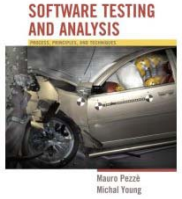
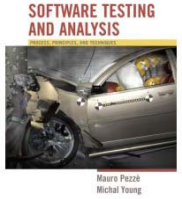
Basic Principles



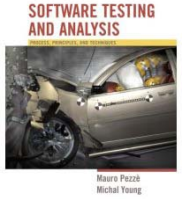
Learning objectives

- Understand the basic principles undelying A&T techniques
- Grasp the motivations and applicability of the main principles



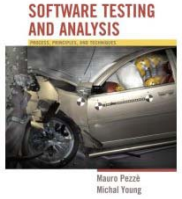
Main A&T Principles

- General engineering principles:
  - Partition: divide and conquer
  - Visibility: making information accessible
  - Feedback: tuning the development process
- Specific A&T principles:
  - Sensitivity: better to fail every time than sometimes
  - Redundancy: making intentions explicit
  - Restriction: making the problem easier



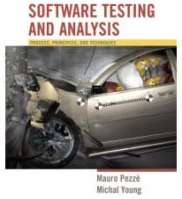
Sensitivity: better to fail every time than sometimes

- Consistency helps:
  - a test selection criterion works better if every selected test provides the same result, i.e., if the program fails with one of the selected tests, it fails with all of them (reliable criteria)
  - run time deadlock analysis works better if it is machine independent, i.e., if the program deadlocks when analyzed on one machine, it deadlocks on every machine



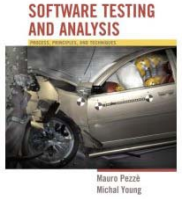
Redundancy: making intentions explicit

- Redundant checks can increase the capabilities of catching specific faults early or more efficiently.
  - Static type checking is redundant with respect to dynamic type checking, but it can reveal many type mismatches earlier and more efficiently.
  - Validation of requirement specifications is redundant with respect to validation of the final software, but can reveal errors earlier and more efficiently.
  - Testing and proof of properties are redundant, but are often used together to increase confidence



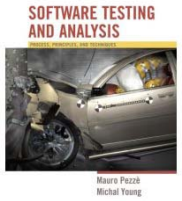
Partition: divide and conquer

- Hard testing and verification problems can be handled by suitably partitioning the input space:
  - both structural and functional test selection criteria identify suitable partitions of code or specifications (partitions drive the sampling of the input space)
  - verification techniques fold the input space according to specific characteristics, grouping homogeneous data together and determining partitions



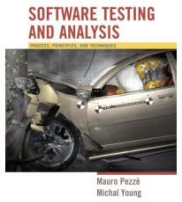
Restriction: making the problem easier

- Suitable restrictions can reduce hard (unsolvable) problems to simpler (solvable) problems
  - A weaker spec may be easier to check: it is impossible (in general) to show that pointers are used correctly, but the simple Java requirement that pointers are initialized before use is simple to enforce.
  - A stronger spec may be easier to check: it is impossible (in general) to show that type errors do not occur at run-time in a dynamically typed language, but statically typed languages impose stronger restrictions that are easily checkable.



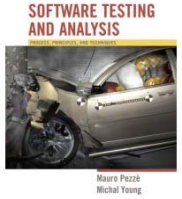
Visibility: Judging status

- The ability to measure progress or status against goals
  - X visibility = ability to judge how we are doing on X, e.g., schedule visibility = “Are we ahead or behind schedule,” quality visibility = “Does quality meet our objectives?”
  - Involves setting goals that can be assessed at each stage of development
    - The biggest challenge is early assessment, e.g., assessing specifications and design with respect to product quality
- Related to observability
  - Example: Choosing a simple or standard internal data format to facilitate unit testing



Feedback: tuning the development process

- Learning from experience: Each project provides information to improve the next
- Examples
  - Checklists are built on the basis of errors revealed in the past
  - Error taxonomies can help in building better test selection criteria
  - Design guidelines can avoid common pitfalls



Summary

- The discipline of test and analysis is characterized by 6 main principles:
  - Sensitivity: better to fail every time than sometimes
  - Redundancy: making intentions explicit
  - Restriction: making the problem easier
  - Partition: divide and conquer
  - Visibility: making information accessible
  - Feedback: tuning the development process
- They can be used to understand andvantages and limits of different approaches and compare different techniques