

Software Testing and Quality Assurance

Theory and Practice

Chapter 2

Theory of Program Testing

- Basic Concepts in Testing Theory
- Theory of Goodenough and Gerhart
- Theory of Weyuker and Ostrand
- Theory of Gourlay
- Adequacy of Testing
- Limitations of Testing
- Summary

- Testing theory puts emphasis on
 - Detecting defects through program execution
 - Designing test cases from different sources: requirement specification, source code, and input and output domains of programs
 - Selecting a subset of tests cases from the entire input domain
 - Effectiveness of test selection strategies
 - Test oracles used during testing
 - Prioritizing the execution of test cases
 - Adequacy analysis of test cases

- Fundamental Concepts

- Let P be a program, and D be its input domain. Let $T \subseteq D$. $P(d)$ is the result of executing P with input d .

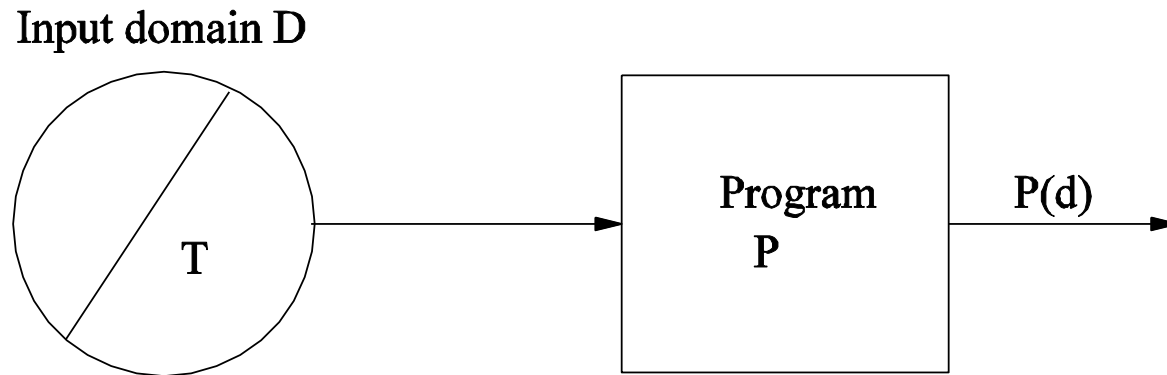


Figure 2.1: Executing a program with a subset of the input domain.

- $OK(d)$: Represents the acceptability of $P(d)$. $OK(d) = true$ iff $P(d)$ is acceptable.
- $SUCCESSFUL(T)$: T is a successful test iff $\forall t \in T, OK(t)$.
- Ideal Test: T is an ideal test if $OK(t), \forall t \in T \Rightarrow OK(d), \forall d \in D$.

- Fundamental Concepts (Contd.)

- *Reliable Criterion*: A test selection criterion C is reliable iff either every test selected by C is successful, or no test selected is successful.
- *Valid Criterion*: A test selection criterion C is valid iff whenever P is incorrect, C selects at least one test set T which is not successful for P .
- Let C denote a set of test predicates. If $d \models D$ satisfies test predicate $c \in C$, then $c(d)$ is said to be true.
- $COMPLETE(T, C) \equiv \underbrace{(\bigwedge_{c \in C} (\bigvee_{t \in T} c(t))) \vee (\bigwedge_{t \in T} (\bigwedge_{c \in C} \neg c(t)))}$

- Fundamental Theorem

- $(\bigvee_{d \models D} (COMPLETE(T, C) \wedge RELIABLE(C) \wedge VALID(C) \wedge SUCCESSFUL(T))) \Rightarrow (\bigwedge_{d \models D} OK(d))$

- Program faults occur due to our
 - inadequate understanding of all conditions that a program must deal with.
 - failure to realize that certain combinations of conditions require special care.
- Kinds of program faults
 - Logic fault
 - Requirement fault
 - Design fault
 - Construction fault
 - Performance fault
 - Missing control-flow paths
 - Inappropriate path selection
 - Inappropriate or missing action
- Test predicate: It is a description of conditions and combinations of conditions relevant to correct operation of the program.

- Conditions for Reliability of a set of test predicates C
 - Every branching condition must be represented by a condition in C.
 - Every potential termination condition must be represented in C.
 - Every condition relevant to the correct operation of the program must be represented in C.
- Drawbacks of the Theory
 - Difficulty in assessing the reliability and validity of a criterion.
 - The concepts of reliability and validity are defined w.r.t. to a program. The goodness of a test should be independent of individual programs.
 - Neither reliability nor validity is preserved throughout the debugging process.

