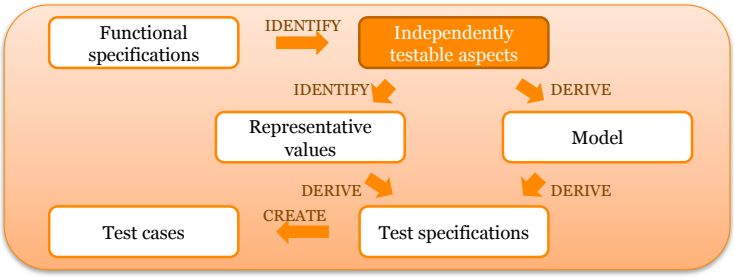
Adapted from "*Software Testing and Analysis*", (Pezzè and Young), and Saarland University course on Software Testing (Fraser and Zeller)



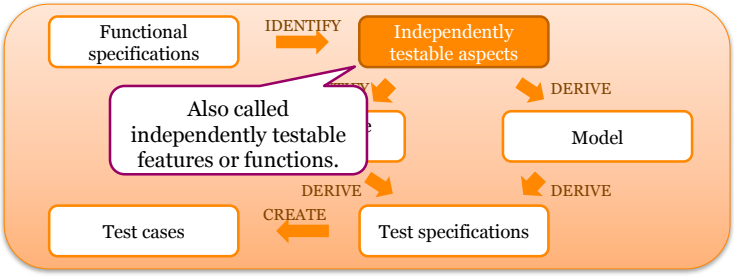
A Systematic Approach to Functional Test Design



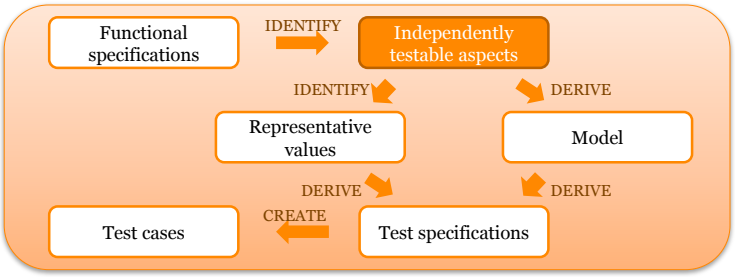
A Systematic Approach to Functional Test Design



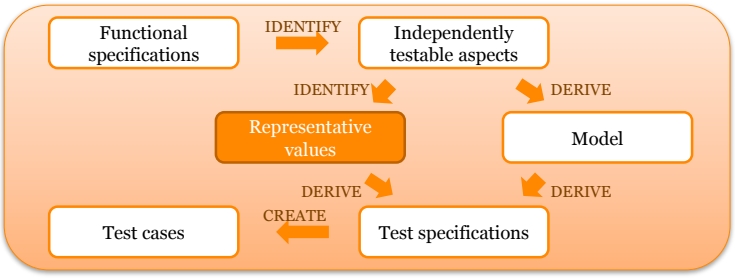
A Systematic Approach to Functional Test Design



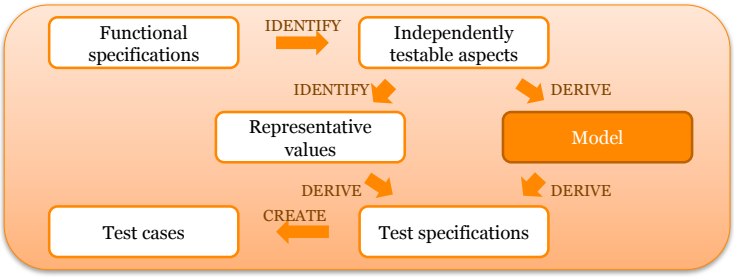
A Systematic Approach to Functional Test Design



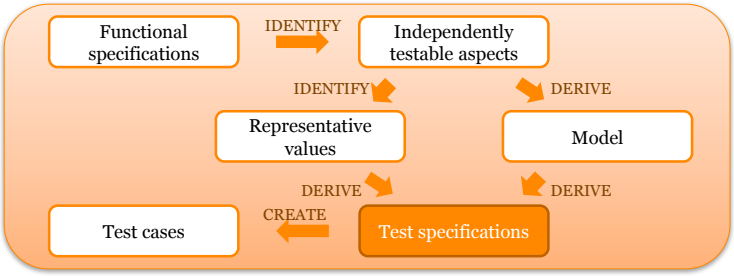
A Systematic Approach to Functional Test Design



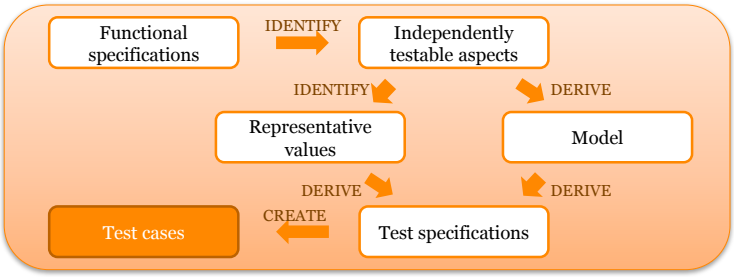
A Systematic Approach to Functional Test Design



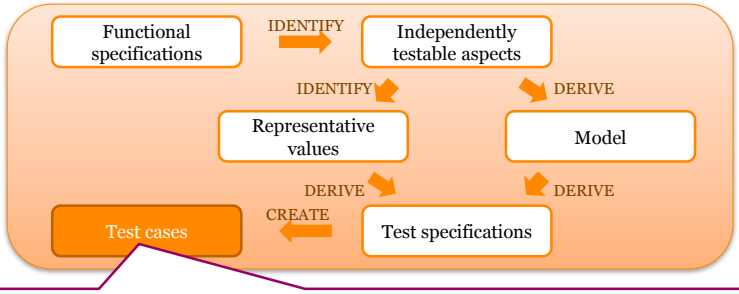
A Systematic Approach to Functional Test Design



A Systematic Approach to Functional Test Design



A Systematic Approach to Functional Test Design



Again, note here that a test case not only requires an **input**, but also an **expected output** in order to be able to provide a verdict.
The expected output is typically derived based on the specifications