This glossary gives definitions for important terms as used in this book. If multiple definitions are given, definition 1 is the one as used in this book (definition 2 and later are found in other places). References within the glossary always refer to definition 1.

**Accident** An unplanned event or series of events resulting in death, injury, occupational illness, damage to or loss of data and equipment or property, or damage to the environment. Synonym of *mishap*.

**Adaptive testing** Executing a sequence of tests in which later tests depend on the outcome of earlier tests.

**Algorithmic debugging** An automated technique that narrows down an error by querying the correctness of intermediate results.

**Anomaly** A program behavior that deviates from expectations based on other runs or other programs. Also known as *incident*.

**Backward slice** The *slice* that may influence a specific statement.

**Bug** 1. Synonym of *defect*. 2. Synonym of *failure*. 3. Synonym of *problem*. 4. Synonym of *infection*.

**Bug report** Synonym of *problem report*.

**Cause** An event preceding the *effect* without which the effect would not have occurred.

**Cause–effect chain** A sequence of events in which each event is a *cause* of the following event.

**Change request** Synonym of *problem report*.

**Circumstance** An event or aspect that may affect the function of a system.

**Code smell** A program property likely to be a *defect*. See also *Defect pattern*.

**Configuration** An arrangement of *circumstances* that affect the function of a system.

**Correction** A *fix* to the code that removes a *defect* from the program. See also *Debugging*. Compare *Workaround*.

**Correctness** The degree to which software is free from *errors* in its specification, design, and coding.

**Crash** The sudden and complete *failure* of a computer system or component.

**Debuggee** The program that is subject to *debugging*.

**Debugger** Tool to facilitate *debugging*.

**Debugging** 1. Relating a *failure* or an *infection* to a *defect* (via an *infection chain*) and subsequent *fixing* of the defect. 2. Removing defects from software. See also *Validation* and *Verification*.

**Deduction** Reasoning from the abstract to the concrete. See also *Static analysis*. Compare *Induction*.

**Defect** An *error* in the program—especially one that can cause an *infection* and thus a *failure*. Also known as *bug* or *fault*. Compare *Flaw*.

**Defect pattern** A pattern matching a *code smell*.

**Delta** Difference between (or change to) *configurations*—especially code, states, or circumstances.

**Delta debugging** An automatic technique that narrows down a cause by running automated *experiments*.

**Diagnosis** A *theory* that explains a *failure*.

**Dynamic analysis** Runtime techniques for *observing* or *inducing* abstractions to the set of values or behaviors seen so far when executing a program. Compare *Static analysis*.

**Effect** An event following the *cause* that would not have occurred without the cause.

**Error** 1. An unwanted and unintended deviation from what is correct, right, or true. 2. Synonym of *infection*. 3. Synonym of *mistake*.

**Exception** An event that causes suspension of normal program operation.

**Experiment** A set of actions and *observations*, performed to verify or falsify a *hypothesis*.

**Experimental analysis** A *dynamic analysis* in which program executions are initiated and/or conducted by the technique, typically within *experiments*.

**Failure** An externally visible *error* in the program behavior. Also known as *malfunction*. See also *Problem*.

**Fallacy** An *error* in logical argument that is independent of the truth of the premises.

**Fault** Synonym of *defect*.

**Feature** An intended property or behavior of a program. Compare *Problem*.

**Fix** A *delta* such that the failure in question no longer occurs. See also *Correction* and *Workaround*.

**Fixing** The act of applying a *fix*.

**Flaw** A *defect* that cannot be attributed to some specific location within the program, but rather its overall design or architecture.

**Forward slice** The *slice* that may be influenced by a specific statement.

**Hanging** Waiting for an event that will never occur.

**Heisenbug** A *failure* that is altered or disappears when one attempts to probe or isolate it.

**Hypothesis** A proposed explanation for a phenomenon. See also *Theory* and *Diagnosis*.

**Incident** Synonym of *anomaly*.

**Induction** Reasoning from the concrete to the abstract. Compare *Deduction*.

**Inductive analysis** A *dynamic analysis* technique that uses *induction* over multiple program executions to find common abstractions.

**Infection** An *error* in the program state—especially one that can cause a *failure*.

**Infection chain** A *cause–effect chain* from *defect* to *failure* along *infections*.

**Invariant** A property that does not change under a set of transformations, such as loop iterations (for loop invariants) or method calls (for class invariants).

**Issue** Synonym of *problem*.

**Malfunction** Synonym of *failure*.

**Mishap** Synonym of *accident*.

**Mistake** A human act or decision resulting in an *error*.

**Observation** Watching something and taking note of anything it does—for instance, observing a program run using a *debugger*.

**Observational analysis** A *dynamic-analysis* technique that *observes* a single program execution to gather findings.

**Oracle** A device that is able to decide any problem of a certain type—in particular, correctness.

**Patch** 1. Synonym of *fix*. 2. A change made directly to an object program without reassembling or recompiling from the source program.

**Problem** A questionable property or behavior of a program. Also known as *issue*. See also *Failure*. Compare *Feature*.

**Problem report** The information required to reproduce a *problem*.

**Regression testing** *Testing* that functionality present in the past is still working in the present.

**Scientific method** A collection of processes that are considered characteristic for the acquisition of new scientific knowledge based on physical evidence.

**Slice** A subset of a program; either a *forward slice* or a *backward slice*.

**Specification** A document that specifies in a complete, precise, and verifiable manner the behavior and other characteristics of a program.

**Static analysis** Compile-time techniques for *deducing* safe and computable approximations to the set of values or behaviors arising dynamically at runtime when executing a program. Compare *Dynamic analysis*.

**Surprise** A property or behavior of a program that cannot be classified as *feature* or *problem*, due to the lack of *specification.*

**Test case** A documentation specifying inputs, predicted results, and a set of execution circumstances for a program.

**Testing** The execution of a program with the intent to produce some *problem*—especially a *failure*. In the context of *debugging*, testing is typically intended to produce a given problem.

**Theory** A *hypothesis* offering valid predictions that can be *observed.*

**Validation** Producing evidence that the program meets its *specification* for a specific intended use. In other words, "You built the right thing." Compare *Verification*.

**Verification** Proving the absence of *defects* with regard to a *specification*. In other words, "You built it right." Compare *Validation*.

**Workaround** A *fix* to the code where the *defect* remains in the program. Compare *Correction*.

"And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.

– Lewis Carroll
*Through the Looking-Glass* (1872)