- $d \in D$, the input domain of program P and $T \in D$.
- $OK(P, d) = \text{true}$ iff $P(d)$ is acceptable.
- $SUCC(P, T)$: T is a successful test for P iff for all $t \in T$, $OK(P, t)$.
- *Uniformly valid* criterion: Criterion C is uniformly valid iff
 - $(\forall P) [(\forall d \in D) (OK(P, d)) \Rightarrow (\exists T \in D) (C(T) \wedge SUCC(P, T))]$.
- *Uniformly reliable* criterion: Criterion C is uniformly reliable iff
 - $(\forall P) (\forall T_1, \forall T_2 \in D) [(C(T_1) \wedge C(T_2)) \Rightarrow (SUCC(P, T_1) \Leftrightarrow SUCC(P, T_2))]$.
- *Uniformly Ideal Test Selection*
 - A uniformly ideal test selection criterion for a given specification is both uniformly valid and uniformly reliable.
- A subdomain S is a subset of D .
 - Criterion C is revealing for a subdomain S if whenever S contains an input which is processed incorrectly, then every test set which satisfies C is unsuccessful.
 - $REVEALING(C, S)$ iff

$$(\exists d \in S) (\neg OK(d)) \Rightarrow (\forall T \in S) (C(T) \Rightarrow \neg SUCC(T))$$

- The theory establishes a relationship between three sets of entities
 - specifications, programs and tests.
- Notation
 - P: The set of all programs ($p \in P$)
 - S: The set of all specifications ($s \in S$)
 - T: The set of all tests ($t \in T$)
 - “ $p \text{ ok}(t) s$ ” means the result of testing p with t is judged to be acceptable by s .
 - “ $p \text{ ok}(T) s$ ” means “ $p \text{ ok}(t) s$,” $\forall t \in T$.
 - “ $p \text{ corr } s$ ” means p is correct w.r.t. s .
- A **testing system** is a collection $\langle P, S, T, \text{corr}, \text{ok} \rangle$, where $\text{corr} \subseteq P \times S$ and $\text{ok} \subseteq T \times P \times S$, and $\forall p \in P, s \in S, t \in T (p \text{ corr } s \Rightarrow p \text{ ok}(t) s)$.
- A **test method** is a function $M: P \times S \rightarrow T$
 - Program dependent: $T = M(P)$
 - Specification dependent: $T = M(S)$
 - Expectation dependent

- Power of test methods: Let M and N be two test methods.
 - For M to be **at least as good as** N , we want the following to occur:
 - **Whenever N finds an error, so does M .**
 - (F_M and F_N are sets of faults discovered by test sets produced by test methods M and N , respectively.)
 - (T_M and T_N are test sets produced by test methods M and N , respectively.)
 - Two cases: (a) $T_N \subseteq T_M$ and (b) T_M and T_N overlap

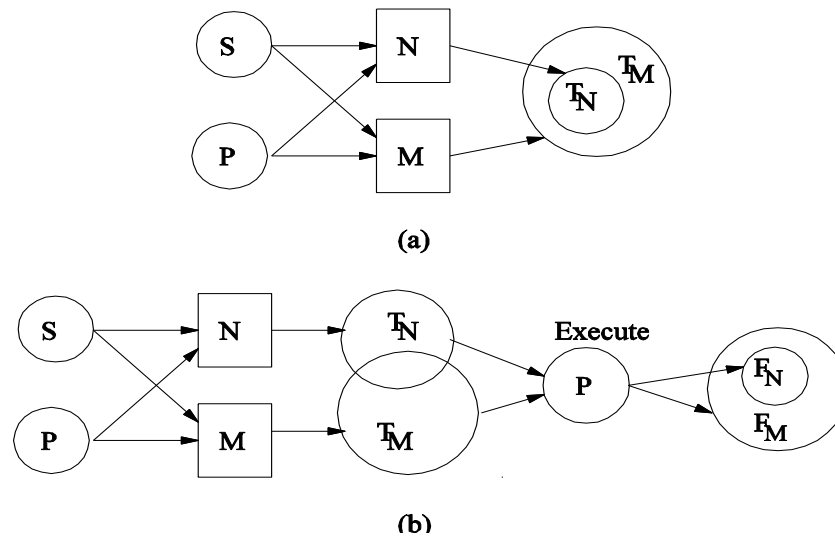


Figure 2.3: Different ways of comparing the power of test methods.

- **Reality:** New test cases, in addition to the planned test cases, are designed while performing testing. Let the test set be T .
- If a test set T does not reveal any more faults, we face a dilemma:
 - P is fault-free. OR
 - T is not good enough to reveal (more) faults.
 - ➔ Need for evaluating the adequacy (i.e. goodness) of T .
- Some *ad hoc* stopping criteria
 - Allocated time for testing is over.
 - It is time to release the product.
 - Test cases no more reveal faults.

