Basic Principles

Learning objectives for this slideset

- Be able to explain why the basic principles underlying Analysis & Testing (A&T) techniques are useful and important
- Provide examples of the application of the principles.

A&T Principles (General Engineering)

- Partition: divide and conquer *[and ensure the interaction of components is simple]*
- Visibility: making information accessible *[at least to the tester/developer]*
- Feedback: tuning the development process *[in test-driven development, test drives the development]*

Specific A&T Principles

- Sensitivity: a good test selection criterion should be "sensitive" in the sense that whatever particular tests are chosen to verify a requirement they should give you the same result.
- Redundancy/Diversity: can we get "independent" reassurance that a system satisfies a requirement by using redundant/diverse methods
- Restriction/Simplification: making the problem easier by coding in a restricted way OR changing the requirement OR both

Sensitivity

- A test selection criterion is a process by which we select or construct a tests for a given purpose.
- 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)
- For example, suppose a system fails because some fixed area of storage is too small:
  - A selection criterion might choose a range of different tests data some of which overflows the storage.
  - The requirement might say that the system works for all test data.
  - How does this violate sensitivity? How do we fix it?

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 assertions of properties in the code are redundant, but are often used together to increase confidence

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.

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

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 (e.g. using a good JSON editor)

Feedback: tuning the development process

- Learning from experience: Each project provides information to improve the next.
- This works best if you are developing similar products
- Past experience can hamper solving new problems (because we focus excessively on the previous bad experience)
- 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 advantages and limits of different approaches and compare different techniques