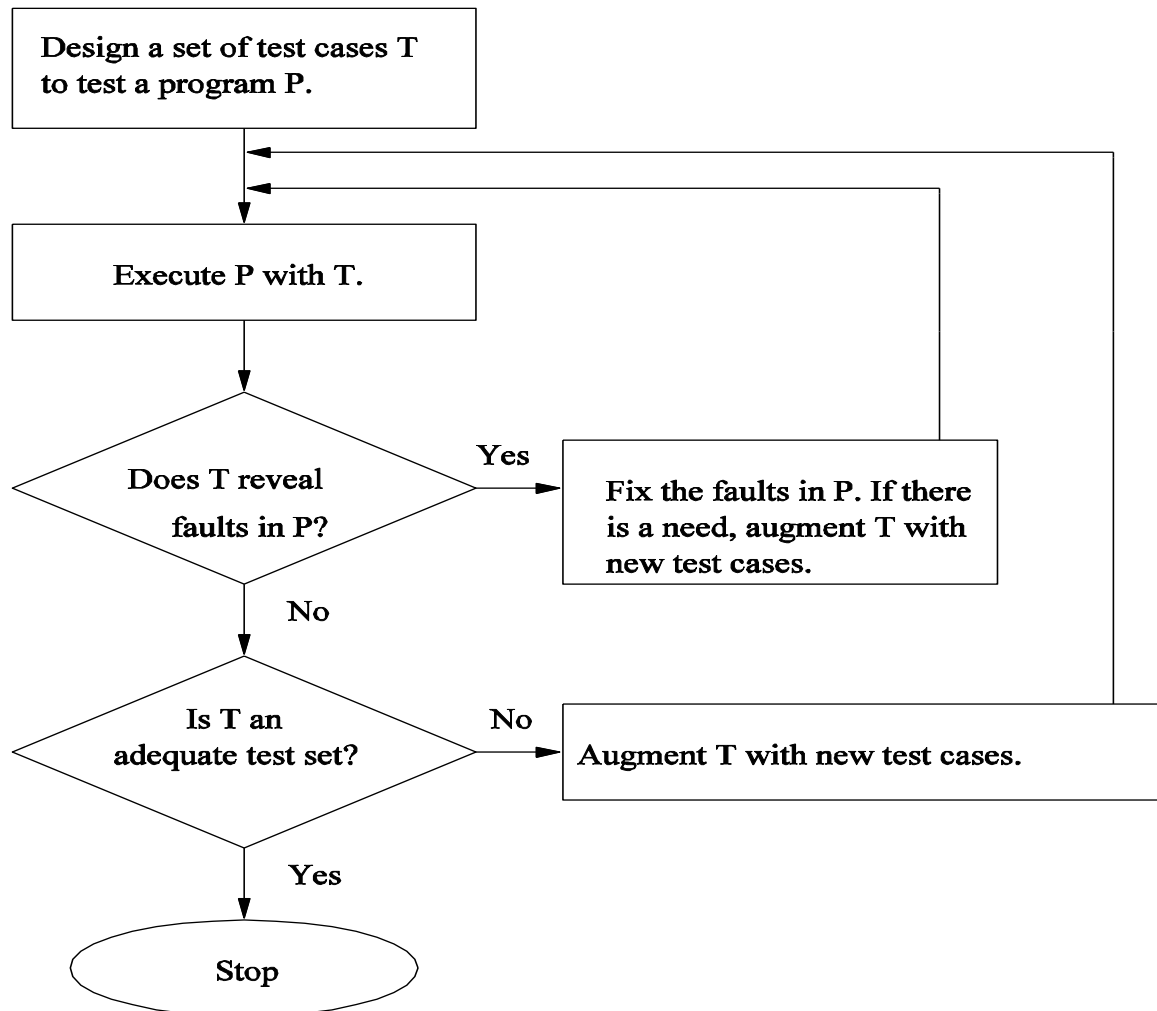


Figure 2.4: Context of applying test adequacy.

- Two practical methods for evaluating test adequacy
 - Fault seeding
 - Program mutation
- Fault seeding
 - Implant a certain number (say, X) of known faults in P , and test P with T .
 - If $k\%$ of the X faults are revealed, T has revealed $k\%$ of the unknown faults.
 - (More in Chapter 13)
- Program mutation
 - A mutation of P is obtained by making a small change to P .
 - Some mutations are faulty, whereas the others are equivalent to P .
 - T is said to be adequate if it causes every faulty mutations to produce unexpected results.
 - (More in Chapter 3)

- Dijkstra's famous observation
 - Testing can reveal the presence of faults, but not their absence.
- Faults are detected by running P with a **small** test set T, where $|T| \ll |D|$, where $| \cdot |$ denotes the “size-of” function and “ \ll ” denoted “much smaller.”
 - Testing with a small test set raises the concern of testing efficacy.
 - Testing with a small test set is less expensive.
- The result of each test must be verified with a **test oracle**.
 - Verifying a program output is not a trivial task.
 - There are **non-testable** programs. A program is non-testable if
 - There is no test oracle for the program.
 - It is too difficult to determine the correct output.

- Theory of Goodenough and Gerhart
 - Ideal test, Test selection criteria, Program faults, Test predicates
- Theory of Weyuker and Ostrand
 - Uniformly ideal test selection
 - Revealing subdomain
- Theory of Gourlay
 - Testing system
 - Power of test methods (“at least as good as” relation)
- Adequacy of Testing
 - Need for evaluating adequacy
 - Methods for evaluating adequacy: fault seeding and program mutation
- Limitations of Testing
 - Testing is performed with a test set T , s.t. $|T| \ll |D|$.
 - Dijkstra’s observation
 - Test oracle problem